

i.MX RT500 Low-Power Crossover MCU Reference Manual



Contents

Preface About This Manual.....	36
Chapter 1 Introduction.....	45
1.1 Introduction.....	45
1.2 Master ID Assignments.....	47
1.3 Features and benefits.....	48
1.4 Target Applications.....	52
1.5 Endianness Support	52
Chapter 2 Memory Maps.....	53
2.1 Memory system overview.....	53
2.2 System memory map.....	54
2.3 Shared RAM Address Map.....	55
2.4 APB0 (32 Peripheral slots)	57
2.5 APB1 (32 Peripheral slots)	58
2.6 AHB Peripherals	60
Chapter 3 Interrupt, DMA Events, and DSP Assignments.....	64
3.1 Nested vector interrupts.....	64
3.2 DMA request configuration.....	67
3.3 DMA request assignments.....	68
3.4 DMA trigger input multiplexing.....	69
3.5 DMA input trigger assignments.....	70
3.6 DMA output triggers.....	71
3.7 Fusion DSP interrupt configuration.....	72
3.8 Fusion DSP interrupt input multiplexer.....	73
3.9 Fusion DSP interrupt priority assignments.....	74
Chapter 4 System Controller (SYSCTL).....	76
4.1 Chip-specific SYSCTL information.....	76
4.2 Overview.....	76
4.3 Signals.....	76
4.4 Functional description.....	76
4.5 Memory map and register definition.....	81
Chapter 5 Reset Controller (RSTCTL).....	218
5.1 Chip-specific RSTCTL information.....	218
5.2 Overview.....	218
5.3 Functional description.....	218
5.4 Signals.....	218
5.5 Memory map and register definition.....	219
Chapter 6 Clock Control (CLKCTL).....	257
6.1 Chip-specific CLKCTL information.....	257

6.2 Overview.....	265
6.3 Signals.....	265
6.4 Functional description.....	266
6.5 Memory map and registers.....	270
Chapter 7 Power Control API.....	472
7.1 Overview.....	472
7.2 Functional Description.....	472
Chapter 8 Power Management Controller (PMC).....	484
8.1 Chip-specific PMC information.....	484
8.2 Overview.....	484
8.3 Features.....	484
8.4 Functional Description.....	485
8.5 Memory Map and register definition.....	495
Chapter 9 Processor Configurations.....	513
9.1 ARM Cortex-M33 Details.....	513
9.2 Fusion DSP Details	514
Chapter 10 Fusion DSP.....	515
10.1 Overview.....	515
10.2 Memory Map.....	515
10.3 Reset behavior.....	516
10.4 Security.....	516
10.5 Interrupts.....	516
10.6 DMA.....	517
10.7 Inter-CPU communications.....	517
Chapter 11 AXI Switch.....	518
11.1 Overview.....	518
11.2 Memory map and register definition.....	518
Chapter 12 IO PAD Controller (IOPCTL).....	525
12.1 Chip-specific IOPCTL information.....	525
12.2 Overview.....	525
12.3 Signals.....	527
12.4 Memory map and register definition.....	527
12.5 RT500 pin multiplexing table (FOWLP).....	536
12.6 RT500 pin multiplexing table (WLCSP).....	553
Chapter 13 Peripheral Input Mux Controller (INPUTMUX).....	563
13.1 Chip-specific INPUTMUX information.....	563
13.2 Overview.....	563
13.3 Signals.....	564
13.4 Configuration.....	566
13.5 Memory map and register definition.....	566

Chapter 14 General Purpose Input/Output (GPIO)	662
14.1 Chip-specific GPIO information	662
14.2 Overview	663
14.3 Application information	663
14.4 Functional description	663
14.5 Memory map and register definition	665
Chapter 15 Pin Interrupt and Pattern Match (PINT)	700
15.1 Chip-specific PINT information	700
15.2 Overview	701
15.3 Signals	702
15.4 Functional description	702
15.5 Application information	704
15.6 Memory Map and register definition	707
Chapter 16 DMA Controller	722
16.1 Chip-specific DMA controller information	722
16.2 Overview	728
16.3 Signals	729
16.4 Functional description	729
16.5 Application information	734
16.6 Memory map and register definition	734
Chapter 17 SmartDMA	767
17.1 Introduction	767
17.2 External Signal Description	767
Chapter 18 Non-Secure Boot ROM	768
18.1 Overview	768
18.2 Functional Description	768
18.3 Basic boot ROM function on RT500	778
18.4 Boot modes	778
18.5 Boot ROM Status Error Codes	811
18.6 External Memory Support	814
18.7 RT500 ISP and IAP	832
Chapter 19 Secure Boot ROM	905
19.1 Introduction	905
19.2 Data structures	908
19.3 Plain image structure	918
19.4 Secure boot images	919
19.5 Firmware Update ROM support using SB2 file	926
19.6 OTFAD	928
19.7 ROM TrustZone support	929
19.8 Secure Boot usage	939
Chapter 20 Security	947
20.1 Introduction	947
20.2 AES engine	948

20.3 SHA.....	948
20.4 Digital signatures.....	949
20.5 Hash-based Message Authentication Code (HMAC).....	949
20.6 RNG.....	949
20.7 UUID.....	949
20.8 DICE.....	949
20.9 On-The-Fly AES Decryption (OTFAD).....	950
20.10 RT500 Life cycle state.....	950
Chapter 21 TrustZone Support.....	955
21.1 Overview.....	955
21.2 Example Configuration.....	955
Chapter 22 AHB Secure Controller (AHBSC).....	962
22.1 Chip-specific AHB Secure Controller information.....	962
22.2 Overview.....	962
22.3 Functional description.....	963
22.4 Memory Map and Registers.....	963
Chapter 23 PowerQuad DSP Coprocessor and Accelerator (PowerQuad).....	1178
23.1 Chip-specific PowerQuad information.....	1178
23.2 Overview.....	1178
23.3 Functional description.....	1179
23.4 External signals.....	1193
23.5 Initialization.....	1193
23.6 Application information.....	1194
23.7 Memory map and register definition.....	1209
Chapter 24 Casper.....	1234
24.1 Chip-specific Casper information.....	1234
24.2 Overview.....	1234
24.3 Functional description.....	1236
24.4 Memory map and register definition.....	1238
Chapter 25 Physically Unclonable Function (PUF).....	1254
25.1 Chip-specific PUF controller information.....	1254
25.2 Overview.....	1254
25.3 Functional description.....	1255
25.4 Initialization.....	1262
25.5 Software development.....	1262
25.6 Memory map and register definition.....	1266
Chapter 26 HASH-AES.....	1291
26.1 Chip-specific HASH-AES information.....	1291
26.2 AES engine	1291
26.3 HASH	1296
26.4 Memory map and register definition.....	1299

Chapter 27 Standalone TRNG	1319
27.1 Overview	1319
27.2 Functional Description	1320
27.3 Register Interface Usage	1320
27.4 Operation	1321
27.5 Other Applications	1325
27.6 TRNG register descriptions	1325
Chapter 28 Real Time Clock (RTC)	1371
28.1 Chip-specific RTC information	1371
28.2 Overview	1372
28.3 Functional description	1373
28.4 Signals	1374
28.5 Initialization	1374
28.6 Memory map and register definition	1374
Chapter 29 CPU system tick timer (SYSTICK)	1383
29.1 Overview	1383
29.2 Functional description	1383
29.3 Example timer calculations	1384
29.4 Memory map and register description	1384
Chapter 30 Standard counter/timers (CTIMER)	1389
30.1 Chip-specific CTIMER information	1389
30.2 Introduction	1389
30.3 Signals	1391
30.4 Application information	1392
30.5 Functional description	1392
30.6 Memory map and Register definition	1394
Chapter 31 OS Event Timer (OSTIMER)	1412
31.1 Chip-specific OSTIMER information	1412
31.2 Overview	1412
31.3 Functional description	1414
31.4 Initialization	1415
31.5 Memory map and register definition	1415
Chapter 32 SCTIMER	1429
32.1 Chip-specific SCTimer information	1429
32.2 Overview	1431
32.3 Functional description	1433
32.4 External signals	1438
32.5 Initialization	1438
32.6 Application information	1438
32.7 Memory map and register definition	1442
Chapter 33 Micro-Tick Timer (UTICK)	1484
33.1 Chip-specific UTICK information	1484
33.2 Overview	1484

33.3 Functional description.....	1485
33.4 Signals.....	1486
33.5 Initialization.....	1486
33.6 Memory map and register definition.....	1486
Chapter 34 Multi-Rate Timer (MRT).....	1493
34.1 Chip-specific MRT information.....	1493
34.2 Overview.....	1493
34.3 Functional description.....	1494
34.4 Signals.....	1495
34.5 Initialization.....	1495
34.6 Memory map and register descriptions.....	1495
Chapter 35 Windowed Watchdog Timer (WWDT).....	1505
35.1 Chip-specific WWDT information.....	1505
35.2 Overview.....	1506
35.3 Functional description.....	1508
35.4 Initialization.....	1511
35.5 Memory map and register definition.....	1511
Chapter 36 Frequency Measurement (FREQMEASURE).....	1519
36.1 Chip-specific Frequency Measurement information.....	1519
36.2 Overview.....	1520
36.3 Functional description.....	1520
36.4 Signals.....	1520
36.5 Memory map and register definition.....	1521
Chapter 37 Cyclic Redundancy Check (CRC).....	1524
37.1 Chip-specific CRC information.....	1524
37.2 Overview.....	1524
37.3 Functional description.....	1525
37.4 Signals.....	1525
37.5 Application information.....	1525
37.6 Memory Map and register definition.....	1526
Chapter 38 On-Chip OTP Controller (OCOTP_CTRL).....	1531
38.1 Chip-specific OCOTP_CTRL information.....	1531
38.2 Overview.....	1531
38.3 Functional Description.....	1533
38.4 Initialization.....	1534
38.5 Memory map and register definition.....	1534
Chapter 39 FlexSPI.....	1548
39.1 Chip-specific FLEXSPI information.....	1548
39.2 Overview.....	1550
39.3 Glossary for FlexSPI module.....	1551
39.4 External signals.....	1552
39.5 Functional description.....	1553
39.6 Initialization.....	1605

39.7 Application information.....	1605
39.8 Memory Map and register definition.....	1623
39.9 AHB Memory Map definition.....	1724
Chapter 40 On-the-Fly AES Decryption Module (OTFAD).....	1725
40.1 Chip-specific OTFAD information.....	1725
40.2 Overview.....	1725
40.3 Functional description.....	1728
40.4 External signals.....	1733
40.5 Initialization.....	1734
40.6 Application information.....	1734
40.7 Register descriptions and data organization.....	1738
Chapter 41 Peripheral Bridge (AIPS_Lite).....	1751
41.1 Chip-specific AIPS-Lite information.....	1751
41.2 Overview.....	1751
41.3 Functional description.....	1751
41.4 External signals.....	1752
41.5 Memory map and register definition.....	1752
Chapter 42 Messaging Unit (MU).....	1753
42.1 Chip-specific MU information.....	1753
42.2 Overview.....	1753
42.3 Functional Description.....	1755
42.4 External Signals.....	1758
42.5 Application Information.....	1758
42.6 Register Definition.....	1765
Chapter 43 Semaphores2 (SEMA42).....	1789
43.1 Chip-specific SEMA42 information.....	1789
43.2 Introduction.....	1789
43.3 Functional description.....	1790
43.4 Memory map/register definition.....	1793
43.5 External signals.....	1796
43.6 Initialization.....	1796
43.7 Glossary.....	1796
Chapter 44 Cache Controller.....	1797
44.1 Chip-specific CACHE64 information.....	1797
44.2 Overview.....	1797
44.3 Functional description.....	1798
44.4 External signals.....	1803
44.5 Initialization.....	1803
44.6 Memory map and registers.....	1804
Chapter 45 CACHE64 Policy Select (POLSEL).....	1813
45.1 Chip-specific CACHE64 POLSEL information.....	1813
45.2 Overview.....	1813
45.3 Functional description.....	1813

45.4 Application information.....	1814
45.5 Memory map and register definition.....	1815
Chapter 46 LCD Interface (LCDIF).....	1819
46.1 Chip-specific LCDIF information.....	1819
46.2 Overview.....	1819
46.3 Functional Description.....	1822
46.4 Initialization.....	1825
46.5 Memory Map and Register Definition.....	1828
Chapter 47 Vector Graphics Processing Unit (GPU).....	1869
47.1 Chip-specific GPU information.....	1869
47.2 Overview.....	1869
47.3 Features.....	1869
47.4 Block Diagram.....	1871
47.5 GPU Power Management Features.....	1872
47.6 Software API.....	1872
Chapter 48 MIPI DSI (MIPI_DSI) Interface.....	1873
48.1 Chip-specific MIPI-DSI information.....	1873
48.2 Overview.....	1873
48.3 DSI Host Controller Core.....	1875
48.4 DSI D-PHY.....	1885
48.5 DPHY Programming.....	1894
48.6 Packet Data and Pixel Formats.....	1895
48.7 MIPI_DSI_HOST Memory Map/Register Definition.....	1896
Chapter 49 DMIC Subsystem.....	1945
49.1 Chip-specific DMIC information.....	1945
49.2 Overview.....	1947
49.3 Functional description.....	1948
49.4 Signals.....	1954
49.5 Initialization.....	1956
49.6 Memory map and register definition.....	1957
Chapter 50 Ultra Secured Digital Host Controller (uSDHC).....	1983
50.1 Chip-specific uSDHC information.....	1983
50.2 Introduction.....	1983
50.3 External Signals.....	1987
50.4 Clocks.....	1988
50.5 Functional Description.....	1989
50.6 Initialization/Application of uSDHC.....	2006
50.7 Commands for MMC/SD/SDIO.....	2023
50.8 Software Restrictions.....	2030
50.9 uSDHC memory map/register definition.....	2031
Chapter 51 FlexIO.....	2171
51.1 Chip-specific FLEXIO information.....	2171
51.2 Overview.....	2171

51.3 Functional description.....	2173
51.4 Application Information.....	2182
51.5 FlexIO Signal Descriptions.....	2201
51.6 Memory Map and Registers.....	2201
51.7 Glossary.....	2249
Chapter 52 Improved Inter-Integrated Circuit (I3C).....	2250
52.1 Chip-specific I3C information.....	2250
52.2 Overview.....	2250
52.3 Functional description.....	2252
52.4 External signals.....	2258
52.5 Initialization.....	2258
52.6 Application information.....	2260
52.7 I3C register descriptions.....	2263
Chapter 53 Flexcomm Serial Communication.....	2336
53.1 Chip-specific Flexcomm information.....	2336
53.2 Flexcomm Overview.....	2347
53.3 USART.....	2353
53.4 Serial Peripheral Interfaces (SPI).....	2393
53.5 Inter-integrated Circuit (I2C)	2433
53.6 Inter-IC Sound (I2S).....	2469
Chapter 54 Universal Serial Bus 2.0 HS Host Controller.....	2511
54.1 Chip-specific USB2.0 HS Host Controller information.....	2511
54.2 Overview.....	2511
54.3 Signals.....	2512
54.4 Functional description.....	2512
54.5 Initialization.....	2522
54.6 Application information.....	2522
54.7 Memory map and register definition.....	2523
Chapter 55 Universal Serial Bus 2.0 HS Device Controller.....	2547
55.1 Chip-specific USB2.0 Device Controller information.....	2547
55.2 About this module.....	2547
55.3 Signals.....	2549
55.4 Functional description.....	2549
55.5 Initialization.....	2557
55.6 Application information.....	2557
55.7 Memory Map and register definition.....	2558
Chapter 56 USB Device Charger Detection Module (USBHSDCD).....	2575
56.1 Chip-specific USBDCD information.....	2575
56.2 Overview.....	2575
56.3 Functional Description.....	2576
56.4 External Signals.....	2588
56.5 Initialization Information.....	2589
56.6 Application Information.....	2589
56.7 References.....	2590
56.8 Glossary.....	2590

56.9 Memory map/Register definition.....	2591
Chapter 57 Universal Serial Bus 2.0 Integrated PHY (USBPHY).....	2603
57.1 Chip-specific USB2.0 PHY information.....	2603
57.2 Overview.....	2603
57.3 Functional description.....	2604
57.4 Memory Map and Register Definition.....	2610
Chapter 58 Analog-to-Digital Converter (ADC).....	2716
58.1 Chip-specific ADC information.....	2716
58.2 Overview.....	2718
58.3 Functional description.....	2720
58.4 External signals.....	2728
58.5 Initialization.....	2729
58.6 ADC register descriptions.....	2729
Chapter 59 Analog Comparator (ACMP).....	2767
59.1 Chip-specific ACMP information.....	2767
59.2 Introduction.....	2767
59.3 Features.....	2767
59.4 CMP DAC and ANMUX diagram.....	2769
59.5 CMP block diagram.....	2769
59.6 CMP pin descriptions.....	2771
59.7 CMP Functional Modes.....	2771
59.8 CMP Functional description.....	2777
59.9 CMP interrupts.....	2780
59.10 CMP DMA Support.....	2781
59.11 DAC functional description.....	2781
59.12 DAC resets.....	2781
59.13 DAC clocks.....	2782
59.14 DAC interrupts.....	2782
59.15 Trigger mode.....	2782
59.16 Memory Map Register Definitions.....	2783
Chapter 60 Debug Subsystem.....	2800
60.1 Overview.....	2800
60.2 Functional description.....	2801
60.3 Debug Mailbox protocol.....	2805
60.4 Reset handling.....	2808
60.5 Mailbox commands.....	2808
60.6 Debug authentication.....	2812
60.7 Signals.....	2825
60.8 Memory map and register definition.....	2826
Chapter 61 JTAG.....	2831
61.1 Introduction.....	2831
61.2 External signal description.....	2832
61.3 Functional description.....	2833
61.4 Initialization Application information.....	2837
61.5 Register description.....	2838

Appendix A Revision History.....2840
 A.1 Reference manual changes..... 2840

Legal information..... 2843

Figures

Figure 1. Register Field Conventions.....	38
Figure 1. i.MX RT500 Crossover MCU Family Block Diagram.....	46
Figure 2. Detailed Block Diagram.....	47
Figure 3. Shared SRAM.....	53
Figure 4. AXI-to-RAM Architecture.....	54
Figure 5. DMA trigger multiplexing.....	69
Figure 6. DSP Interrupt Configuration.....	73
Figure 7. I2S Shared signal connections for each Flexcomm Interface (no output sharing).....	78
Figure 8. Shared signal source selection and control.....	78
Figure 9. Example connection to an I2S bidirectional codec.....	79
Figure 10. I2S signal sharing with multiple slave receivers.....	79
Figure 11. I2S signal sharing example showing one master and multiple slave transmitters.....	80
Figure 12. I2S signal sharing example showing one master with mixed transmitters and receivers.....	80
Figure 13. I2S bridging.....	81
Figure 14. RT500 clock diagram page 1.....	260
Figure 15. RT500 clock diagram page 2.....	261
Figure 16. RT500 clock diagram page 3.....	262
Figure 17. RT500 clock diagram page 4.....	263
Figure 18. RT500 clock diagram page 5.....	264
Figure 19. CLKCTL block diagram.....	265
Figure 20. Main / System PLL.....	269
Figure 21. Audio PLL.....	269
Figure 22. Always-On Domain.....	492
Figure 23. Simplified Fail Safe Pin Diagram.....	526
Figure 24. Simplified High Speed Pin Diagram.....	527
Figure 25. Generic input Multiplexing.....	564
Figure 26. Pin interrupt multiplexing.....	565
Figure 27. DMA request configuration.....	565
Figure 28. DMA trigger multiplexing.....	566
Figure 29. Pin Interrupt diagram.....	701
Figure 30. Pattern match bit slice.....	702
Figure 31. Pattern match engine connections.....	703
Figure 32. Pattern match engine examples: sticky edge detect.....	706
Figure 33. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true.....	706
Figure 34. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false.....	707
Figure 35. DMA block diagram.....	728
Figure 36. Interleaved transfer in a single buffer.....	731
Figure 37. Top-Level Boot Process.....	774
Figure 38. FLEXSPI FLASH Boot Flow.....	788
Figure 39. FLEXSPI image remap(Offset=0x200000).....	789
Figure 40. FLEXSPI Boot Image Selection.....	791
Figure 41. FLEXSPI Dual Image Ping Pong Boot Flow.....	792

Figure 42. Set the FLEXSPI config parameter in RAM.....	797
Figure 43. Config the FLEXSPI using the parameter stored in RAM.....	797
Figure 44. Read the NOR flash via blhost tool.....	797
Figure 45. Program the boot image into NOR flash via blhost tool.....	798
Figure 46. read the boot image back via blhost tool.....	798
Figure 47. Store the config FCB parameter into the RAM.....	798
Figure 48. FCB generate and program into flash at offset 0x08000400.....	799
Figure 49. Read the FCB back via blhost tool.....	799
Figure 50. eMMC Boot Flow.....	805
Figure 51. Recovery boot flow.....	809
Figure 52. Serial Boot Flow.....	810
Figure 53. Command with no data phase.....	834
Figure 54. Command with incoming data phase.....	835
Figure 55. Command with outgoing data phase.....	836
Figure 56. Ping Packet Protocol Sequence.....	837
Figure 57. Host read ACK(5a,a1) packet timing restriction.....	845
Figure 58. Host target ACK(5a,a1) packet timing restriction.....	845
Figure 59. Protocol Sequence for GetProperty Command.....	846
Figure 60. Parameters for SetProperty Command.....	848
Figure 61. Protocol Sequence for FlashEraseAll Command.....	849
Figure 62. Protocol Sequence for FlashEraseRegion Command.....	851
Figure 63. Command sequence for ReadMemory.....	852
Figure 64. Protocol Sequence for WriteMemory Command.....	853
Figure 65. Protocol Sequence for FillMemory Command.....	855
Figure 66. Protocol sequence for Call command.....	856
Figure 67. Protocol Sequence for Reset Command.....	857
Figure 68. Protocol Sequence for FlashProgramOnce Command.....	858
Figure 69. Protocol Sequence for FlashReadOnce Command.....	859
Figure 70. Protocol Sequence for ConfigureMemory Command.....	860
Figure 71. Host reads an ACK from target via UART.....	866
Figure 72. Host reads a ping response from target via UART.....	866
Figure 73. Host reads a command response from target via UART.....	867
Figure 74. Physical Interface for SPI ISP.....	868
Figure 75. Host reads ACK from target via SPI.....	869
Figure 76. Host reads response from target via SPI.....	869
Figure 77. Host reads ACK packet from target via I2C.....	871
Figure 78.	872
Figure 79. ROM API Tree structure.....	875
Figure 80. RT500 Boot flow diagram.....	907
Figure 81. Signed XIP image.....	921
Figure 82. Signed LoadToRAM image(signed non-XIP image).....	922
Figure 83. Encrypted and Signed LoadToRAM image.....	923
Figure 84. Security system	947
Figure 85. Key storage.....	948
Figure 86. Example of Cortex-M33 device with Security Extension.....	956

Figure 87. TrustZone Isolation after basic SAU configuration.....	958
Figure 88. TrustZone Isolation after canonical SAU configuration.....	959
Figure 89. TrustZone Isolation after canonical form of SAU and AHB Secure Controller Configuration.....	960
Figure 90. TrustZone Isolation after Combined SAU and AHB Secure Controller Configuration.....	961
Figure 91. PowerQuad block diagram.....	1179
Figure 92. PowerQuad pipeline example.....	1180
Figure 93. Second-order IIR filter.....	1183
Figure 94. Biquad pipeline example.....	1184
Figure 95. PowerQuad using private RAM as the 128-bit wide scratch pad.....	1186
Figure 96. Radix-8 butterfly structure.....	1187
Figure 97. FIR signal flow.....	1191
Figure 98. Rotation and vectoring mode of the CORDIC algorithm.....	1192
Figure 99. CASPER block diagram.....	1235
Figure 100. Possible flows, states, and actions.....	1257
Figure 101. KC header format.....	1260
Figure 102. Key byte order on the Key interface for 128-bit key.....	1260
Figure 103. 2x512-bit buffers are used for AES.....	1293
Figure 104. Showing extra values stored in message 2 for ICB.....	1295
Figure 105. SHA input data block.....	1296
Figure 106. Standalone TRNG Block Diagram.....	1320
Figure 107. Chip-specific RTC block diagram.....	1371
Figure 108. Real-time clock (RTC) block diagram.....	1372
Figure 109.	1383
Figure 110. 32-bit counter/timer block diagram.....	1391
Figure 111. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled.....	1393
Figure 112. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.....	1393
Figure 113. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.....	1394
Figure 114. Block diagram.....	1413
Figure 115. Block diagram.....	1432
Figure 116. Event selection.....	1436
Figure 117. SCT configuration example: pictorial representation.....	1440
Figure 118. SCT event configuration and selection registers.....	1443
Figure 119. Match logic.....	1444
Figure 120. Capture logic.....	1445
Figure 121. Output slice n.....	1446
Figure 122. SCT interrupt generation.....	1446
Figure 123. Block diagram.....	1485
Figure 124. Block diagram.....	1493
Figure 125. WDT clocking.....	1506
Figure 126. Windowed Watchdog Timer block diagram	1507
Figure 127. Correct watchdog feed with windowed mode enabled.....	1509
Figure 128. Early watchdog feed with windowed mode enabled.....	1510
Figure 129. Watchdog warning interrupt.....	1510
Figure 130. CRC block diagram.....	1524
Figure 131. OCOTP_CTRL Block Diagram.....	1532

Figure 132. FlexSPI block diagram.....	1550
Figure 133. FlexSPI connection diagram with two devices.....	1554
Figure 134. AHB access memory map in individual and parallel mode.....	1556
Figure 135. LUT, sequence and instruction structure.....	1561
Figure 136. Flash access sequence example (SDR single I/O read sequence).....	1567
Figure 137. Flash access sequence example (SDR quad I/O read sequence).....	1567
Figure 138. Flash access sequence example (DDR quad I/O read sequence).....	1568
Figure 139. Flash access sequence example (data learning).....	1568
Figure 140. Flash access sequence example (data learning) (Channel B).....	1568
Figure 141. HyperBus device read transaction with single latency count.....	1569
Figure 142. HyperBus device read transaction with additional latency count.....	1570
Figure 143. HyperBus device write transaction with single latency count.....	1570
Figure 144. HyperBus device write transaction with additional latency count.....	1571
Figure 145. Reading IP RX FIFO by processor.....	1575
Figure 146. Reading from IP RX FIFO by DMA.....	1576
Figure 147. Filling IP TX FIFO by processor.....	1577
Figure 148. Filling into IP TX FIFO by DMA.....	1577
Figure 149. Hardware operation in response to AHB write access to Flash.....	1579
Figure 150. AHB write access (INCR/64bit/Bufferable).....	1581
Figure 151. AHB write access (WRAP8/64bit/Bufferable).....	1582
Figure 152. AHB write access (WRAP8/64bit/Non-Bufferable).....	1583
Figure 153. Hardware operation in response to AHB read access to Flash.....	1584
Figure 154. Command Instructions Execution on FlexSPI Interface.....	1588
Figure 155. SDR instruction in SDR sequence (Channel A).....	1590
Figure 156. SDR instruction in SDR sequence (Channel B).....	1590
Figure 157. SDR instruction in DDR sequence (Channel A).....	1590
Figure 158. SDR instruction in DDR sequence (Channel B).....	1590
Figure 159. DDR instruction in DDR sequence (Channel A).....	1591
Figure 160. DDR instruction in DDR sequence (Channel B).....	1591
Figure 161. Chip select output timing for SDR sequence (Channel A).....	1591
Figure 162. Chip select output timing for SDR sequence (Channel B).....	1591
Figure 163. Chip Select Output Timing for DDR Sequence (Channel A).....	1592
Figure 164. Chip Select Output Timing for DDR Sequence (Channel B).....	1592
Figure 165. Chip Select Valid Interval (Channel A).....	1592
Figure 166. Chip Select Valid Interval (Channel B).....	1593
Figure 167. Sampling Block in FlexSPI (Channel A).....	1594
Figure 168. Sampling Block in FlexSPI (Channel B).....	1595
Figure 169. Sampling Blocks for Data Learning Feature (Channel A).....	1596
Figure 170. Sampling Blocks for Data Learning Feature (Channel B).....	1596
Figure 171. Unsupported Read Behavior.....	1597
Figure 172. Supported Read Behavior.....	1597
Figure 173. Input Timing for sampling with dummy read strobe in SDR mode (Channel A).....	1598
Figure 174. Input Timing for sampling with dummy read strobe in SDR mode (Channel B).....	1598
Figure 175. Input timing for sampling with dummy read strobe in DDR mode (Channel A).....	1599
Figure 176. Input timing for sampling with dummy read strobe in DDR mode (Channel B).....	1599

Figure 177. Input timing 2 for Flash provided read strobe in SDR mode (Channel A).....	1599
Figure 178. Input timing 2 for Flash provided read strobe in SDR mode (Channel B).....	1600
Figure 179. Input timing 2 for Flash provided read strobe in DDR mode (Channel A).....	1600
Figure 180. Input timing 2 for Flash provided read strobe in DDR mode (Channel B).....	1601
Figure 181. Data learning flow with flash providing preamble bit.....	1603
Figure 182. Data learning flow with Flash not providing preamble bit.....	1603
Figure 183. Preprogramming data for data learning (Channel A).....	1604
Figure 184. Preprogramming data for data learning (Channel B).....	1604
Figure 185. XIP Enhanced Mode Operation.....	1605
Figure 186. OTFAD and FlexSPI connection topology.....	1726
Figure 187. CTR-AES128 encrypt and decrypt overview.....	1729
Figure 188. OTFAD 64-bit WRAP4 burst timing diagram.....	1733
Figure 189. OTFAD Operating Mode States and Transitions.....	1735
Figure 190. Basic OTFAD 32-byte decryption processing.....	1736
Figure 191. MU Block Diagram.....	1754
Figure 192. Messaging Model Using Transmit and Receive Registers.....	1760
Figure 193. Messaging Model Using a General Purpose Interrupt.....	1761
Figure 194. MU Registers.....	1765
Figure 195. SEMA42 block diagram.....	1790
Figure 196. GATEn state machine.....	1792
Figure 197. Block diagram.....	1797
Figure 198. Cache memory regions	1814
Figure 199. POLSEL configuration example	1815
Figure 200. LCDIF Block Diagram.....	1820
Figure 201. DBI output timing definitions for Type A and Type B.....	1823
Figure 202. Reset Sequence Diagram.....	1824
Figure 203. Double Buffered Data Flow.....	1825
Figure 204. GCNanoLiteV Core Block Diagram.....	1871
Figure 205. DSI Host Controller Core Block Diagram.....	1875
Figure 206. DPI-2 Host Bridge Core Block Diagram.....	1881
Figure 207. DSI Host Controller Clocking Block Diagram.....	1884
Figure 208. MIPI D-PHY Block Diagram.....	1886
Figure 209. D-PHY Lane Diagram.....	1886
Figure 210. Ideal Single-ended and Resulting Differential HS Signals.....	1887
Figure 211. Possible ΔV_{CMTX} and ΔV_{OD} Distortions of the Single-Ended HS Signals.....	1888
Figure 212. Test Circuit for VCMTX and VOD measurement.....	1889
Figure 213. Test Circuit for Output Impedance measurements.....	1890
Figure 214. Signaling and MIPI Contention Voltage Levels.....	1891
Figure 215. Pin-Leakage Measurement DC specification.....	1891
Figure 216. Serializer Timing Diagram.....	1892
Figure 217. High-Speed Transmission from the Master Side.....	1893
Figure 218. Low-Power Data Transmission.....	1894
Figure 219. Turn-around Actions Transmit-to-Receive and Back to Transmit.....	1894
Figure 220. DMIC block diagram.....	1948
Figure 221. DMIC channel block diagram.....	1949

Figure 222. DMIC interface clock domains.....	1949
Figure 223. Principle structure of the PDM to PCM conversion.....	1950
Figure 224. DMIC FIFO and DMA.....	1951
Figure 225. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers).....	1952
Figure 226. Complete HWVAD setup.....	1953
Figure 227. DMIC subsystem signal multiplexing.....	1955
Figure 228. Example connection for a single independent microphone.....	1955
Figure 229. Example connection to two independent microphones.....	1955
Figure 230. Example connection to two microphones sharing clock and data lines.....	1956
Figure 231. Pre-emphasis filter quantized response at 96 kHz.....	1963
Figure 232. System connection of uSDHC.....	1984
Figure 233. uSDHC block diagram.....	1985
Figure 234. uSDHC buffer scheme.....	1989
Figure 235. Data swap between system bus and uSDHC data buffer in the byte little endian mode.....	1990
Figure 236. Data swap between system bus and uSDHC data buffer in half word big endian mode.....	1990
Figure 237. Example for dividing large data transfers.....	1992
Figure 238. DMA AHB interface block.....	1993
Figure 239. Format of the ADMA1 descriptor table.....	1995
Figure 240. Concept and access method of the ADMA1 descriptor table.....	1996
Figure 241. Format of the ADMA2 descriptor table.....	1997
Figure 242. Concept and access method of the ADMA2 descriptor table.....	1998
Figure 243. Register bank diagram.....	1999
Figure 244. Command CRC shift register.....	2000
Figure 245. Two stages of the clock divider.....	2001
Figure 246. Card interrupt scheme, card interrupt detection, and handling procedure.....	2003
Figure 247. MultiMediaCard state diagram (normal boot mode).....	2005
Figure 248. MultiMediaCard state diagram (alternative boot mode).....	2005
Figure 249. Flow diagram for card detection.....	2007
Figure 250. Flow chart for resetting uSDHC and SD I/O card.....	2008
Figure 251. DLL in read path.....	2015
Figure 252. FlexIO block diagram.....	2172
Figure 253. Shifter Microarchitecture.....	2173
Figure 254. State Microarchitecture.....	2175
Figure 255. Logic Microarchitecture.....	2176
Figure 256. State Diagram.....	2198
Figure 257. Flexcomm module block diagram.....	2347
Figure 258. USART block diagram.....	2354
Figure 259. Hardware flow control using RTS and CTS.....	2357
Figure 260. SPI block diagram.....	2394
Figure 261. Data stall examples.....	2397
Figure 262. Pre-delay and post-delay.....	2398
Figure 263. Frame delay.....	2399
Figure 264. Transfer delay.....	2400
Figure 265. Basic SPI operating modes.....	2401
Figure 266. I2C block diagram.....	2433

Figure 267. I2S block diagram.....	2470
Figure 268. Classic I2S mode.....	2472
Figure 269. DSP mode with 50% WS.....	2472
Figure 270. DSP mode with 1 SCK pulsed WS.....	2472
Figure 271. DSP mode with 1 slot pulsed WS.....	2473
Figure 272. TDM in classic I2S mode.....	2473
Figure 273. TDM and DSP modes with 50% WS.....	2473
Figure 274. TDM and DSP modes with 1 SCK pulsed WS.....	2474
Figure 275. TDM and DSP modes with 1 slot pulsed WS.....	2474
Figure 276. I2S mode, mono.....	2474
Figure 277. DSP mode, mono.....	2475
Figure 278. TDM and DSP modes, mono, with WS pulsed for one SCK time.....	2475
Figure 279. SCK and WS polarities.....	2475
Figure 280. PTD scheduler flowchart.....	2514
Figure 281. USB host software interface.....	2523
Figure 282. USB Controller Block Diagram.....	2549
Figure 283. Endpoint command/status list.....	2550
Figure 284. Flowchart of control endpoint 0 - OUT direction.....	2554
Figure 285. Flowchart of control endpoint 0 - IN direction.....	2555
Figure 286. USB1 software interface.....	2558
Figure 287. Block diagram.....	2575
Figure 288. USB battery charging subsystem.....	2576
Figure 289. Full speed charger detection timing for BC1.2.....	2579
Figure 290. Full speed charger detection timing for BC1.1.....	2580
Figure 291. Relative pin positions in USB plugs and receptacles.....	2582
Figure 292. USB2.0 PHY block diagram.....	2604
Figure 293. USB 2.0 PHY Analog Transceiver Block Diagram.....	2606
Figure 294. USB2.0 PHY Transmitter Block Diagram.....	2609
Figure 295. ADC block diagram.....	2719
Figure 296. ADC command execution flow diagram.....	2721
Figure 297. ADC Resync Example.....	2725
Figure 298. Approximate transfer function of the temperature sensor.....	2726
Figure 299. CMP high level diagram.....	2769
Figure 300. Comparator module block diagram.....	2770
Figure 301. Filter block bypass logic.....	2770
Figure 302. Comparator operation in Continuous mode.....	2773
Figure 303. Sampled, Non-Filtered (# 3B): sampling interval internally derived.....	2774
Figure 304. Sampled, Filtered (# 4B): sampling point internally derived.....	2775
Figure 305. Windowed mode timing diagram.....	2776
Figure 306. Windowed mode.....	2776
Figure 307. CMP_C3[RDIVE] = 0.....	2779
Figure 308. CMP_C3[RDIVE] = 1.....	2779
Figure 309. Slow (32 kHz) clock timing.....	2780
Figure 310. 8-bit DAC block diagram.....	2781
Figure 311. Trigger mode	2783

Figure 312. Connecting the SWD pins to a standard SWD connector.....	2802
Figure 313. Serial Wire Debug (SWD) internal connections.....	2802
Figure 314. Debug Authentication protocol usage example.....	2813
Figure 315. Debug Credential certificate fields.....	2817
Figure 316. Debug Authentication Challenge (DAC) fields.....	2819
Figure 317. Debug Authentication Response (DAR) fields.....	2821
Figure 318. Debug Authentication protocol usage example.....	2823
Figure 319. JTAG (IEEE 1149.1) block diagram.....	2831
Figure 320. Shifting data through a register.....	2834
Figure 321. IEEE 1149.1-2001 TAP controller finite state machine.....	2835
Figure 322. Instruction register.....	2838

Tables

Table 1. Suggested Reading.....	36
Table 2. System memory map (M33/DSP/AHB overview).....	55
Table 3. Shared RAM Address Map.....	56
Table 4. APB0 (32 Peripheral slots)	57
Table 5. APB1 (32 Peripheral slots)	58
Table 6. AHB Peripherals	60
Table 7. Arm M33 NVIC assignments.....	64
Table 8. DMA0 and DMA1 request assignments.....	68
Table 9. DMA0 and DMA1 input trigger multiplexing assignments.....	70
Table 10. DMA0 and DMA1 output trigger assignments.....	71
Table 11. Fusion DSP Interrupt input multiplexers (selected by DSP_INT_SELn).....	73
Table 12. Fusion DSP assignments.....	74
Table 13. Reference links to related information.....	76
Table 14. Reference links to related information.....	218
Table 15. Reference links to related information.....	257
Table 16. Clock signal descriptions.....	258
Table 17. CLKCTL pin description.....	265
Table 18. POWER_UpdateOscSettlingTime API.....	472
Table 19. POWER_ApplyPD API.....	472
Table 20. POWER_EnablePD API.....	473
Table 21. POWER_DisablePD API.....	473
Table 22. POWER_ClearEventFlags API.....	473
Table 23. Event flags (statusMask).....	474
Table 24. POWER_GetEventFlags API.....	475
Table 25. POWER_EnableInterrupts API.....	475
Table 26. PMC interrupts (interruptMask).....	475
Table 27. POWER_DisableInterrupts API.....	476
Table 28. POWER_SetAnalogBuffer API.....	476
Table 29. POWER_SetPadVolRange API.....	476
Table 30. Structure of power_pad_vrange_t parameter.....	477
Table 31. Pad voltage range values (power_pad_vrange_val_t).....	477
Table 32. POWER_EnterRbb API.....	477
Table 33. POWER_EnterFbb API.....	478
Table 34. POWER_EnterNbb API.....	478
Table 35. POWER_SetLdoVoltageForFreq API.....	478
Table 36. POWER_SetLvdFallingTripVoltage.....	479
Table 37. Trip voltage values (power_lvd_falling_trip_vol_val_t).....	479
Table 38. POWER_EnterSleep API.....	480
Table 39. POWER_EnterDeepSleep API.....	480
Table 40. POWER_EnterDeepPowerDown API.....	480
Table 41. POWER_EnterFullDeepPowerDown API.....	481
Table 42. POWER_EnterPowerMode API.....	481

Table 43. Power modes (power_mode_cfg_t).....	481
Table 44. EnableDeepSleepIRQ API.....	481
Table 45. DisableDeepSleepIRQ API.....	482
Table 46. POWER_GetLibVersion API.....	482
Table 47. POWER_SetDeepSleepClock.....	483
Table 48. Reference links to related information.....	484
Table 49. Peripheral configuration in reduced power modes.....	489
Table 50. Wake-up sources for reduced power modes.....	490
Table 51. Cortex-M33 options.....	513
Table 52. Reference links to related information.....	525
Table 53. Reference links to related information.....	563
Table 54. INPUTMUX pin description.....	564
Table 55. Reference links to related information.....	662
Table 56. Reference links to related information.....	700
Table 57. Pin interrupt registers for edge- and level-sensitive pins.....	707
Table 58. Reference links to related information.....	722
Table 59. Channel Descriptor offsets for DMA channels on the chip.....	722
Table 60. DMA Trigger Input.....	725
Table 61. DMA Trigger Output.....	726
Table 62. Memory locations in a Channel Descriptor.....	729
Table 63. Reload descriptors.....	729
Table 64. Channel Descriptor for a single buffer.....	730
Table 65. Example descriptors for ping-pong operation: peripheral to buffer.....	731
Table 66. DMA with the I2C Monitor function.....	733
Table 67. Trigger setting summary.....	759
Table 68. Primary Boot Source based on PRIMARY_BOOT_SRC bits[3:0] in BOOT_CFG [0].....	768
Table 69. Boot mode and ISP Downloader modes based on ISP pins.....	770
Table 70. ISP peripheral pin mux assignments.....	771
Table 71. imageType Field.....	775
Table 72. Image Header Format (shared for all imageTypes).....	776
Table 73. Plain Image Layout.....	777
Table 74. XIP Image layout (FlexSPI).....	777
Table 75. Image offset on different boot media.....	779
Table 76. FLASH CONFIG TABLE.....	779
Table 77. Boot image 1 offset(BOOT_CFG3: OTP fuse word 0x63).....	789
Table 78. Boot Image 1 size (BOOT_CFG2).....	789
Table 79. Pin Assignments for blhost tool communication	793
Table 80. FLEXSPI pin assignments for NOR flash connections.....	793
Table 81. serial_nor_config_option_t definition.....	795
Table 82. Option0 definition.....	795
Table 83. Option1 definition.....	796
Table 84. Typical NOR flash config parameters(flash connected to FlexSPI PORT A).....	799
Table 85. BOOT_CFG2 boot configuration.....	801
Table 86. BOOT_CFG3 boot configuration.....	803
Table 87. Master boot fail log.....	806

Table 88. Master boot, state, sub state and status value.....	807
Table 89. Recovery boot OTP field.....	808
Table 90. Boot ROM Status Error Codes, sorted by Value.....	811
Table 91. Memory ID for external memory devices.....	814
Table 92. Boot mode and ISP Downloader modes based on ISP pins	815
Table 93. UART pins used for boot ROM.....	815
Table 94. Lookup Table index pre-assignment for FlexSPI NOR.....	821
Table 95. eMMC Configuration Block Definition.....	826
Table 96. 1-bit SPI NOR Configuration Option Block.....	831
Table 97. Ping Packet Format.....	837
Table 98. Ping Response packet format.....	838
Table 99. Framing Packet Format.....	838
Table 100. Special Framing Packet Format.....	839
Table 101. packetType Field.....	839
Table 102. Command Packet Format.....	840
Table 103. Command Header Format.....	841
Table 104. Command Tags.....	841
Table 105. Response Tags.....	842
Table 106. GenericResponse Parameters.....	843
Table 107. GetPropertyResponse Parameters.....	843
Table 108. ReadMemoryResponse Parameters.....	843
Table 109. FlashReadOnceResponse Parameters.....	844
Table 110. KeyProvisionResponse Parameters.....	844
Table 111. GetProperty Command Packet Format (Example).....	846
Table 112. GetProperty Response Packet Format (Example).....	847
Table 113. Parameters for SetProperty Command.....	847
Table 114. SetProperty Command Packet Format (Example).....	848
Table 115. SetProperty Response Status Codes.....	848
Table 116. Parameter for FlashEraseAll Command.....	849
Table 117. FlashEraseAll Command Packet Format (Example).....	850
Table 118. Parameters for FlashEraseRegion Command.....	850
Table 119. FlashEraseRegion Response Status Codes.....	851
Table 120. Parameters for read memory command.....	851
Table 121. ReadMemory Command Packet Format (Example).....	852
Table 122. Parameters for WriteMemory Command.....	853
Table 123. WriteMemory Command Packet Format (Example).....	854
Table 124. Parameters for FillMemory Command.....	854
Table 125. FillMemory Command Packet Format (Example).....	855
Table 126. Parameters for Execute Command.....	856
Table 127. Parameters for Call Command.....	856
Table 128. Reset Command Packet Format (Example).....	857
Table 129. Parameters for FlashProgramOnce Command.....	857
Table 130. FlashProgramOnce Command Packet Format (Example).....	858
Table 131. Parameters for FlashReadOnce Command.....	859
Table 132. FlashReadOnce Command Packet Format (Example).....	859

Table 133. FlashReadOnce Response Format (Example).....	859
Table 134. Parameters for Configure Memory Command.....	860
Table 135. Parameters for Receive SB File Command.....	861
Table 136. Parameters for KeyProvision Command.....	861
Table 137. KeyProvision Operation Details.....	861
Table 138. Key Type Details.....	862
Table 139. KeyProvision command packet format(example).....	863
Table 140. KeyProvision Response packet format (Example).....	863
Table 141. Location of runboot ROM API in ROM API	875
Table 142. Combination of the user boot options.....	877
Table 143. OTP API calls.....	885
Table 144. FLEXSPI FLASH Driver API List.....	890
Table 145. flexspi_nor_flash_init API.....	891
Table 146. flexspi_nor_flash_page_program API.....	891
Table 147. flexspi_nor_flash_erase_all API.....	892
Table 148. flexspi_nor_flash_erase API.....	892
Table 149. . flexspi_nor_flash_erase_sector API.....	893
Table 150. flexspi_nor_flash_erase_block API.....	893
Table 151. flexspi_nor_flash_read API.....	894
Table 152. flexspi_clock_config_API.....	895
Table 153. flexspi_nor_set_clock_source.....	896
Table 154. flexspi_nor_get_config API.....	896
Table 155. . serial_nor_config_option_t definition.....	897
Table 156. . Option0 definition.....	898
Table 157. Option1 Definition.....	899
Table 158. Return and Error codes for Flash APIs.....	900
Table 159. IAP bootloader API status code.....	904
Table 160. ISP commands	908
Table 161. OTP Key store summary.....	908
Table 162. OTP Key store summary.....	909
Table 163. OTFAD_KEK_SEED organization in OTP fuses.....	910
Table 164. PUF Key code storage area structure.....	910
Table 165. PUF Key code storage area structure.....	910
Table 166. BOOT_CFG[0] bit fields.....	912
Table 167. BOOT_CFG[5] bit fields.....	913
Table 168. REVOKE_ROOTKEY field description.....	914
Table 169. BOOT_CFG[6] bit fields.....	915
Table 170. RKTH layout in OTP.....	916
Table 171. NT_FW_VER layout in OTP.....	917
Table 172. TZ_FW_VERSION layout in OTP.....	917
Table 173. Image Type Word.....	918
Table 174. Signed image - Image Type (Word at offset 0x24).....	919
Table 175. Standard Cortex_M33 NVIC vector table.....	920
Table 176. Certificate block header.....	925
Table 177. OTFAD Key Blob layout in the external NOR flash.....	928

Table 178. Key Blob “n” content after ROM key blob unwrap.....	929
Table 179. TrustZone image type.....	930
Table 180. TrustZone data structure.....	937
Table 181. TrustZone configuration data.....	937
Table 182. TrustZone image type restriction control.....	938
Table 183. Secure ROM API.....	944
Table 184. HASH_IRQHandler.....	945
Table 185. Example memory layout after TrustZone configuration.....	956
Table 186. Basic SAU Configuration.....	957
Table 187. Canonical SAU Configuration.....	959
Table 188. Combined SAU Configuration.....	960
Table 189. Reference links to related information.....	962
Table 190. ROM_MEM_RULE0.....	967
Table 191. ROM_MEM_RULE1.....	967
Table 192. ROM_MEM_RULE2.....	968
Table 193. ROM_MEM_RULE3.....	968
Table 194. FLEXSPI0_REGION0_RULE0.....	971
Table 195. FLEXSPI0_REGION0_RULE1.....	972
Table 196. FLEXSPI0_REGION0_RULE2.....	972
Table 197. FLEXSPI0_REGION0_RULE3.....	972
Table 198. FLEXSPI0_REGION1_RULE0.....	976
Table 199. FLEXSPI0_REGION2_RULE0.....	976
Table 200. FLEXSPI0_REGION3_RULE0.....	976
Table 201. FLEXSPI0_REGION4_RULE0.....	977
Table 202. RAM00_RULE0.....	979
Table 203. RAM00_RULE1.....	979
Table 204. RAM00_RULE2.....	979
Table 205. RAM00_RULE3.....	980
Table 206. RAM01_RULE0.....	983
Table 207. RAM01_RULE1.....	983
Table 208. RAM01_RULE2.....	984
Table 209. RAM01_RULE3.....	984
Table 210. RAM02_RULE0.....	987
Table 211. RAM02_RULE1.....	987
Table 212. RAM02_RULE2.....	988
Table 213. RAM02_RULE3.....	988
Table 214. RAM03_RULE0.....	991
Table 215. RAM03_RULE1.....	992
Table 216. RAM03_RULE2.....	992
Table 217. RAM03_RULE3.....	992
Table 218. RAM04_RULE0.....	996
Table 219. RAM04_RULE1.....	996
Table 220. RAM04_RULE2.....	996
Table 221. RAM04_RULE3.....	996
Table 222. RAM05_RULE0.....	1000

Table 223. RAM05_RULE1.....	1000
Table 224. RAM05_RULE2.....	1000
Table 225. RAM05_RULE3.....	1001
Table 226. RAM06_RULE0.....	1004
Table 227. RAM06_RULE1.....	1004
Table 228. RAM06_RULE2.....	1004
Table 229. RAM06_RULE3.....	1005
Table 230. RAM07_RULE0.....	1008
Table 231. RAM07_RULE1.....	1008
Table 232. RAM07_RULE2.....	1008
Table 233. RAM07_RULE3.....	1009
Table 234. RAM08_RULE0.....	1012
Table 235. RAM08_RULE1.....	1012
Table 236. RAM08_RULE2.....	1012
Table 237. RAM08_RULE3.....	1013
Table 238. RAM09_RULE0.....	1016
Table 239. RAM09_RULE1.....	1016
Table 240. RAM09_RULE2.....	1016
Table 241. RAM09_RULE3.....	1017
Table 242. RAM10_RULE0.....	1020
Table 243. RAM10_RULE1.....	1020
Table 244. RAM10_RULE2.....	1020
Table 245. RAM10_RULE3.....	1021
Table 246. RAM11_RULE0.....	1024
Table 247. RAM11_RULE1.....	1024
Table 248. RAM11_RULE2.....	1024
Table 249. RAM11_RULE3.....	1025
Table 250. RAM12_RULE0.....	1028
Table 251. RAM12_RULE1.....	1028
Table 252. RAM12_RULE2.....	1028
Table 253. RAM12_RULE3.....	1029
Table 254. RAM13_RULE0.....	1032
Table 255. RAM13_RULE1.....	1032
Table 256. RAM13_RULE2.....	1032
Table 257. RAM13_RULE3.....	1033
Table 258. RAM14_RULE0.....	1036
Table 259. RAM14_RULE1.....	1036
Table 260. RAM14_RULE2.....	1036
Table 261. RAM14_RULE3.....	1037
Table 262. RAM15_RULE0.....	1040
Table 263. RAM15_RULE1.....	1040
Table 264. RAM15_RULE2.....	1040
Table 265. RAM15_RULE3.....	1041
Table 266. RAM16_RULE0.....	1044
Table 267. RAM16_RULE1.....	1044

Table 268. RAM16_RULE2.....	1044
Table 269. RAM16_RULE3.....	1045
Table 270. RAM17_RULE0.....	1048
Table 271. RAM17_RULE1.....	1048
Table 272. RAM17_RULE2.....	1048
Table 273. RAM17_RULE3.....	1049
Table 274. RAM18_RULE0.....	1052
Table 275. RAM18_RULE1.....	1052
Table 276. RAM18_RULE2.....	1052
Table 277. RAM18_RULE3.....	1053
Table 278. RAM19_RULE0.....	1056
Table 279. RAM19_RULE1.....	1056
Table 280. RAM19_RULE2.....	1056
Table 281. RAM19_RULE3.....	1057
Table 282. RAM20_RULE0.....	1060
Table 283. RAM20_RULE1.....	1060
Table 284. RAM20_RULE2.....	1060
Table 285. RAM20_RULE3.....	1061
Table 286. RAM21_RULE0.....	1064
Table 287. RAM21_RULE1.....	1064
Table 288. RAM21_RULE2.....	1064
Table 289. RAM21_RULE3.....	1065
Table 290. RAM22_RULE0.....	1068
Table 291. RAM22_RULE1.....	1068
Table 292. RAM22_RULE2.....	1068
Table 293. RAM22_RULE3.....	1069
Table 294. RAM23_RULE0.....	1072
Table 295. RAM23_RULE1.....	1072
Table 296. RAM23_RULE2.....	1072
Table 297. RAM23_RULE3.....	1073
Table 298. RAM24_RULE0.....	1076
Table 299. RAM24_RULE1.....	1076
Table 300. RAM24_RULE2.....	1076
Table 301. RAM24_RULE3.....	1077
Table 302. RAM25_RULE0.....	1080
Table 303. RAM25_RULE1.....	1080
Table 304. RAM25_RULE2.....	1080
Table 305. RAM25_RULE3.....	1081
Table 306. RAM26_RULE0.....	1084
Table 307. RAM26_RULE1.....	1084
Table 308. RAM26_RULE2.....	1084
Table 309. RAM26_RULE3.....	1085
Table 310. RAM27_RULE0.....	1088
Table 311. RAM27_RULE1.....	1088
Table 312. RAM27_RULE2.....	1088

Table 313. RAM27_RULE3.....	1089
Table 314. RAM28_RULE0.....	1092
Table 315. RAM28_RULE1.....	1092
Table 316. RAM28_RULE2.....	1092
Table 317. RAM28_RULE3.....	1093
Table 318. RAM29_RULE0.....	1096
Table 319. RAM29_RULE1.....	1096
Table 320. RAM29_RULE2.....	1096
Table 321. RAM29_RULE3.....	1097
Table 322. RAM30_RULE0.....	1100
Table 323. RAM30_RULE1.....	1100
Table 324. RAM30_RULE2.....	1100
Table 325. RAM30_RULE3.....	1101
Table 326. RAM31_RULE0.....	1104
Table 327. RAM31_RULE1.....	1104
Table 328. RAM31_RULE2.....	1104
Table 329. RAM31_RULE3.....	1105
Table 330. SDMA_RAM_RULE0.....	1108
Table 331. SDMA_RAM_RULE1.....	1108
Table 332. SDMA_RAM_RULE2.....	1108
Table 333. SDMA_RAM_RULE3.....	1109
Table 334. FLEXSPI1_REGION0_RULE0.....	1112
Table 335. FLEXSPI1_REGION0_RULE1.....	1112
Table 336. FLEXSPI1_REGION0_RULE2.....	1113
Table 337. FLEXSPI1_REGION0_RULE3.....	1113
Table 338. FLEXSPI1_REGION1_RULE0.....	1116
Table 339. FLEXSPI1_REGION2_RULE0.....	1117
Table 340. FLEXSPI1_REGION3_RULE0.....	1117
Table 341. FLEXSPI1_REGION4_RULE0.....	1117
Table 342. APB_BRIDGE_PER0_RULE0.....	1119
Table 343. APB_BRIDGE_PER0_RULE1.....	1122
Table 344. APB_BRIDGE_PER0_RULE3.....	1123
Table 345. APB_BRIDGE_PER1_RULE0.....	1125
Table 346. APB_BRIDGE_PER1_RULE1.....	1127
Table 347. APB_BRIDGE_PER1_RULE2.....	1131
Table 348. APB_BRIDGE_PER1_RULE3.....	1133
Table 349. AHB_PERIPH0_SLAVE_RULE0.....	1135
Table 350. AIPS_BRIDGE0_PER_RULE0.....	1138
Table 351. AHB_PERIPH1_SLAVE_RULE0.....	1140
Table 352. AHB_PERIPH1_SLAVE_RULE1.....	1144
Table 353. AIPS_BRIDGE1_PER_RULE0.....	1145
Table 354. AIPS_BRIDGE1_PER_RULE1.....	1148
Table 355. AHB_PERIPH2_SLAVE_RULE0.....	1150
Table 356. AHB_SECURE_CTRL_PERIPH_RULE0.....	1152
Table 357. AHB_PERIPH3_SLAVE_RULE0.....	1154

Table 358. AHB_PERIPH3_SLAVE_RULE1.....	1157
Table 359. Reference links to related information.....	1178
Table 360. Performance using CM33 alone versus with PowerQuad	1182
Table 361. Performance using CM33 alone versus with PowerQuad	1183
Table 362. Performance using PowerQuad	1184
Table 363. Performance using CM33 alone versus with PowerQuad	1188
Table 364. Matrix operation dimensions.....	1190
Table 365. Performance using CM33 alone versus with PowerQuad	1190
Table 366. Performance using CM33 alone versus with PowerQuad	1192
Table 367. Performance using CM33 alone versus with PowerQuad	1193
Table 368. MCR and MRC opcodes	1194
Table 369. Parameter descriptions.....	1194
Table 370. Commands.....	1195
Table 371. Read back commands.....	1197
Table 372. PQ_MatrixAddition parameters.....	1204
Table 373. PQ_TransformCFFT parameters.....	1206
Table 374. PQ_FIR parameters.....	1207
Table 375. PQ_ArctanFixed parameters.....	1209
Table 376. Decode Opcodes.....	1220
Table 377. Reference links to related information.....	1234
Table 378. CASPER performance improvements.....	1235
Table 379. CASPER modes.....	1237
Table 380. Reference links to related information.....	1254
Table 381. Number of clock cycles per operation.....	1258
Table 382. Coding of KEYSIZE.....	1258
Table 383. KC header field description.....	1260
Table 384. Key target interfaces per key index.....	1262
Table 385. Function parameters.....	1263
Table 386. Data access functions.....	1263
Table 387. Reference links to related information.....	1291
Table 388. Reference links to related information.....	1371
Table 389. RTC pins.....	1374
Table 390. Reference links to related information.....	1389
Table 391. Timer/Counter pin description.....	1391
Table 392. Suggested CTIMER timer pin settings.....	1392
Table 393. Reference links to related information.....	1412
Table 394. OSTIMER registers and their corresponding resets	1414
Table 395. Reference links to related information.....	1429
Table 396. SCT0 signals (inputs).....	1429
Table 397. SCT0 signals (outputs).....	1430
Table 398. Suggested SCT input pin settings.....	1430
Table 399. Suggested SCT output pin settings.....	1430
Table 400. Event conditions.....	1437
Table 401. SCT signals.....	1438
Table 402. SCT configuration example: settings.....	1440

Table 403. Reference links to related information.....	1484
Table 404. Reference links to related information.....	1493
Table 405. Possible actions in Repeat Interrupt mode.....	1494
Table 406. Possible actions in One-Shot Interrupt mode.....	1495
Table 407. Reference links to related information.....	1505
Table 408. Watchdog operating modes selection.....	1508
Table 409. Reference links to related information.....	1519
Table 410. Reference links to related information.....	1524
Table 411. Common CRC standard configurations.....	1526
Table 412. Reference links to related information.....	1531
Table 413. Reference links to related information.....	1548
Table 414. FlexSPI0 Master ID Assignments.....	1549
Table 415. FlexSPI1 Master ID Assignments.....	1549
Table 416. Glossary.....	1551
Table 417. External Signal List.....	1553
Table 418. FlexSPI address translation.....	1557
Table 419. Flash address, row address and column address.....	1559
Table 420. Instruction set.....	1562
Table 421. Clock Usage.....	1571
Table 422. AHB read command flash start address and data size.....	1585
Table 423. WRITE ENABLE command.....	1606
Table 424. Write Registers command.....	1606
Table 425. PAGE PROGRAM command (Cypress serial NOR flash).....	1606
Table 426. PAGE PROGRAM (QPI mode) command (Cypress serial NOR flash).....	1607
Table 427. READ STATUS 1 command.....	1607
Table 428. READ command.....	1607
Table 429. FAST_READ command.....	1608
Table 430. Dual IO FAST_READ command (Continuous Read mode).....	1608
Table 431. Dual IO FAST_READ command (Non-continuous Read mode).....	1609
Table 432. Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode).....	1609
Table 433. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode).....	1610
Table 434. Quad IO FAST_READ command (QPI mode, Continuous read mode).....	1610
Table 435. DDR Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode).....	1611
Table 436. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode).....	1612
Table 437. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode).....	1612
Table 438. Read status command.....	1613
Table 439. Read (memory) command.....	1614
Table 440. Word program command.....	1615
Table 441. Write-to-Buffer and Program-Buffer-to-Flash command.....	1616
Table 442. Write (memory) command.....	1618
Table 443. Page Read command.....	1619
Table 444. Get Feature command.....	1619
Table 445. Read from Cache x4 command.....	1619
Table 446. Program Load command.....	1620
Table 447. Program Execute command.....	1621

Table 448. Error category, triggered sources and flags in FlexSPI.....	1621
Table 449. Reference links to related information.....	1725
Table 450. Cryptographic terms.....	1727
Table 451. OTFAD and decrypted data buffer operation.....	1732
Table 452. Static configuration signal settings.....	1734
Table 453. OTFAD normal mode configurations.....	1734
Table 454. OTFAD mode state transitions.....	1735
Table 455. Reference links to related information.....	1751
Table 456. Reference links to related information.....	1753
Table 457. Major Features of the MU.....	1755
Table 458. MU programmable resets.....	1758
Table 459. Interrupt Messaging Protocol (Generalized).....	1759
Table 460. Interrupt Messaging Protocol (Generalized).....	1761
Table 461. How the Processor A Performs an Exclusive Access to Shared Memory.....	1762
Table 462. How the Processor B Scans for Transaction Information.....	1762
Table 463. How the Processor B Accepts Exclusive Access by Processor A.....	1762
Table 464. How the Processor B Rejects Exclusive Access by Processor A.....	1763
Table 465. Packet Data Transfer Sequence.....	1763
Table 466. Reference links to related information.....	1789
Table 467. GATEn state transitions.....	1792
Table 468. Reference links to related information.....	1797
Table 469. Modes of operation.....	1798
Table 470. Tag cache address use.....	1800
Table 471. Data cache address use.....	1800
Table 472. Cache set commands.....	1801
Table 473. Cache line commands.....	1802
Table 474. Line command results.....	1803
Table 475. Reference links to related information.....	1819
Table 476. LCDIF Signals.....	1819
Table 477. Host and memory interface features.....	1821
Table 478. Display features.....	1821
Table 479. MIPI Protocol features.....	1822
Table 480. Clocks.....	1823
Table 481. Window alignment.....	1827
Table 482. Reference links to related information.....	1869
Table 483. Graphics Hardware Features.....	1870
Table 484. Graphics Formats Available.....	1870
Table 485. Blit and Draw Controls in VGLite API.....	1872
Table 486. Reference links to related information.....	1873
Table 487. DSI Host Controller Core Transmit Packet Interface.....	1876
Table 488. DSI Host Controller Core Receive Packet interface.....	1878
Table 489. DSI Host Controller Status Interface Port.....	1879
Table 490. DSI Host Controller Core Control And Status Packet Interface.....	1880
Table 491. DSI Host Controller Core Receive Packet interface.....	1881
Table 492. DSI DPHY PPI Signals.....	1892

Table 493. Reference links to related information.....	1945
Table 494. DMIC subsystem PDM signals description.....	1946
Table 495. Suggested PDM signal settings for the audio input.....	1946
Table 496. DMIC input and output clock rates.....	1950
Table 497. Reference links to related information.....	1983
Table 498. uSDHC external signals.....	1987
Table 499. uSDHC clocks.....	1988
Table 500. Commands for MMC/SD/SDIO cards.....	2024
Table 501. EXT_CSD access modes.....	2029
Table 502. Transfer type register setting for various transfer types.....	2036
Table 503. Relationship between parameters and the name of the response type.....	2037
Table 504. Response bit definition for each response type.....	2042
Table 505. uSDHC status for command timeout error/command complete bit combinations.....	2060
Table 506. uSDHC status for data timeout error/transfer complete bit combinations.....	2060
Table 507. uSDHC status for command CRC error/command timeout error bit combinations.....	2060
Table 508. Relationship between command CRC error and command timeout error for auto CMD12.....	2073
Table 509. Transfer type register setting for various transfer types.....	2080
Table 510. ADMA error state coding.....	2086
Table 511. Transfer type register setting for various transfer types.....	2108
Table 512. Relationship between parameters and the name of the response type.....	2108
Table 513. Response bit definition for each response type.....	2113
Table 514. uSDHC status for command timeout error/command complete bit combinations.....	2131
Table 515. uSDHC status for data timeout error/transfer complete bit combinations.....	2131
Table 516. uSDHC status for command CRC error/command timeout error bit combinations.....	2132
Table 517. Relationship between command CRC error and command timeout error for auto CMD12.....	2144
Table 518. Transfer type register setting for various transfer types.....	2151
Table 519. ADMA error state coding.....	2157
Table 520. Reference links to related information.....	2171
Table 521. Next State computation for SHIFTSTATE[STATE]=i.....	2175
Table 522. Logic Look-up table for Shifter 'i'.....	2176
Table 523. FlexIO Interrupts and DMA Requests.....	2182
Table 524. UART Transmit Configuration.....	2182
Table 525. UART Transmit Configuration for Slow Baud Rate.....	2183
Table 526. UART Receiver Configuration.....	2184
Table 527. UART Receiver with RTS Configuration.....	2185
Table 528. SPI Master (CPHA=0) Configuration.....	2186
Table 529. SPI Master (CPHA=1) Configuration.....	2187
Table 530. SPI Slave (CPHA=0) Configuration.....	2188
Table 531. SPI Slave (CPHA=1) Configuration.....	2189
Table 532. I2C Master Configuration.....	2190
Table 533. I2S Master Configuration.....	2192
Table 534. I2S Slave Configuration.....	2193
Table 535. SENT Receiver Configuration.....	2195
Table 536. Camera Interface Configuration for 8-bit CMOS sensor.....	2195
Table 537. Motorola 68K/Intel 8080 Write Configuration.....	2196

Table 538. Motorola 68K/Intel 8080 Read Configuration.....	2196
Table 539. State Machine Configuration.....	2198
Table 540. Keypad Interface Configuration.....	2200
Table 541.	2201
Table 542. Reference links to related information.....	2250
Table 543. Configuration parameters used to initialize the I3C module.....	2259
Table 544. Register Types.....	2263
Table 545. Reference links to related information.....	2336
Table 546. Flexcomm module base addresses and functions.....	2336
Table 547. USART signals	2339
Table 548. Suggested USART pin settings.....	2340
Table 549. SPI Signals.....	2342
Table 550. Suggested SPI pin settings.....	2343
Table 551. SPI Signals.....	2344
Table 552. Suggested I2C pins settings.....	2344
Table 553. I2S Pin Description.....	2346
Table 554. Suggested I2S pins settings.....	2346
Table 555. Flexcomm pins.....	2348
Table 556. USART signals	2359
Table 557. SPI mode summary.....	2401
Table 558. SPI signals.....	2402
Table 559. Settings for 400 kHz clock rate.....	2438
Table 560. Automatic operation cases.....	2440
Table 561. List of some terminology used in I2S.....	2470
Table 562. I2S Signals.....	2478
Table 563. Reference links to related information.....	2511
Table 564. USB Host pin description.....	2512
Table 565. PTD bit definition.....	2514
Table 566. PTD on asynchronous list (regular and split transaction).....	2519
Table 567. PTD on periodic list (regular transaction).....	2519
Table 568. PTD on periodic list (split transaction).....	2521
Table 569. Polling rate for periodic transactions.....	2522
Table 570. Reference links to related information.....	2547
Table 571. USB Device Controller Signals.....	2549
Table 572. Endpoint command/status bit definitions.....	2550
Table 573. Fixed endpoint configuration.....	2552
Table 574. Reference links to related information.....	2575
Table 575. USB battery charger subsystem components.....	2576
Table 576. Module modes and their conditions.....	2577
Table 577. Entering and exiting module modes.....	2578
Table 578. Timing parameters for the charger detection sequence for BC1.2.....	2579
Table 579. Timing parameters for the charger detection sequence for BC1.1.....	2581
Table 580. Overview of the charger detection sequence.....	2581
Table 581. Sampling D– in the charging port detection phase.....	2583
Table 582. Sampling D+ in the charger type detection phase (BC1.2).....	2585

Table 583. Sampling D– in the charger type detection phase (BC1.1).....	2585
Table 584. Software reset and register fields affected.....	2587
Table 585. Events triggering an interrupt by sequence phase.....	2588
Table 586. Signal descriptions.....	2589
Table 587. Glossary of terms.....	2590
Table 588. Reference links to related information.....	2603
Table 589. Reference links to related information.....	2716
Table 590. Chip-specific ADC signal descriptions.....	2716
Table 591. ADC configuration register.....	2717
Table 592. ADC channel mapping.....	2717
Table 593. ADC channels for Analog Signals.....	2717
Table 594. Hardware trigger sources.....	2717
Table 595. Power option settings.....	2722
Table 596. Chip modes supported by the ADC module.....	2722
Table 597. Compare modes.....	2727
Table 598. Compare operations.....	2727
Table 599. ADC signal descriptions.....	2728
Table 600. Data result register format description.....	2765
Table 601. Reference links to related information.....	2767
Table 602. CMP signal descriptions.....	2771
Table 603. Comparator sample/filter controls.....	2772
Table 604. Comparator sample/filter maximum latencies.....	2778
Table 605. Possible Combinations of CMP_C3[ACSAT], CMP_C3[ACPH1TC] and CMP_C3[ACPH2TC].....	2779
Table 606. CMP interrupt generations.....	2780
Table 607. Glossary.....	2800
Table 608. SWD blocks.....	2802
Table 609. Pin setting for JTAG boundary scan entry.....	2805
Table 610. Request register byte description.....	2808
Table 611. DM-AP commands.....	2809
Table 612. Response register byte description.....	2810
Table 613. Register overview: DM-AP legacy return codes.....	2811
Table 614. Register overview: DM-AP new protocol return codes.....	2811
Table 615. ACK_TOKEN register byte description.....	2811
Table 616. Access restriction levels.....	2814
Table 617. CC List Table.....	2815
Table 618. Layout of DCFG_CC_SOCU (OTP word 95) and DCFG_CC_SOCU_NS (OTP word 100).....	2815
Table 619. Debug Credential certificate fields.....	2817
Table 620. Debug Authentication Challenge (DAC) fields.....	2820
Table 621. Debug Authentication Response (DAR) fields.....	2821
Table 622. Serial wire debug signals.....	2825
Table 623. JTAG boundary scan signals.....	2826
Table 624. Trace signals.....	2826
Table 625. JTAG signal properties in Boundary Scan mode.....	2832
Table 626. JTAG instructions.....	2836
Table 627. Device identification register (JTAG ID).....	2838

Table 628. Device identification register field descriptions..... 2839

Table 629. Reference manual changes..... 2840

Preface

About This Manual

About This Document

The i.MX RT500 processors are NXP's latest additions to a growing family of real-time processing products offering high-performance processing optimized for lowest power consumption and best real-time response. The i.MX RT500 processors feature NXP's advanced implementation of the Arm Cortex-M33 core as well as the additional DSP Fusion core.

Audience

This manual is intended for the board-level product designers and product software developers. This manual assumes that the reader has a background in computer engineering and/or software engineering and understands the concepts of the digital system design, microprocessor architecture, Input/Output (I/O) devices, industry standard communication, and device interface protocols.

Organization

This document is organized into two main sets of chapters.

1. First set covers the chip at a system level and provides an architectural overview. It also covers the system memory map, system-level interrupt events, clock, reset and system controller.
2. Chapters in the second set provides a technical description of individual modules along with chip-specific contents at the beginning of each chapter.

Suggested Reading

This section lists the additional resources that provide background for the information in this manual, as well as general information about the architecture.

Table 1. Suggested Reading

Type	Description	Resource
Fact Sheet	The Fact Sheet is an overview of the product key features and its uses.	Fact Sheet
Reference Manual	The Reference Manual contains a comprehensive description of the structure and function (operation) of a device.	Reference Manual
Data Sheet	The Data Sheet includes electrical characteristics and signal connections.	Datasheet
Chip Errata	The Chip mask set Errata provides additional or corrective information for a particular device mask set.	Chip Errata
Application Notes	Provides additional information about particular device feature or function.	AN
Web Sites	Product summary page on nxp.com with documentation, software, and resources for the device.	RT500

Table continues on the next page...

Table 1. Suggested Reading (continued)

Type	Description	Resource
	Product documentation page on nxp.com with a list of all documentation related to the device.	RT500
Community Forum	i.MX RT Community support forum for questions, support, and information about the device.	NXP Community

General Information

The following documentation provides useful background information about the ARM Cortex processor.

For information about the ARM Cortex processor see:

- <http://infocenter.arm.com>

Related Documentation

For a current list of documentation, refer to <http://www.nxp.com>.

Conventions

This document uses the following notational conventions:

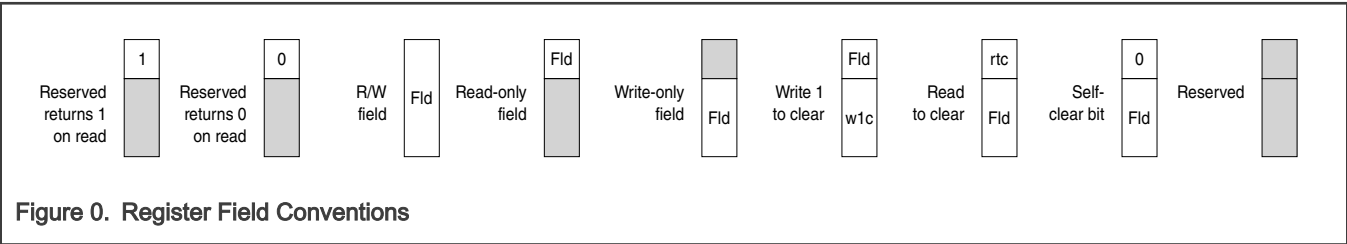
cleared / set	When a bit has a value of zero, it is said to be cleared; when it has a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, bcctr <i>x</i> . The book titles in the text are set in italics.
15	An integer in decimal.
0x	the prefix to denote a hexadecimal number.
0b	The prefix to denote a binary number. Binary values of 0 and 1 are written without a prefix.
n'H4000CA00	The n-bit hexadecimal number.
BLK_REG_NAME	The register names are all uppercase. The block mnemonic is prepended with an underscore delimiter (_).
BLK_REG[FIELD]	The fields within registers appear in brackets. For example, ESR[RLS] refers to the Receive Last Slot field of the ESAI Status Register.
BLK_REG[<i>n</i>]	The bit number <i>n</i> within the BLK.REG register.
BLK_REG[<i>l</i>:<i>r</i>]	The register bit ranges. The ranges are indicated by the left-most bit number <i>l</i> and the right-most bit number <i>r</i> , separated by a colon (:). For example, ESR[15:0] refers to the lower half word in the ESAI Status Register.
x, U	In some contexts, such as signal encodings, an unitalicized x indicates a "don't care" or "uninitialized". The binary value can be 1 or 0.
<i>x</i>	An italicized <i>x</i> indicates an alphanumeric variable.
<i>n, m</i>	Italicized <i>n</i> or <i>m</i> represent integer variables.
!	Binary logic operator NOT.
&&	Binary logic operator AND.

	Binary logic operator OR.
^ or <O+>	Binary logic operator XOR. For example, A <O+> B.
	Bit-wise OR. For example, 0b0001 0b1000 yields the value of 0b1001.
&	Bit-wise AND. For example, 0b0001 and 0b1000 yields the value of 0b0000.
{A,B}	Concatenation, where the <i>n</i> -bit value A is prepended to the <i>m</i> -bit value B to form an (<i>n+m</i>)-bit value. For example, {0, REG <i>m</i> [14:0]} yeilds a 16-bit value with 0 in the most significant bit.
- or grey fill	Indicates a reserved bit field in a register. Although these bits can be written to with ones or zeros, they always read zeros.
>>	Shift right logical one position.
<<	Shift left logical one position.
<=	Assignment.
==	Compare equal.
!=	Compare not equal.
>	Greater than.
<	Less than.

Register Access

Register Diagram Field Access Type Legend

This figure provides the interpretation of the notation used in the register diagrams for a number of common field access types:



NOTE

For reserved register fields, the software should mask off the data in the field after a read (the software can't rely on the contents of data read from a reserved field) and always write all zeros.

Register Macro Usage

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. This is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, some hardware registers are implemented as a group, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all the bits set to 1 perform the associated operation on the primary register, while the bits set to 0 are not affected. The SCT registers always read back 0, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one-bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read-modify-write operations. When an atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

A set of SCT registers is offered for registers in many modules on this device, as described in this manual. In a module memory map table, the suffix `_SET`, `_CLR`, or `_TOG` is added to the base name of the register. For example, the `CCM_ANALOG_PLL_ARM` register has three other registers called `CCM_ANALOG_PLL_ARM_SET`, `CCM_ANALOG_PLL_ARM_CLR`, and `CCM_ANALOG_PLL_ARM_TOG`.

In the sub-section that describes one of these sets of registers, a short-hand convention is used to denote that a register has the SCT register set. There is an italicized *n* appended to the end of the short register name. Using the above example, the name used for this register is `CCM_ANALOG_PLL_ARMn`. When you see this designation, there is a SCT register set associated with the register, and you can verify this by checking it in the memory map table. The address offset for each of these registers is given in the form of the following example:

Address: `20C_8000h` base + `0h` offset + $(4d \times i)$, where $i=0d$ to $3d$

In this example, the address for each of the base registers and their three SCT registers can be calculated as:

Register	Address
<code>CCM_ANALOG_PLL_ARM</code>	<code>20C_8000h</code>
<code>CCM_ANALOG_PLL_ARM_SET</code>	<code>20C_8004h</code>
<code>CCM_ANALOG_PLL_ARM_CLR</code>	<code>20c_8008h</code>
<code>CCM_ANALOG_PLL_ARM_TOG</code>	<code>20C_800Ch</code>

Signal Conventions

- `_n, _N, _B`** When appended to a signal name, this indicates that a signal is active-low.
- UPPERCASE** Package pin names, Block I/O signals.
- lowercase** Lowercase is used to indicate internal signals.

Acronyms and Abbreviations

The table below contains acronyms and abbreviations used in this document.

Acronyms and Abbreviated Terms

Term	Meaning
ACMP	Analog Comparator
ADC	Analog-to-Digital Converter

Table continues on the next page...

Table continued from the previous page...

Term	Meaning
AHB	Advanced High-performance Bus
AIPS	Arm IP Bus
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASRC	Asynchronous Sample Rate Converter
AXI	Advanced eXtensible Interface
BIST	Built-In Self Test
CM	Arm Cortex-M33
CAN	Controller Area Network
CB AB	Credential Beacon Authentication beacon
CC	Credential Constraint
CCM	Clock Controller Module
CPIA	Customer Programmable in-field area
CPFA	Customer Programmable Factory area
CPU	Central Processing Unit
CSI	CMOS Sensor Interface
CSU	Central Security Unit
CTI	Cross Trigger Interface
DAP	Debug Access Port
DC	Debug Credential
DCFG_*	Debug Config
DCK	Debug Credential Key
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
ECC	Error correcting codes

Table continues on the next page...

Table continued from the previous page...

Term	Meaning
EDMA	Enhanced Direct Memory Access
EIM	External Interface Module
ENET	Ethernet
EPIT	Enhanced Periodic Interrupt Timer
EPROM	Erasable Programmable Read-Only Memory
ETF	Embedded Trace FIFO
ETM	Embedded Trace Macrocell
FIFO	First-In-First-Out
GIC	General Interrupt Controller
GPC	General Power Controller
GPIO	General-Purpose I/O
GPR	General-Purpose Register
GPS	Global Positioning System
GPT	General-Purpose Timer
GPU	Graphics Processing Unit
GPV	Global Programmers View
HAB	High-Assurance Boot
I2C or I ² C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IOMUX	Input-Output Multiplexer
IP	Intellectual Property
IrDA	Infrared Data Association
JTAG	Joint Test Action Group (a serial bus protocol usually used for test purposes)
LDO	Low-Dropout
LIFO	Last-In-First-Out
LRU	Least-Recently Used

Table continues on the next page...

Table continued from the previous page...

Term	Meaning
LSB	Least-Significant Byte
LUT	Look-Up Table
LVDS	Low Voltage Differential Signaling
MAC	Medium Access Control
MMC	Multimedia Card
MSB	Most-Significant Byte
MT/s	Mega Transfers per second
OCRAM	On-Chip Random-Access Memory
OCOTP	On-Chip One-Time Programmable Controller
PCI	Peripheral Component Interconnect
PCIe	PCI enhanced
PCMCIA	Personal Computer Memory Card International Association
PGC	Power Gating Controller
PIC	Programmable Interrupt Controller
PMU	Power Management Unit
POR	Power-On Reset
PSRAM	Pseudo-Static Random Access Memory
PWM	Pulse Width Modulation
PXP	Pixel Pipeline
QoS	Quality of Service
R2D	Radians to Degrees
RISC	Reduced Instruction Set Computing
ROM	Read-Only Memory
ROMCP	ROM Controller with Patch
RTOS	Real-Time Operating System
RoT	Root of Trust

Table continues on the next page...

Table continued from the previous page...

Term	Meaning
RoTpub	RoT Public Key
RoTID	RoT Identifier
RoTMETA	RoT meta-data
Rx	Receive
SAI	Synchronous Audio Interface
SCU	Snoop Control Unit
SD	Secure Digital
SDIO	Secure Digital Input/Output
SDLC	Synchronous Data Link Control
SDMA	Smart DMA
SIM	Subscriber Identification Module
SoC	System-on-Chip
SoCC	SoC Class
SoCU	SoC Usage
SPBA	Shared Peripheral Bus Arbiter
SPDIF	Sony Phillips Digital Interface
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRC	System Reset Controller
TFT	Thin-Film Transistor
TPIU	Trace Port Interface
TSGEN	Time Stamp Generator
Tx	Transmit
TZASC	TrustZone Address Space Controller
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USDHC	Ultra Secured Digital Host Controller
VU	Vendor Usage

Table continues on the next page...

Table continued from the previous page...

Term	Meaning
WDOG	Watchdog
WLAN	Wireless Local Area Network
WXGA	Wide Extended Graphics Array

Chapter 1

Introduction

1.1 Introduction

The i.MX RT500 is a family of dual-core microcontrollers for embedded applications featuring an Arm Cortex-M33 CPU combined with a Cadence Xtensa Fusion F1 Audio Digital Signal Processor CPU. The Cortex-M33 includes two hardware coprocessors providing enhanced performance for an array of complex algorithms. The family offers a rich set of peripherals and very low power consumption. The device has up to 5 MB SRAM, two FlexSPIs (Octal/Quad SPI Interfaces) each with 32 KB cache, one with dynamic decryption, high-speed USB device/ host + PHY, 12-bit 1 MS/s ADC, Analog Comparator, Audio subsystems supporting up to 8 DMIC channels, 2.5D Vector GPU and LCD Controller with MIPI DSI PHY, 2 SDIO/eMMC; FlexIO; AES/SHA/Crypto M33 coprocessor and PUF key generation.

1.1.1 Block Diagrams

The functional block diagram is shown in the figure below. This diagram provides a view of the chip's major functional components and core complexes.

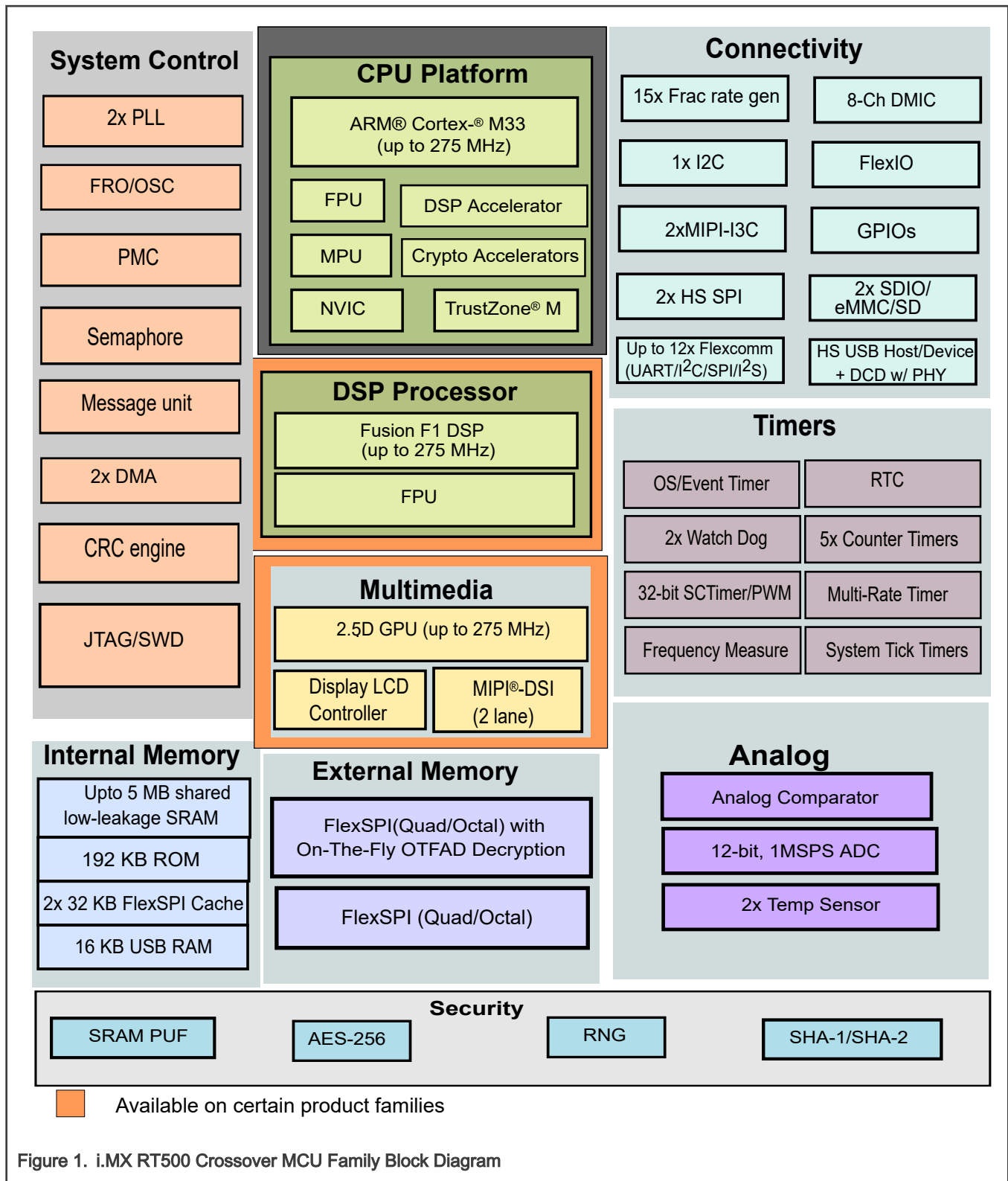


Figure 1. i.MX RT500 Crossover MCU Family Block Diagram

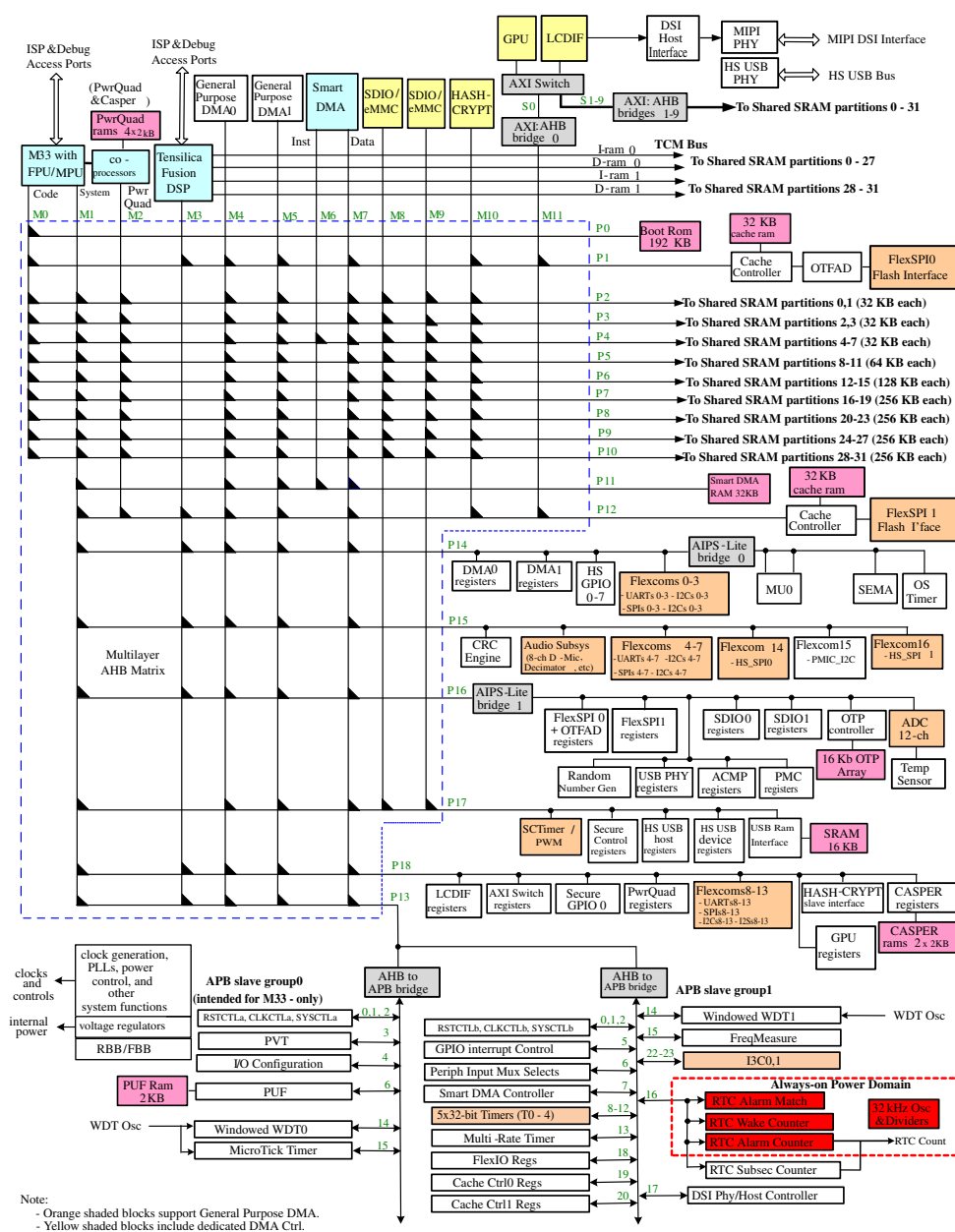


Figure 2. Detailed Block Diagram

1.2 Master ID Assignments

Each master is assigned a unique Master ID that remains static that is used by certain modules like FlexSPI to moderate access.

Master Port	Master Name	Master ID
M0	M33 Code	0000
M1	M33 System	0001

Table continues on the next page...

Table continued from the previous page...

Master Port	Master Name	Master ID
M2	Power Quad	0010
M3	Fusion DSP	0011
M4	DMA0	0100
M5	DMA1	0101
M6	Smart DMA Instruction	0110
M7	Smart DMA Data	0111
M8	SDIO0	1000
M9	SDIO1	1001
M10	HASH_CRYPT	1010
M11	AXI:AHB Bridge	1011

1.3 Features and benefits

- Control processor core
 - Arm Cortex-M33 processor, running at frequencies of up to 275 MHz.
 - Arm TrustZone.
 - Arm Cortex-M33 built-in Memory Protection Unit (MPU) supporting eight regions.
 - Hardware Floating Point Unit (FPU).
 - Arm Cortex-M33 built-in Nested Vectored Interrupt Controller (NVIC).
 - Non-maskable Interrupt (NMI) input.
 - Two coprocessors for the Cortex-M33: a hardware accelerator for fixed and floating point DSP functions (PowerQuad) and a Crypto/FFT engine (Casper).
 - The DSP coprocessor uses a bank of four dedicated 8 KB SRAMs. The Crypto/FFT engine uses a bank of two 2 KB SRAMs that are also AHB accessible by the CPU and the DMA engine.
 - Serial Wire Debug; includes serial wire trace (SWO), ETM Trace (with external buffer), eight breakpoints, four watchpoints, and a debug timestamp counter.
 - Cortex-M33 System tick timer.
- DSP processor core:
 - Cadence Xtensa Fusion F1 DSP processor, running at frequencies of up to 275 MHz.
 - Hardware Floating Point Unit.
 - Serial Wire Debug (shared with Cortex-M33 Control Domain CPU).
 - System tick timer.
 - The DSP coprocessor uses a bank of four dedicated 8 KB SRAMs. The Crypto/FFT engine uses a bank of two 2 KB SRAMs that are also AHB accessible by the CPU and the DMA engine.
- Five I/O Power Rails:
 - Five independent supplies powering different clusters of pins to permit interfacing directly to off-chip peripherals operating at different supply levels.

- On-chip Memory:
 - Up to 5 MB of system SRAM accessible by both CPUs, both DMA engines, the Graphics Subsystem and all other AHB masters.
 - Additional SRAMs for USB traffic (16 KB), Cortex-M33 co-processors (4 x 2 KB), SDIO FIFOs (2 x 512 B dual-port), PUF secure key generation (2 KB), Octal/Quad SPI caches (32 KB each), SmartDMA commands (32KB), and a variety of dual and single port rams for Graphics.
 - 16 KB OTP fuses.
 - Up to 192 KB ROM memory for factory-programmed drivers and APIs.
 - System boot from High-speed SPI, FlexSPI Flash, HS USB, I2C, UART or eMMC via on-chip bootloader software included in ROM. FlexSPI boot mode will include an option for Execute-in-place start-up for nonsecure boot.
- Digital Peripherals
 - Two general-purpose DMA engines, each with 37 channels and up to 27 programmable request/trigger sources.
 - Can be configured such that one DMA is secure and the other is non-secure and/or one can be designated for use by the M33 CPU and the other by the Fusion DSP.
 - Smart DMA Controller with dedicated 32 KB code RAM.
 - USB high-speed host/device controller with on-chip PHY and dedicated DMA controller.
 - Two FlexSPI (Octal/Quad) Interfaces up to 200 MHz DDR/SDR. 32 KB caches with selectable cache policies based on programmable address regions. One of the FlexSPI interface will include on-the-fly decryption for execute-in-place and address remapping to support dual-image boot. DMA supported (both modules).
 - Two SD/eMMC memory card interfaces with dedicated DMA controllers. One supports eMMC 5.0 with HS400/DDR operation.
- Communication interfaces
 - 12 configurable universal serial interface modules (Flexcomm Interfaces). Each module contains an integrated FIFO and DMA support. Each of the modules can be configured as:
 - A USART with dedicated fractional baud rate generation and flow-control handshaking signals. The USART can optionally be clocked at 32 kHz and operated when the chip is in reduced power mode, using either the 32 kHz clock or an externally supplied clock. The USART also provides partial support for LIN2.2.
 - An I2C-bus interface with multiple address recognition, and a monitor mode. It supports 400 Kb/sec Fast-mode and 1 Mb/sec Fast-mode Plus. It also supports 3.4 Mb/sec high-speed when operating in slave mode.
 - An SPI interface.
 - An I2S (Inter-IC Sound) interface for digital audio input or output. Each I2S supports up to four channel-pairs.
 - Two additional high-speed SPI interfaces supporting 50 MHz operation
 - One additional I2C interface with open-drain pads available on some products
 - Two I3C bus interfaces
 - A digital microphone interface supporting up to 8 channels with associated decimators and Voice Activation Detect. One pair of channels can be streamed directly to I2S. The DMIC supports DMA.
- Timers
 - One 32-bit SCTimer/PWM module (SCT). Multi-purpose timer with extensive event-generation, match/compare, and complex PWM and output control features.
 - 8 inputs
 - 10 outputs
 - 16 match/capture registers

- 16 events
- 32 states
- 10 general-purpose/PWM outputs, 8 general-purpose inputs
- It supports DMA and can trigger external DMA events
- It supports fractional match values for high resolution
- Five general purpose, 32-bit timer/counter modules with PWM capability
- 24-bit multi-rate timer module with 4 channels each capable of generating repetitive interrupts at different, programmable frequencies.
- Two Windowed Watchdog Timers (WDT) with dedicated watchdog oscillator (1 MHz LPOSC)
- Frequency measurement module to determine the frequency of a selection of on-chip or off-chip clock sources.
- Real-Time Clock (RTC) with independent power supply and dedicated oscillator. Integrated wake-up timer can be used to wake the device up from low-power modes. The RTC resides in the “always-on” voltage domain. RTC includes eight 32-bit general-purpose registers which can retain contents when power is removed from the rest of the chip.
- Ultra-low power micro-tick Timer running from the Watchdog oscillator with capture capability for timestamping. Can be used to wake up the device from low-power modes.
- 64-bit OS/Event Timer common to both processors with individual match/capture and interrupt generation logic. Enabled on POR.
- Clock Generation Unit
 - Crystal oscillator with an operating range of 4 MHz to 26 MHz.
 - Internal 192/96 MHz FRO oscillator. Trimmed to 1% accuracy.
 - FRO capable of being tuned using an accurate reference clock (eg. XTAL Osc) to 0.1% accuracy with 46% duty cycle.
 - Internal 1 MHz low-power oscillator with 10% accuracy. Serves as the watchdog oscillator and clock for the OS/Event Timer and the SysTick among others. Also available as the system clock to both domains.
 - 32 kHz real-time clock (RTC) oscillator that can optionally be used as a system clock.
 - Main/System PLL:
 - allows CPU operation up to the maximum rate without the need for a high-frequency crystal. May be run from the FRO, the crystal oscillator or the CLKIN pin.
 - a second, independent PLL output provides alternate high-frequency clock source for the Fusion DSP CPU if the required frequency is different from the main system clock. (Note: 2nd PFD output from Main System PLL)
 - two additional PLL outputs provide potential clock sources to various peripherals.
 - Audio PLL for the audio subsystem.
- Power Control
 - Main external power supply: 1.8V $\pm$ 5%
 - Vddcore supply (from PMIC or internal PMU): adjustable from 0.6 V to 1.1 V
 - Five VDDIO supplies (can be shared or independent): 1.71 - 3.6 V
 - USB Supply: 3.0-3.6 V
 - Reduced power modes:
 - Sleep mode: CPU clock shut down (each CPU independently)
 - Deep_sleep mode: User-selectable configuration via PDSLEEPCFG

- Deep_powerdown mode: Internal power removed from entire chip except “always-on” domain
- Each individual SRAM partition can be independently powered-off or put into a low-power retain mode
- Fusion DSP Domain can be powered-off independently from the rest of the system.
- Ability to operate the synchronous serial interfaces in sleep or deep-sleep as a slave or USART clocked by the 32 kHz RTC oscillator
- Wake-up from low-power modes via interrupts from various peripherals including the RTC and the OS/Event timer
- RBB/FBB to provide additional control over power/performance trade-offs
- Power-On Reset (POR).
- I/O Peripherals
 - Up to 136 general purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors. Ports can be written as words, half-words, bytes, or bits.
 - Mirrored, secure GPIO0.
 - Individual GPIO pins can be used as edge and level sensitive interrupt sources, each with its own interrupt vector.
 - All GPIO pins can contribute to one of two ganged (OR'd) interrupts from the GPIO_HS module.
 - A group of up to 8 GPIO pins (from Port0/1) can be selected for Boolean pattern matching which can generate interrupts and/or drive a “pattern-match” output.
 - Adjustable output driver slew rates.
 - JTAG boundary scan
- Graphics/Multimedia
 - 2.5D Graphics Processing Unit, running at frequencies of up to 275 MHz.
 - LCD Display Interface supporting smart LCD displays and video mode.
 - MIPI DSI Interface with on-chip PHY supporting transfer rates up to 895.1 Mbps.
 - FlexIO can be configured to provide a parallel interface to an LCD
- Analog Peripherals
 - One 12-bit ADC with sampling rates of 1 Msamples/sec and an enhanced ADC controller. It supports up to 6 single-ended channels or 2 differential channels. The ADC supports DMA.
 - Temperature sensor.
 - Analog comparator
- Security
 - AES256 Encryption module. Random number generator which can be used to create keys. Key storage in OTP. DMA supported
 - PUF (Physical Unclonable Function) key generation module
 - SHA1/SHA2 Secure Hash Algorithm module. Supports secure boot. Hash crypt engine with built-in DMA controller connected to OTFAD, shared SRAM and FlexSPI. Dedicated DMA controller or can be used with general-purpose DMA
 - Cryptography hardware co-processor attached to Cortex M33 CPU
- Operating Characteristics
 - Temperature range (ambient): -20 °C to +70 °C
 - VDDCORE: 0.6V - 1.155V

- VDDIO_0/1/2/4: 1.71 V - 1.89 V
- VDDIO_3: 1.71 V - 3.6 V

1.4 Target Applications

This processor can be used in areas such as industrial , IoT, motor control and home appliances, etc.

Specifically, RT500 targets wearables, outdoor GPS, low-end thermostats and security panels, along with low-end systems with displays.

1.5 Endianness Support

The chip supports only the Little Endian mode.

Chapter 2 Memory Maps

2.1 Memory system overview

This section introduces the memory architecture of the chip.

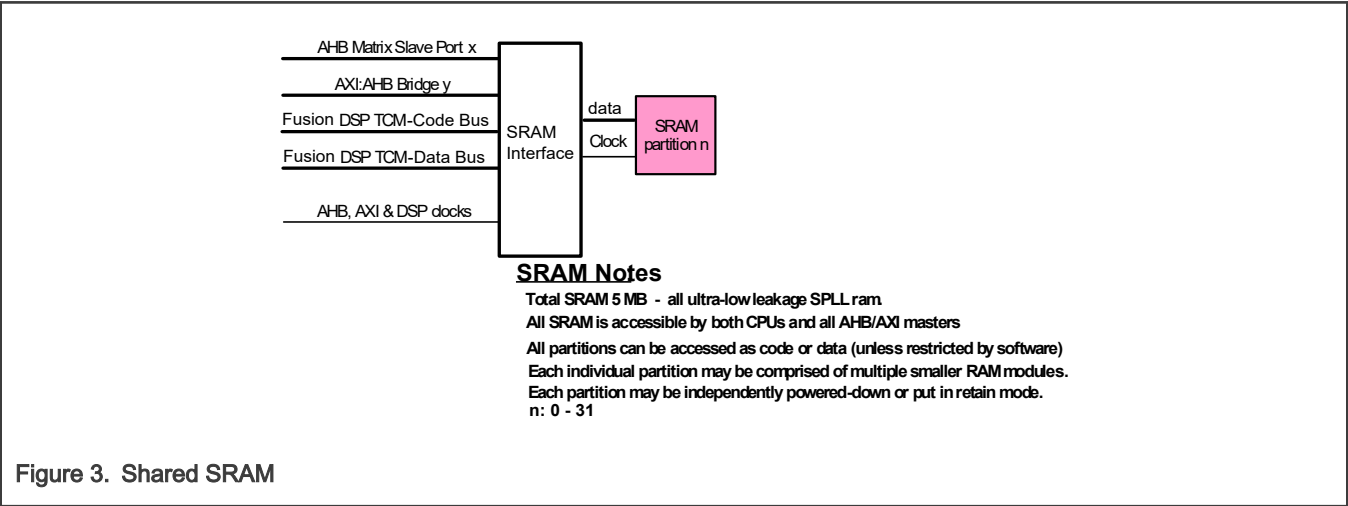
This device provides up to 5 MB of centrally-located SRAM. This SRAM will be divided into a collection of separate partitions. There will be 32 such partitions ranging in size from 32 KB to 256 KB each, as shown in AXI-to-RAM Architecture figure below.

All SRAM partitions are accessible by both CPUs or by any other AHB bus master (including the DMAs). The Fusion DSP accesses these ram partitions via its dedicated, tightly-couple memory (TCM) buses. The GPU and the LCD Display Interface (Display Controller) will access this ram via dedicated 64 bit AHB buses. The M33, DMA, and all other AHB masters access the rams via the AHB matrix. A ram interface associated with each partition arbitrates between the AHB matrix, the graphics AHB buses and the Fusion DSP TCMs. The user may elect to restrict certain partitions to the exclusive use of just the Fusion or just the AHB or AXI (graphics) masters. All of the SRAMs will be mapped into both the Code and Data address spaces of the M33 processor so any of the partitions may be accessed from either the C-code or System buses. Partitions will be distributed among several AHB slave ports (on each matrix) to minimize conflicts.

Moreover, because there is delay each time a ram partition is handed-over from one master to another, it is strongly recommended not to share partitions among masters that will be frequently or continuously competing for access. Frequent re-arbitration will seriously degrade system performance. Sharing ram partitions should be limited to cases where conflicts are relatively infrequent such as “ping-pong” scenarios where one master fills a ram and then another takes it to access that data.

Each SRAM partition is constructed from some number of 32 KB, 64 KB or 128 KB RAM blocks. Each block of memory is constructed of a periphery containing the sense amps and the row and column decoders, and the actual memory array. The array and the periphery have independent power switches on each of the instances. The periphery and/or the array can be turned off depending on how much RAM needs to be kept alive. These periphery and array switches will be controlled on a partition-by-partition basis meaning that all of the individual RAM modules comprising a single partition will be controlled together. The RAM periphery and array power switches are controlled via the PDRUNCFG and PDSLEEPCFG mechanisms.

The following figures show the Shared RAM and AXI-to-RAM Architecture Detail.



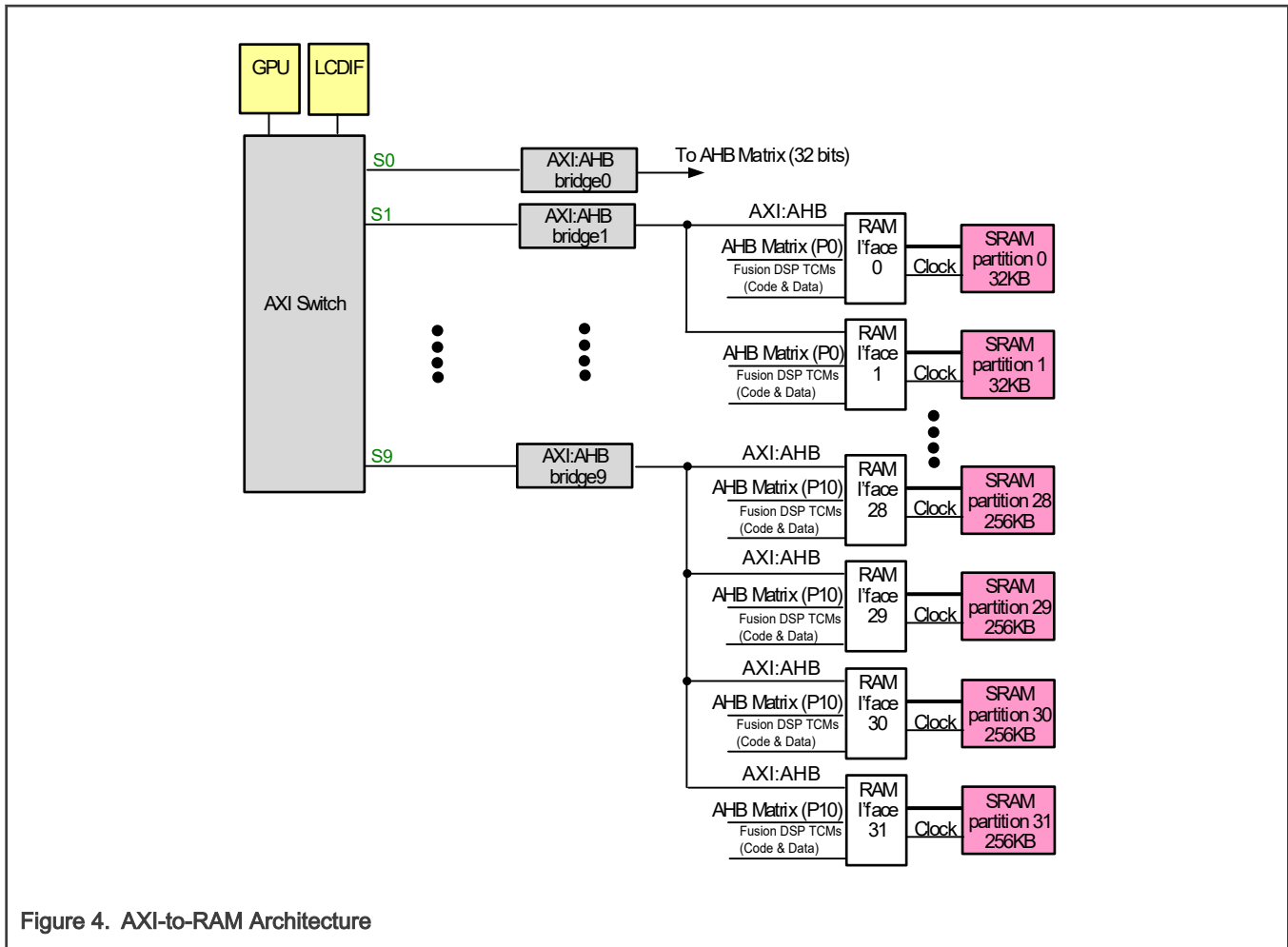


Figure 4. AXI-to-RAM Architecture

2.2 System memory map

The following table shows the device's high-level memory map as seen by the Arm Cortex-M33 core. Additionally, the various sections are split into "Secure" and "Non-Secure" apertures for each IP. Although these apertures address the same IP, access through the "Secure" aperture can only be performed by code with the appropriate security settings. The table also shows the corresponding AHB port numbers.

- Only Shared RAM partition 0 are powered through a reset. All other RAM partitions will lose their contents when any reset occurs. The address range for powered SRAM is:
 - 0x0000_0000 - 0x0000_7FFF
 - 0x1000_0000 - 0x1000_7FFF
 - 0x2000_0000 - 0x2000_7FFF
 - 0x3000_0000 - 0x3000_7FFF
- Shared RAM partitions 2-3 are reserved for bootloader use when the bootloader is running. They need to be reserved for the ROM API use before calling the ROM APIs in user application (IAP calls). SRAM reserved regions are:
 - 0x0001_0000 - 0x0001_FFFF
 - 0x1001_0000 - 0x1001_FFFF
 - 0x2001_0000 - 0x2001_FFFF
 - 0x3001_0000 - 0x3001_FFFF

Table 2. System memory map (M33/DSP/AHB overview)

Non-Secure Start address	Secure Start address	Description	Name	Size	Access	DSP Access
0000_0000	1000_0000	Shared RAM partitions 0-31 via M33 code bus (overlaps data region)	SRAM	5 MB	AHB Ports 2 - 10 (P2 - P10)	Yes
0300_0000	1300_0000	Read Only Memory - Boot (ROM-BOOT)	BOOT-ROM	192 KB	AHB Port 0 (P0)	No
0800_0000	1800_0000	FlexSPI0 (Quad/Octal SPI0) Interface memory-mapped space	FlexSPI0	128 MB	AHB Port 1 (P1)	Yes
2000_0000	3000_0000	Shared RAM partitions 0-1 via data bus (overlaps M33 code region)	SRAM	5 MB	AHB Ports 2 - 10 (P2 - P10)	Yes
2410_0000	3410_0000	Smart DMA Controller RAM	Smart DMA RAM	32 KB	AHB Port 11 (P11)	No
2800_0000	3800_0000	FlexSPI1 (Quad/Octal SPI1) Interface memory-mapped space	FlexSPI1	128 MB	AHB Port 12 (P12)	Yes
4000_0000	5000_0000	APB0 (32 Peripheral slots)	APB0	128 KB	AHB Port 13 (P13)	Yes
4002_0000	5002_0000	APB1 (32 Peripheral slots)	APB1	128 KB	AHB Port 13 (P13)	Yes
4010_0000	5010_0000	AHB Peripherals	AHB	128 KB	AHB Port 14 (P14)	Yes
4012_0000	5012_0000	AHB Peripherals	AHB	64 KB	AHB Port 15 (P15)	Yes
4013_0000	5013_0000	AHB Peripherals	AHB	64 KB	AHB Port 16 (P16)	Yes
4014_0000	5014_0000	AHB Peripherals	AHB	768 KB	AHB Port 17 (P17)	No
4020_0000	5020_0000	AHB Peripherals	AHB	512 KB	AHB Port 18 (P18)	Yes
E000_0000	F000_0000	Private Peripheral Bus (Internal). Includes NVI and SYSTICK timer	PPB	256 KB		No
E004_0000	F004_0000	Private Peripheral Bus (External)	EPPPB	768 KB		No

2.3 Shared RAM Address Map

A variety of shared RAM partition sizes are provided to allow more flexibility in assigning the uses of those spaces. For each application, shared RAM usage should be planned to minimize collision of accesses by the two buses of the Cortex-M33, as well as other bus masters, including DMA controllers and the HiFi4.

A best case would be if each shared RAM partition is accessed by only one master at any particular time, "ownership" being passed to another master (for instance) when a buffer is filled from a peripheral, a block of data is processed by an algorithm, etc.

To summarize, access collisions can occur under the following conditions:

- On the AHB matrix: when two AHB masters access a resource on the same slave port at the same time. AHB masters include the HiFi4 when it is using the AHB matrix, not when it is accessing shared RAM.
- HiFi4 accessing shared RAM: when the HiFi4 and an AHB master access the same shared RAM partition at the same time. Note that in this case, the access collision happens at the partition, not at the slave port. Since there are multiple partitions for each slave port, this allows even more opportunity to avoid collisions.

Table 3. Shared RAM Address Map

Partition numbers	Size	M33 code/DSP code Address	Fusion DSP Data Address	All other AHB masters (including M33 System bus & DMA) and GPU/LCD Address
0	32 KB	0x0000 0000	0x0080 0000	0x2000 0000 (AHB P2)
1	32 KB	0x0000 8000	0x0080 8000	0x2000 8000 (AHB P2)
2	32 KB	0x0001 0000	0x0081 0000	0x2001 0000 (AHB P3)
3	32 KB	0x0001 8000	0x0081 8000	0x2001 8000 (AHB P3)
4	32 KB	0x0002 0000	0x0082 0000	0x2002 0000 (AHB P4)
5	32 KB	0x0002 8000	0x0082 8000	0x2002 8000 (AHB P4)
6	32 KB	0x0003 0000	0x0083 0000	0x2003 0000 (AHB P4)
7	32 KB	0x0003 8000	0x0083 8000	0x2003 8000 (AHB P4)
8	64 KB	0x0004 0000	0x0084 0000	0x2004 0000 (AHB P5)
9	64 KB	0x0005 0000	0x0085 0000	0x2005 0000 (AHB P5)
10	64 KB	0x0006 0000	0x0086 0000	0x2006 0000 (AHB P5)
11	64 KB	0x0007 0000	0x0087 0000	0x2007 0000 (AHB P5)
12	128 KB	0x0008 0000	0x0088 0000	0x2008 0000 (AHB P6)
13	128 KB	0x000A 0000	0x008A 0000	0x200A 0000 (AHB P6)
14	128 KB	0x000C 0000	0x008C 0000	0x200C 0000 (AHB P6)
15	128 KB	0x000E 0000	0x008E 0000	0x200E 0000 (AHB P6)
16	256 KB	0x0010 0000	0x0090 0000	0x2010 0000 (AHB P7)
17	256 KB	0x0014 0000	0x0094 0000	0x2014 0000 (AHB P7)
18	256 KB	0x0018 0000	0x0098 0000	0x2018 0000 (AHB P7)
19	256 KB	0x001C 0000	0x009C 0000	0x201C 0000 (AHB P7)
20	256 KB	0x0020 0000	0x00A0 0000	0x2020 0000 (AHB P8)
21	256 KB	0x0024 0000	0x00A4 0000	0x2024 0000 (AHB P8)
22	256 KB	0x0028 0000	0x00A8 0000	0x2028 0000 (AHB P8)
23	256 KB	0x002C 0000	0x00AC 0000	0x202C 0000 (AHB P8)

Table continues on the next page...

Table 3. Shared RAM Address Map (continued)

Partition numbers	Size	M33 code/DSP code Address	Fusion DSP Data Address	All other AHB masters (including M33 System bus & DMA) and GPU/LCD Address
24	256 KB	0x0030 0000	0x00B0 0000	0x2030 0000 (AHB P9)
25	256 KB	0x0034 0000	0x00B4 0000	0x2034 0000 (AHB P9)
26	256 KB	0x0038 0000	0x00B8 0000	0x2038 0000 (AHB P9)
27	256 KB	0x003C 0000	0x00BC 0000	0x203C 0000 (AHB P9)
28	256 KB	0x0040 0000	0x00C0 0000	0x2040 0000 (AHB P10)
29	256 KB	0x0044 0000	0x00C4 0000	0x2044 0000 (AHB P10)
30	256 KB	0x0048 0000	0x00C8 0000	0x2048 0000 (AHB P10)
31	256 KB	0x004C 0000	0x00CC 0000	0x204C 0000 (AHB P10)

2.4 APB0 (32 Peripheral slots)

The following table provides details of the non-secure addresses for APB peripherals. APB peripherals have both secure and non-secure access possibilities.

The start address of APB0 peripherals for the secure access is 5000_0000. See [Table 2](#).

Table 4. APB0 (32 Peripheral slots)

Base	Slot	Module	Name
SYS Modules			
4000_0000	0	Reset control group A	RSTCTL0
4000_1000	1	Clock control group A	CLKCTL0
4000_2000	2	System control group A	SYSCTL0
4000_3000	3	Reserved	—
4000_4000	4	Pin function selection and pin control setup 0	IOPCTL
4000_5000	5	Reserved	—
4000_6000	6	PUF 0	PUF
SEC Modules			
4000_7000	7	Reserved	—
4000_8000	8	Reserved	—
4000_9000	9	Reserved	—
4000_A000	10	Reserved	—
4000_B000	11	Reserved	—
4000_C000	12	Reserved	—
4000_D000	13	Reserved	—

Table continues on the next page...

Table 4. APB0 (32 Peripheral slots) (continued)

Base	Slot	Module	Name
TMR Modules			
4000_E000	14	Windowed watchdog timer 0	WWDT0
4000_F000	15	Microtick timer 0	UTICK0
4001_0000	16	Reserved	—
4001_1000	17	Reserved	—
4001_2000	18	Reserved	—
4001_3000	19	Reserved	—
4001_4000	20	Reserved	—
4001_5000	21	Reserved	—
4001_6000	22	Reserved	—
4001_7000	23	Reserved	—
4001_8000	24	Reserved	—
4001_9000	25	Reserved	—
4001_A000	26	Reserved	—
4001_B000	27	Reserved	—
4001_C000	28	Reserved	—
4001_D000	29	Reserved	—
4001_E000	30	Reserved	—
4001_F000	31	Reserved	—

2.5 APB1 (32 Peripheral slots)

The following table provides details of the non-secure addresses for APB peripherals. APB peripherals have both secure and non-secure access possibilities.

The start address of APB1 peripherals for the secure access is 5002_0000. See [Table 2](#).

Table 5. APB1 (32 Peripheral slots)

Base	Slot	Module	Name
SYS Modules			
4002_0000	0	Reset control group B	RSTCTL1
4002_1000	1	Clock control group B	CLKCTL1
4002_2000	2	System control group B	SYSCTL1
4002_3000	3	Reserved	—
4002_4000	4	Reserved	—
4002_5000	5	GPIO interrupts	GPIO_INT

Table continues on the next page...

Table 5. APB1 (32 Peripheral slots) (continued)

Base	Slot	Module	Name
4002_6000	6	Peripheral Multiplexing Controls	PERIPHERAL MUXES
DMA Modules			
4002_7000	7	Smart DMA Controller 0	SmartDMA
TMR Modules			
4002_8000	8	Standard counter/timer 0	CT32B0
4002_9000	9	Standard counter/timer 1	CT32B1
4002_A000	10	Standard counter/timer 2	CT32B2
4002_B000	11	Standard counter/timer 3	CT32B3
4002_C000	12	Standard counter/timer 4	CT32B4
4002_D000	13	Multi-Rate Timer 0	MRT0
4002_E000	14	Windowed watchdog timer 1	WWDT1
4002_F000	15	Frequency measurement unit	FREQMEASURE
4003_0000	16	RTC and Wake-up timer 0	RTC/WAKEUP
HMI Modules			
4003_1000	17	DSI Host Controller 0	DSI Host Controller
COM Modules			
4003_2000	18	Flexible I/O 0	FLEXIO Registers
MEM Modules			
4003_3000	19	Cache Control Registers 0	CACHE_Control_0 Registers
4003_4000	20	Cache Control Registers 1	CACHE_Control_1 Registers
4003_5000	21	Reserved	—
COM Modules			
4003_6000	22	I3C Interface 0	I3C0
4003_7000	23	I3C Interface 1	I3C1
4003_8000	24	Reserved	—
4003_9000	25	Reserved	—
4003_A000	26	Reserved	—
4003_B000	27	Reserved	—

Table continues on the next page...

Table 5. APB1 (32 Peripheral slots) (continued)

Base	Slot	Module	Name
4003_C000	28	Reserved	—
4003_D000	29	Reserved	—
4003_E000	30	Reserved	—
TMR Modules			
4003_F000	31	Reserved	—

2.6 AHB Peripherals

The following table provides details of the non-secure base addresses for AHB peripherals. AHB peripherals have both secure and non-secure access possibilities. Some AHB matrix ports are accessible by the DSP core (for peripherals that are not secure) and some are accessible by only the Cortex-M33 core.

NOTE

Flexcomms 0-8 are available on this chip. Please disregard references to Flexcomm 9-13.

The start address of AHB peripherals for the secure access is 4010_0000. See [Table 2](#).

Table 6. AHB Peripherals

Base	Module	Name	AHB Port	Accessible by Fusion DSP
SYS Modules				
4010_0000	High Speed GPIO (general purpose I/O for port pins that are not selected for some other function by IOPCTL)	HS-GPIO	P14	Yes
DMA Modules				
4010_4000	DMA registers 0	DMA0	P14	Yes
4010_5000	DMA registers 1	DMA1		
COM Modules				
4010_6000	Flexcomm Interface 0	FLEXCOMM0	P14	Yes
4010_7000	Flexcomm Interface 1	FLEXCOMM1		
4010_8000	Flexcomm Interface 2	FLEXCOMM2		
4010_9000	Flexcomm Interface 3	FLEXCOMM3		
4010_A000	Reserved	—		
4010_B000	Reserved	—		
DBG Modules				
4010_F000	Debug mailbox 0	Debug	P14	Yes
CORE Modules				
4011_0000	Messaging Unit 0 (Cortex-M33 port)	MU	P14	Yes

Table continues on the next page...

Table 6. AHB Peripherals (continued)

Base	Module	Name	AHB Port	Accessible by Fusion DSP
4011_1000	Messaging Unit 0 (Fusion DSP port)	MU		
4011_2000	Semaphore 0	SEMA4		
TMR Modules				
4011_3000	OS_Event Timer 0 (for access by Cortex-M33)	OSTIMER_CM33	P14	
4011_4000	OS_Event Timer 1 (for access by Fusion DSP)	OSTIMER_DSP	P14	Yes
MEM Modules				
4011_5000	ROM Controller Patch 0	ROMC	P14	Yes
4011_7000	Reserved	—		
4011_8000	Reserved	—		
COM Modules				
4012_0000	CRC Engine 0	CRC	P15	Yes
4012_1000	D-MIC (8 channel PDM digital microphone interface) 0	DMIC		
4012_2000	Flexcomm Interface 4	FLEXCOMM4		
4012_3000	Flexcomm Interface 5	FLEXCOMM5		
4012_4000	Flexcomm Interface 6	FLEXCOMM6		
4012_5000	Flexcomm Interface 7	FLEXCOMM7		
4012_6000	Flexcomm Interface (HS-SPI) 14	FLEXCOMM14		
4012_7000	Flexcomm Interface (PMIC-I2C) 15	FLEXCOMM15		
4012_8000	Flexcomm Interface (HS-SPI) 16	FLEXCOMM16 (HS-SPI)		
4012_9000	Reserved	—		
SYS Modules				
4013_0000	OTP Controller 0	OCOTP	P16	Yes
HMI Modules				
4013_4000	FlexSPI0 (Quad/Octal) + OTFAD Registers 0	FlexSPI0	P1	Yes
SYS Modules				
4013_5000	PMC (PMU Control) 0	PMU	P16	Yes
HMI Modules				
4013_6000	SDIO Registers 0	SDIO0	P16	Yes
4013_7000	SDIO Registers 1	SDIO1		
SEC Modules				
4013_8000	Random Number Generator 0	RNG	P16	Yes

Table continues on the next page...

Table 6. AHB Peripherals (continued)

Base	Module	Name	AHB Port	Accessible by Fusion DSP
ANA Modules				
4013_9000	Analog Comparator 0	ACMP0	P16	Yes
4013_A000	Analog-to-Digital Converter 0	ADC0		
COM Modules				
4013_B000	HS-USB PHY 0	USBPHY	P16	Yes
4013_C000	FlexSPI1 (Quad/Octal) registers	FlexSPI1	P12	Yes
4013_D000	Reserved	—		
4013_E000	Reserved	—		
4013_F000	Reserved	—		
4014_0000	HS USB SRAM Interface	USB-HS	P17	No
4014_4000	HS USB Device Registers	USB-HS		
4014_5000	HS USB Host Registers	USB-HS		
TMR Modules				
4014_6000	SCTimer/PWM 0	SCT	P17	No
4014_7000	Reserved	—		
SEC Modules				
4014_8000	Security Control Registers 0	—	P17	No
4014_C000	Reserved	—		
CORE Modules				
4020_0000	PowerQuad Coprocessor	PowerQuad	P18	Yes
4020_1000	Casper Coprocessor	Casper		
4020_2000	Casper RAM Interface	Casper RAM		
4020_3000	Reserved	—		
SYS Modules				
4020_4000	Secure HS GPIO(0)	Secure HS GPIO	P18	Yes
SEC Modules				
4020_8000	HashCrypt Registers	HashCrypt	P18	Yes
COM Modules				
4020_9000	Flexcomm Interface 8	FLEXCOMM8	P18	Yes
4020_A000	Flexcomm Interface 9	FLEXCOMM9		
4020_B000	Flexcomm Interface 10	FLEXCOMM10		
4020_C000	Flexcomm Interface 11	FLEXCOMM11		

Table continues on the next page...

Table 6. AHB Peripherals (continued)

Base	Module	Name	AHB Port	Accessible by Fusion DSP
4020_D000	Flexcomm Interface 12	FLEXCOMM12		
4020_E000	Flexcomm Interface 13	FLEXCOMM13		
4020_F000	Reserved	—		
HMI Modules				
4021_0000	LCDIF Registers 0	LCDIF	P18	Yes
4021_8000	Reserved	—		
4024_0000	GPU Registers	GPU	P18	Yes
MEM Modules				
4028_0000	AXI Switch Registers	AXI	P18	Yes

Chapter 3

Interrupt, DMA Events, and DSP Assignments

3.1 Nested vector interrupts

The following table presents the Arm M33 NVIC assignments.

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

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

Table 7. Arm M33 NVIC assignments

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0000	0	-	-	-	MSP initial value	MSP initial value
0000_0004	1	-	-	-	Reset	Reset
0000_0008	2	-	-	-	NMI	NMI
0000_000C	3	-	-	-	Hard Fault	Hard Fault
0000_0010	4	-	-	-	MemManage Fault	MemManage Fault
0000_0014	5	-	-	-	Bus Fault	Bus Fault
0000_0018	6	-	-	-	Usage	Usage
0000_001C	7	-	-	-	SecureFault	SecureFault
0000_0020	8	-	-	-	Reserved	Reserved
0000_0024	9	-	-	-	Reserved	Reserved
0000_0028	10	-	-	-	Reserved	Reserved
0000_002C	11	-	-	-	SVCall	SVCall
0000_0030	12	-	-	-	Debug Monitor	Debug Monitor
0000_0034	13	-	-	-	Reserved	Reserved
0000_0038	14	-	-	-	PendSV	PendSV
0000_003C	15	-	-	-	SysTick	SysTick
0000_0040	16	0	0	0	WDT0_IRQ	Watchdog timer interrupt
0000_0044	17	1	0	0	DMA0_IRQ	DMA interrupt
0000_0048	18	2	0	0	GPIO_INTA	GPIO Interrupt A
0000_004C	19	3	0	0	GPIO_INTB	GPIO Interrupt B
0000_0050	20	4	0	1	PIN_INT0_IRQ	Pin interrupt 0 or pattern match engine slice 0

Table continues on the next page...

Table 7. Arm M33 NVIC assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0054	21	5	0	1	PIN_INT1_IRQ	Pin interrupt 1 or pattern match engine slice 1
0000_0058	22	6	0	1	PIN_INT2_IRQ	Pin interrupt 2 or pattern match engine slice 2
0000_005C	23	7	0	1	PIN_INT3_IRQ	Pin interrupt 3 or pattern match engine slice 3
0000_0060	24	8	0	2	UTICK0_IRQ	Microtick interrupt
0000_0064	25	9	0	2	MRT0_IRQ	Multi-Rate Timer
0000_0068	26	10	0	2	CTIMER0_IRQ	CTimer interrupt
0000_006C	27	11	0	2	CTIMER1_IRQ	CTimer interrupt
0000_0070	28	12	0	3	SCT0_IRQ	SCTimer interrupt
0000_0074	29	13	0	3	CTIMER3_IRQ	CTimer interrupt
0000_0078	30	14	0	3	FLEXCOMM0_IRQ	FlexComm interrupt
0000_007C	31	15	0	3	FLEXCOMM1_IRQ	FlexComm interrupt
0000_0080	32	16	0	4	FLEXCOMM2_IRQ	FlexComm interrupt
0000_0084	33	17	0	4	FLEXCOMM3_IRQ	FlexComm interrupt
0000_0088	34	18	0	4	FLEXCOMM4_IRQ	FlexComm interrupt
0000_008C	35	19	0	4	FLEXCOMM5_IRQ	FlexComm interrupt
0000_0090	36	20	0	5	FLEXCOMM14_IRQ	FlexComm interrupt. Standalone SPI
0000_0094	37	21	0	5	FLEXCOMM15_IRQ	FlexComm interrupt. Standalone I2C
0000_0098	38	22	0	5	ADC0_IRQ	Analog-to-Digital Converter interrupt
0000_009C	39	23	0	5	Reserved	-
0000_00A0	40	24	0	6	ACMP_IRQ	Analog Comparator interrupt
0000_00A4	41	25	0	6	DMIC0	Digital Microphone Interface interrupt
0000_00A8	42	26	0	6	Reserved	-
0000_00AC	43	27	0	6	HYPERVISOR_IRQ	Hypervisor interrupt
0000_00B0	44	28		7	SECURE_VIOLATION_IRQ	Secure violation interrupt
0000_00B4	45	29	0	7	HWVAD	Hardware Voice Activity Detector interrupt
0000_00B8	46	30	0	7	Reserved	-
0000_00BC	47	31	0	7	RNG_IRQ	Random Number Generator interrupt

Table continues on the next page...

Table 7. Arm M33 NVIC assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_00C0	48	32	1	8	RTC_IRQ	Real Time Clock Alarm interrupt OR Real Time Clock Wakeup timer interrupt
0000_00C4	49	33	1	8	DSP_TIE_EXPSTATE1	DSP interrupt
0000_00C8	50	34	1	8	MU_A_IRQ	Messaging Unit interrupt
0000_00CC	51	35	1	8	PIN_INT4_IRQ	Pin interrupt 4 or pattern match engine slice 4
0000_00D0	52	36	1	9	PIN_INT5_IRQ	Pin interrupt 5 or pattern match engine slice 5
0000_00D4	53	37	1	9	PIN_INT6_IRQ	Pin interrupt 6 or pattern match engine slice 6
0000_00D8	54	38	1	9	PIN_INT7_IRQ	Pin interrupt 7 or pattern match engine slice 7
0000_00DC	55	39	1	9	CTIMER2_IRQ	CTimer interrupt
0000_00E0	56	40	1	10	CTIMER4_IRQ	CTimer interrupt
0000_00E4	57	41	1	10	OS_EVENT_TIMER_M33_IRQ	Event timer M33 Wakeup/interrupt
0000_00E8	58	42	1	10	FlexSPI0_IRQ (Quad/Octal SPI0) OR FlexSPI1_IRQ	Flex (Quad/Octal) Serial Peripheral Interface interrupt OR Flex Serial Peripheral Interface interrupt
0000_00EC	59	43	1	10	FLEXCOMM6_IRQ	FlexComm interrupt
0000_00F0	60	44	1	11	FLEXCOMM7_IRQ	FlexComm interrupt
0000_00F4	61	45	1	11	SDIO0_IRQ	SDIO (uSDHC) interrupt
0000_00F8	62	46	1	11	SDIO1_IRQ	SDIO (uSDHC) interrupt
0000_00FC	63	47	1	11	SGPIO_INTA	Secure GPIO HS interrupt A
0000_0100	64	48	1	12	SGPIO_INTB	Secure GPIO HS interrupt B
0000_0104	65	49	1	12	I3C0_IRQ	Improved Inter Integrated Circuit interrupt
0000_0108	66	50	1	12	USB0_IRQ	Universal Serial Bus interrupt
0000_010C	67	51	1	12	USB0_NEEDCLK	Universal Serial Bus interrupt
0000_0110	68	52	1	13	WDT1_IRQ	Watchdog Timer interrupt
0000_0114	69	53	1	13	USB_PHYDCD_IRQ	USBPHY DCD interrupt
0000_0118	70	54	1	13	DMA1_IRQ	DMA interrupt
0000_011C	71	55	1	13	PUF_IRQ	QuidKey interrupt
0000_0120	72	56	1	14	POWERQUAD_IRQ	Powerquad interrupt

Table continues on the next page...

Table 7. Arm M33 NVIC assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0124	73	57	1	14	CASPER_IRQ	Casper interrupt
0000_0128	74	58	1	14	PMU_PMIC_IRQ	PMU interrupt
0000_012C	75	59	1	14	SHA_IRQ	SHA interrupt
0000_0130	76	60	1	15	FLEXCOMM8_IRQ	FlexComm interrupt
0000_0134	77	61	1	15	FLEXCOMM9_IRQ	FlexComm interrupt
0000_0138	78	62	1	15	FLEXCOMM10_IRQ	FlexComm interrupt
0000_013C	79	63	1	15	FLEXCOMM11_IRQ	FlexComm interrupt
0000_0140	80	64	2	16	FLEXCOMM12_IRQ	FlexComm interrupt
0000_0144	81	65	2	16	FLEXCOMM13_IRQ	FlexComm interrupt
0000_0148	82	66	2	16	FLEXCOMM16_IRQ	FlexComm interrupt
0000_014C	83	67	2	16	I3C1_IRQ	Improved Inter Integrated Circuit interrupt
0000_0150	84	68	2	17	FLEXIO_IRQ	Flexible I/O interrupt
0000_0154	85	69	2	17	LCDIF_IRQ	Liquid Crystal Display interface interrupt
0000_0158	86	70	2	17	GPU_IRQ	Graphics Processor Unit interrupt
0000_015C	87	71	2	17	MIPI_IRQ	MIPI interrupt
0000_0160	88	72	2	18	Reserved	-
0000_0164	89	73	2	18	SmartDMA_IRQ	Smart DMA Engine Controller interrupt

3.1.1 Interrupt priority levels

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

	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
R	IRQ3								IRQ2								IRQ1								IRQ0							
W																																

3.2 DMA request configuration

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals, as shown in the table below. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

3.3 DMA request assignments

Table 8. DMA0 and DMA1 request assignments

Slot Number	Alias	Source Description
COM Modules		
0	FLEXCOM0_RX_DMA	DMA receive request
1	FLEXCOM0_TX_DMA	DMA transmit request
2	FLEXCOM1_RX_DMA	DMA receive request
3	FLEXCOM1_TX_DMA	DMA transmit request
4	FLEXCOM2_RX_DMA	DMA receive request
5	FLEXCOM2_TX_DMA	DMA transmit request
6	FLEXCOM3_RX_DMA	DMA receive request
7	FLEXCOM3_TX_DMA	DMA transmit request
8	FLEXCOM4_RX_DMA	DMA receive request
9	FLEXCOM4_TX_DMA	DMA transmit request
10	FLEXCOM5_RX_DMA	DMA receive request
11	FLEXCOM5_TX_DMA	DMA transmit request
12	FLEXCOM6_RX_DMA	DMA receive request
13	FLEXCOM6_TX_DMA	DMA transmit request
14	FLEXCOM7_RX_DMA	DMA receive request
15	FLEXCOM7_TX_DMA	DMA transmit request
16	DMIC_CH0_DMA/ FLEXCOMM8_RX_DMA	Channel 0 DMA request/Flexcomm receive request
17	DMIC_CH1_DMA/ FLEXCOMM8_TX_DMA	Channel 1 DMA request/Flexcomm transmit request
18	DMIC_CH2_DMA/ FLEXCOMM9_RX_DMA	Channel 2 DMA request/Flexcomm receive request
19	DMIC_CH3_DMA/ FLEXCOMM9_TX_DMA	Channel 3 DMA request/Flexcomm transmit request
20	DMIC_CH4_DMA/ FLEXCOMM10_RX_DMA	Channel 4 DMA request/Flexcomm receive request
21	DMIC_CH5_DMA/ FLEXCOMM10_TX_DMA	Channel 5 DMA request/Flexcomm transmit request
22	DMIC_CH6_DMA/ FLEXCOMM13_RX_DMA	Channel 6 DMA request/Flexcomm receive request/FlexSPI0 receive request
23	DMIC_CH7_DMA/ FLEXCOMM13_TX_DMA	Channel 7 DMA request/Flexcomm transmit request/FlexSPI0 transmit request
24	I3C0_RX_DMA	DMA receive request

Table continues on the next page...

Table 8. DMA0 and DMA1 request assignments (continued)

Slot Number	Alias	Source Description
25	I3C0_TX_DMA	DMA transmit request
26	FLEXCOM14_RX_DMA	DMA receive request
27	FLEXCOM14_TX_DMA	DMA transmit request
28	FLEXCOM16_RX_DMA	FLEXCOMM16 Receive
29	FLEXCOM16_TX_DMA	FLEXCOMM16 Transmit
30	I3C1_RX_DMA	I3C1 Receive
31	I3C1_TX_DMA	I3C1 Transmit
32	FLEXCOMM11_RX	FLEXCOMM11 Receive DMA request
33	FLEXCOMM11_TX	FLEXCOMM11 Transmit DMA request
34	FLEXCOMM12_RX	FLEXCOMM12 Receive DMA request
35	FLEXCOMM12_TX	FLEXCOMM12 Transmit DMA request
SEC Modules		
36	HASHCRYPT_IN_DMA	Hashcrypt DMA request

3.4 DMA trigger input multiplexing

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines. The trigger input multiplexing for each DMA controller is configured as shown in [Figure 5](#)

In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. See [Table 10](#) for DMA0 and DMA1 output trigger multiplexers.

With the DMA trigger input mux registers (DMAC_ITRIG_SEL), one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected. [Table 9](#) shows the peripheral connections for DMA0 and DMA1 trigger inputs.

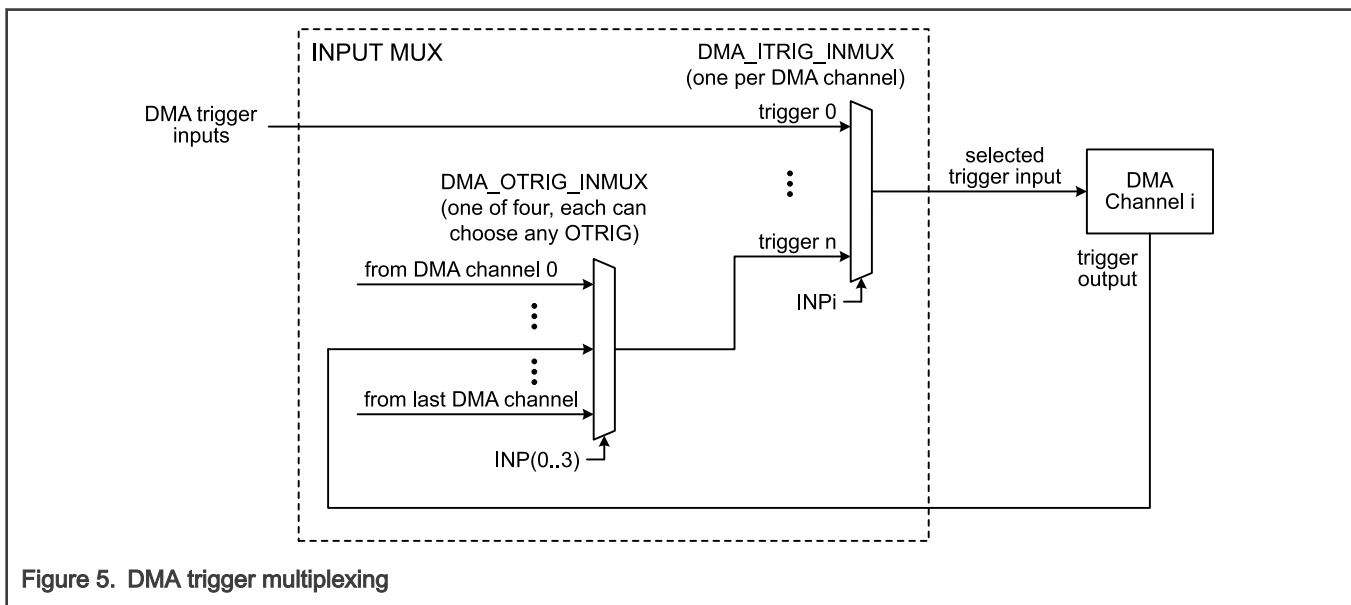


Figure 5. DMA trigger multiplexing

3.5 DMA input trigger assignments

Table 9. DMA and DMA1 input trigger multiplexing assignments

Slot Number	Alias	Source Description
SYS Modules		
0	GPIO_INT0	General Purpose Input-Output Interrupt 0
1	GPIO_INT1	General Purpose Input-Output Interrupt 1
2	GPIO_INT2	General Purpose Input-Output Interrupt 2
3	GPIO_INT3	General Purpose Input-Output Interrupt 3
DMA Modules		
4	CTIMER0_DMAREQ_M0	CTIMER0_DMAREQ_M0
5	CTIMER0_DMAREQ_M1	CTIMER0_DMAREQ_M1
6	CTIMER1_DMAREQ_M0	CTIMER1_DMAREQ_M0
7	CTIMER1_DMAREQ_M1	CTIMER1_DMAREQ_M1
8	CTIMER2_DMAREQ_M0	CTIMER2_DMAREQ_M0
9	CTIMER2_DMAREQ_M1	CTIMER2_DMAREQ_M1
10	CTIMER3_DMAREQ_M0	CTIMER3_DMAREQ_M0
11	CTIMER3_DMAREQ_M1	CTIMER3_DMAREQ_M1
12	CTIMER4_DMAREQ_M0	CTIMER4_DMAREQ_M0
13	CTIMER4_DMAREQ_M1	CTIMER4_DMAREQ_M1
14	DMA _n _TRIGOUT_A	DMA Trigger output A
15	DMA _n _TRIGOUT_B	DMA Trigger output B
16	DMA _n _TRIGOUT_C	DMA Trigger output C
17	DMA _n _TRIGOUT_D	DMA Trigger output C
TMR Modules		
18	SCT_DMA_REQ0	SCTimer DMA request 0
19	SCT_DMA_REQ1	SCTimer DMA request 1
SEC Modules		
20	HASHCRYPT_OUT_DMA	HASHCRYPT DMA output
ANA Modules		
21	ACMP_DMA	Analog Comparator DMA
COM Modules		
22	FlexSPI0_RX_DMA	FlexSPI0 (Quad/Octal) receive DMA, also requires DMA request fixed on channel 22
23	FlexSPI0_TX_DMA	FlexSPI0 (Quad/Octal) transmit DMA, also requires DMA request fixed on channel 23

Table continues on the next page...

Table 9. DMA0 and DMA1 input trigger multiplexing assignments (continued)

Slot Number	Alias	Source Description
ANA Modules		
24	ADC_DMA	Analog to Digital Converter DMA
COM Modules		
25	FlexSPI1_RX_DMA	FlexSPI1 (Quad/Octal) receive DMA, also requires DMA request fixed on channel 25
26	FlexSPI1_TX_DMA	FlexSPI1 (Quad/Octal) transmit DMA, also requires DMA request fixed on channel 26

3.6 DMA output triggers

Table 10. DMA0 and DMA1 output trigger assignments

Slot Number	Alias	Source Description
0	DMA_CH0_TRIGOUT	DMA channel 0 output trigger
1	DMA_CH1_TRIGOUT	DMA channel 1 output trigger
2	DMA_CH2_TRIGOUT	DMA channel 2 output trigger
3	DMA_CH3_TRIGOUT	DMA channel 3 output trigger
4	DMA_CH4_TRIGOUT	DMA channel 4 output trigger
5	DMA_CH5_TRIGOUT	DMA channel 5 output trigger
6	DMA_CH6_TRIGOUT	DMA channel 6 output trigger
7	DMA_CH7_TRIGOUT	DMA channel 7 output trigger
8	DMA_CH8_TRIGOUT	DMA channel 8 output trigger
9	DMA_CH9_TRIGOUT	DMA channel 9 output trigger
10	DMA_CH10_TRIGOUT	DMA channel 10 output trigger
11	DMA_CH11_TRIGOUT	DMA channel 11 output trigger
12	DMA_CH12_TRIGOUT	DMA channel 12 output trigger
13	DMA_CH13_TRIGOUT	DMA channel 13 output trigger
14	DMA_CH14_TRIGOUT	DMA channel 14 output trigger
15	DMA_CH15_TRIGOUT	DMA channel 15 output trigger
16	DMA_CH16_TRIGOUT	DMA channel 16 output trigger
17	DMA_CH17_TRIGOUT	DMA channel 17 output trigger
18	DMA_CH18_TRIGOUT	DMA channel 18 output trigger
19	DMA_CH19_TRIGOUT	DMA channel 19 output trigger
20	DMA_CH20_TRIGOUT	DMA channel 20 output trigger
21	DMA_CH21_TRIGOUT	DMA channel 21 output trigger

Table continues on the next page...

Table 10. DMA0 and DMA1 output trigger assignments (continued)

Slot Number	Alias	Source Description
22	DMA_CH22_TRIGOUT	DMA channel 22 output trigger
23	DMA_CH23_TRIGOUT	DMA channel 23 output trigger
24	DMA_CH24_TRIGOUT	DMA channel 24 output trigger
25	DMA_CH25_TRIGOUT	DMA channel 25 output trigger
26	DMA_CH26_TRIGOUT	DMA channel 26 output trigger
27	DMA_CH27_TRIGOUT	DMA channel 27 output trigger
28	DMA_CH28_TRIGOUT	DMA channel 28 output trigger
29	DMA_CH29_TRIGOUT	DMA channel 29 output trigger
30	DMA_CH30_TRIGOUT	DMA channel 30 output trigger
31	DMA_CH31_TRIGOUT	DMA channel 31 output trigger
32	DMA_CH32_TRIGOUT	DMA channel 32 output trigger
33	DMA_CH33_TRIGOUT	DMA channel 33 output trigger
34	DMA_CH34_TRIGOUT	DMA channel 34 output trigger
35	DMA_CH35_TRIGOUT	DMA channel 35 output trigger
36	DMA_CH36_TRIGOUT	DMA channel 36 output trigger

3.7 Fusion DSP interrupt configuration

Interrupts to the Fusion DSP CPU are multiplexed in order to allow more control over priorities and more general flexibility. The sources shown in the table below are the inputs to a "periphinmux" with 41 inputs and 27 outputs. These inputs can be selected by the Fusion DSP Interrupt Select (DSP_INT_SEL) register. Any of these inputs can be selected to be routed to any one of the 27 mux outputs. Those 27 outputs, in turn, connect to the Fusion DSP IRQ slots 5-31.

The connection of interrupts to the Fusion DSP is shown in [Table 12](#). Most interrupts are user selectable using the input multiplexing feature described in [Table 11](#).

The figure below shows how the interrupts are connected and how they can be configured.

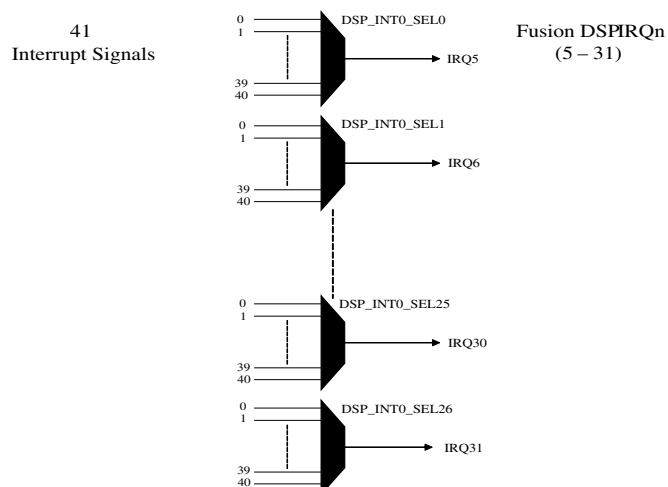


Figure 6. DSP Interrupt Configuration

In addition to giving a choice of interrupts needed, selecting interrupts allows allocating interrupt priorities to fit the application. L1 interrupts are the lowest priority, L3 are the highest. One of the Fusion DSP interrupt sources (from interrupt 5 through interrupt 31) can be selected as the Fusion DSP NMI source by configuring the Fusion DSP NMI source selection.

More details will be provided in the Input Multiplexing chapter.

3.8 Fusion DSP interrupt input multiplexer

Table 11. Fusion DSP Interrupt input multiplexers (selected by DSP_INT_SELn)

Address	IRQ	Alias	Source Description
0000_0040	0	FLEXCOMM0_IRQ	FlexComm interrupt
0000_0044	1	FLEXCOMM1_IRQ	FlexComm interrupt
0000_0048	2	FLEXCOMM2_IRQ	FlexComm interrupt
0000_004C	3	FLEXCOMM3_IRQ	FlexComm interrupt
0000_0050	4	FLEXCOMM4_IRQ	FlexComm interrupt
0000_0054	5	FLEXCOMM5_IRQ	FlexComm interrupt
0000_0058	6	FLEXCOMM6_IRQ	FlexComm interrupt
0000_005C	7	FLEXCOMM7_IRQ	FlexComm interrupt
0000_0060	8	FLEXCOMM14_IRQ	FlexComm interrupt
0000_0064	9	FLEXCOMM16_IRQ	FlexComm interrupt
0000_0068	10	GPIO_INT0_IRQ0	General Purpose Input/Output interrupt 0
0000_006C	11	GPIO_INT0_IRQ1	General Purpose Input/Output interrupt 1
0000_0070	12	GPIO_INT0_IRQ2	General Purpose Input/Output interrupt 2
0000_0074	13	GPIO_INT0_IRQ3	General Purpose Input/Output interrupt 3
0000_0078	14	GPIO_INT0_IRQ4	General Purpose Input/Output interrupt 4

Table continues on the next page...

Table 11. Fusion DSP Interrupt input multiplexers (selected by DSP_INT_SELn) (continued)

Address	IRQ	Alias	Source Description
0000_007C	15	GPIO_INT0_IRQ5	General Purpose Input/Output interrupt 5
0000_0080	16	GPIO_INT0_IRQ6	General Purpose Input/Output interrupt 6
0000_0084	17	GPIO_INT0_IRQ7	General Purpose Input/Output interrupt 7
0000_0088	18	GPIO_INTA	Non-Secure GPIO HS interrupt 0
0000_008C	19	GPIO_INTB	Non-Secure GPIO HS interrupt 1
0000_0090	20	WDT1_IRQ	Watchdog Timer interrupt
0000_0094	21	DMA0_IRQ	DMA interrupt
0000_0098	22	DMA1_IRQ	DMA interrupt
0000_009C	23	MU_B_IRQ	Messaging Unit interrupt
0000_00A0	24	UTICK0_IRQ	Microtick interrupt
0000_00A4	25	MRT0_IRQ	Multi-Rate Timer
0000_00A8	26	OS_EVENT_TIMER_DSP_IRQ	Event Timer DSP Wakeup/interrupt
0000_00AC	27	CTIMER0_IRQ	CTimer interrupt
0000_00B0	28	CTIMER1_IRQ	CTimer interrupt
0000_00B4	29	CTIMER2_IRQ	CTimer interrupt
0000_00B8	30	CTIMER3_IRQ	CTimer interrupt
0000_00BC	31	CTIMER4_IRQ	CTimer interrupt
0000_00C0	32	RTC_LITE0_ALARM_IRQ OR RTC_LITE0_WAKEUP_IRQ	RTC Alarm/wakeup interrupt
0000_00C4	33	I3C0_IRQ	Improved Inter Integrated Circuit interrupt
0000_00C8	34	I3C1_IRQ	Improved Inter Integrated Circuit interrupt
0000_00CC	35	D_MIC0	Digital Microphone interrupt
0000_00D0	36	HWVAD	Harware Voice Activity Detector interrupt
0000_00D4	37	LCDIF_IRQ	Liquid Crystal Display Interface interrupt
0000_00D8	38	GPU_IRQ	Graphics Processor Unit interrupt
0000_00DC	39	SmartDMA_IRQ	Smart DMA Controller interrupt
0000_00E0	40	FLEXIO_IRQ	Flexible I/O interrupt

3.9 Fusion DSP interrupt priority assignments

Table 12. Fusion DSP assignments

Address	IRQ	Alias	Priority level
0000_0040	0	SYS_IRQ	NMI

Table continues on the next page...

Table 12. Fusion DSP assignments (continued)

Address	IRQ	Alias	Priority level
0000_0044	1	SOFTWARE IRQ0	L2
0000_0048	2	INTERNAL RTOS TIMER0	L2
0000_004C	3	INTERNAL RTOS TIMER1	L3
0000_0050	4	PROFILING IRQ	L3
0000_0054	5	Interrupt selected by DSP_INT0_SEL0	L1 (L1 is lowest priority)
0000_0058	6	Interrupt selected by DSP_INT0_SEL1	L1
0000_005C	7	Interrupt selected by DSP_INT0_SEL2	L1
0000_0060	8	Interrupt selected by DSP_INT0_SEL3	L1
0000_0064	9	Interrupt selected by DSP_INT0_SEL4	L1
0000_0068	10	Interrupt selected by DSP_INT0_SEL5	L1
0000_006C	11	Interrupt selected by DSP_INT0_SEL6	L1
0000_0070	12	Interrupt selected by DSP_INT0_SEL7	L1
0000_0074	13	Interrupt selected by DSP_INT0_SEL8	L1
0000_0078	14	Interrupt selected by DSP_INT0_SEL9	L1
0000_007C	15	Interrupt selected by DSP_INT0_SEL10	L1
0000_0080	16	Interrupt selected by DSP_INT0_SEL11	L2
0000_0084	17	Interrupt selected by DSP_INT0_SEL12	L2
0000_0088	18	Interrupt selected by DSP_INT0_SEL13	L2
0000_008C	19	Interrupt selected by DSP_INT0_SEL14	L2
0000_0090	20	Interrupt selected by DSP_INT0_SEL15	L2
0000_0094	21	Interrupt selected by DSP_INT0_SEL16	L2
0000_0098	22	Interrupt selected by DSP_INT0_SEL17	L2
0000_009C	23	Interrupt selected by DSP_INT0_SEL18	L2
0000_00A0	24	Interrupt selected by DSP_INT0_SEL19	L3
0000_00A4	25	Interrupt selected by DSP_INT0_SEL20	L3
0000_00A8	26	Interrupt selected by DSP_INT0_SEL21	L3
0000_00AC	27	Interrupt selected by DSP_INT0_SEL22	L3
0000_00B0	28	Interrupt selected by DSP_INT0_SEL23	L3
0000_00B4	29	Interrupt selected by DSP_INT0_SEL24	L3
0000_00B8	30	Interrupt selected by DSP_INT0_SEL25	L3
0000_00BC	31	Interrupt selected by DSP_INT0_SEL26	L3

Chapter 4

System Controller (SYSCTL)

4.1 Chip-specific SYSCTL information

Table 13. Reference links to related information

Topic	Related module	Reference
Full description	SYSCTL	SYSCTL
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

4.1.1 Module instances

This device has two instances of the SYSCTL module, SYSCTL0 and SYSCTL1.

4.2 Overview

This section provides an introduction to the SYSCTL module.

4.3 Signals

This table describes the module signals.

Signal	I/O	Description
CLKIN	I	External Clock Input
CLKOUT	O	External Clock Output

4.4 Functional description

The following sections describe functional details of the SYSCTL module.

4.4.1 I2S Bridging and Signal Sharing

Bridging allows control of pins connected to selected I2S functions and external devices to be switched between the chip and an external master.

Signal sharing allows more than one on-chip I2S interface to be connected to clock, WS, and input data on the same pins without external board wiring. Multiple I2S functions contributing output data to a single data line must still be accomplished with an external connection.

4.4.1.1 I2S Signal Sharing

It is sometimes desirable to use multiple I2S functions together in a single TDM stream without sacrificing more pins than are needed. I2S signal sharing allows this kind of use without the need for external connections to multiple pins outside of the device. This is only needed when the requirements exceed what can be accomplished with a single I2S interface that includes four channel pairs.

In general, each Flexcomm Interface configured for I2S can choose:

- Its own SCK, or a shared SCK.
- Its own WS, or a shared WS.
- Its own DATA in, or a shared DATA in.

Each Flexcomm Interface potentially contributes to shared signals:

- Its own SCK (in or out, depending on whether it is a master or slave)
- Its own WS (in or out, depending on whether it is a master or slave)
- Its own DATA input

4.4.1.1.1 I2S Signal Sharing Configuration

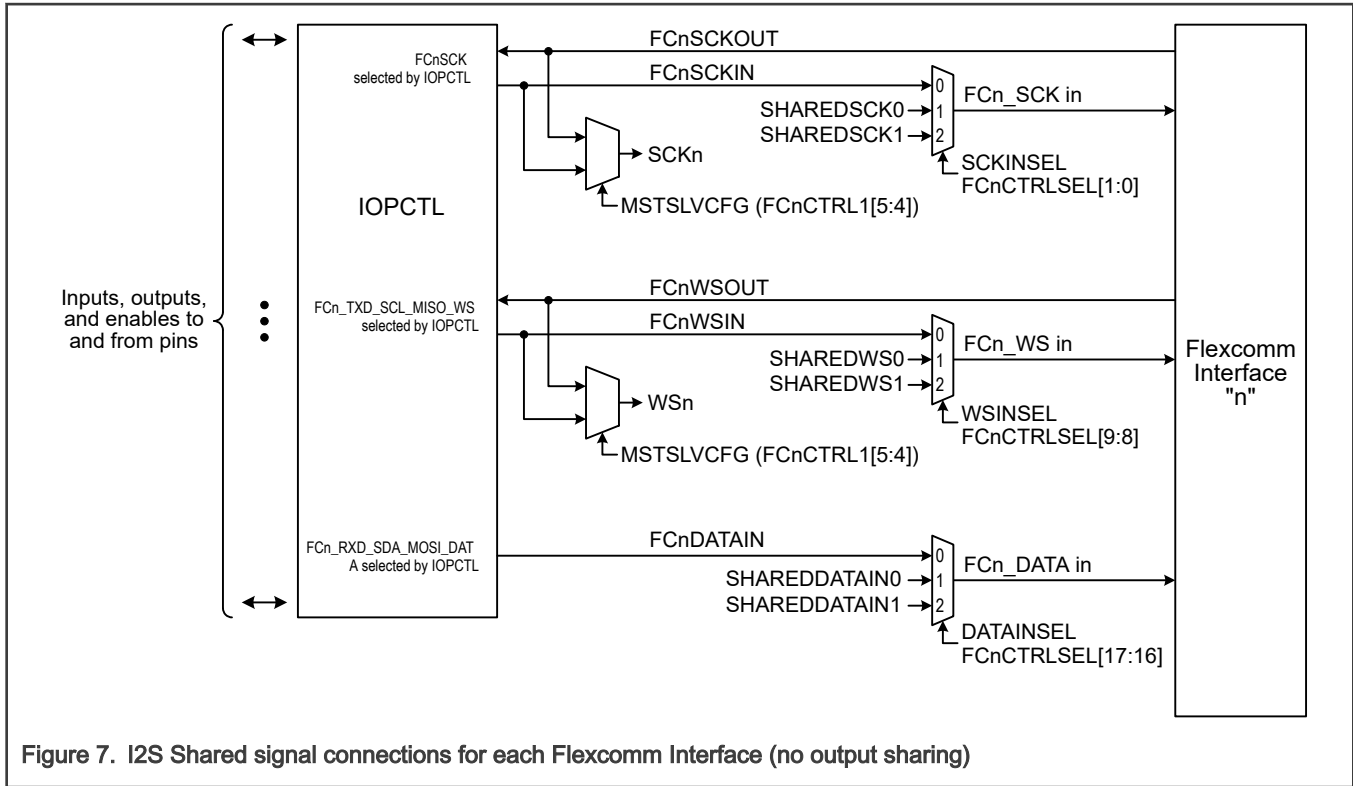
I2S signal sharing can be configured as follows:

1. Select the appropriate functions in IOPCTL for the pins that will actually be connected externally for I2S operation.
2. Set up shared signal sets that will be used by writing to the SHAREDCTRLSET0 and/or SHAREDCTRLSET1 registers (see SYSCTL).
3. Set up signal sharing for each Flexcomm Interface that uses shared signals by writing to FCnCTRLSEL registers as needed (see SYSCTL).
4. Set up Flexcomm Interfaces using I2S signal sharing as needed. If any Flexcomm Interface is acting as a master, connect it first. Then connect Flexcomm Interfaces acting as slaves.

NOTE

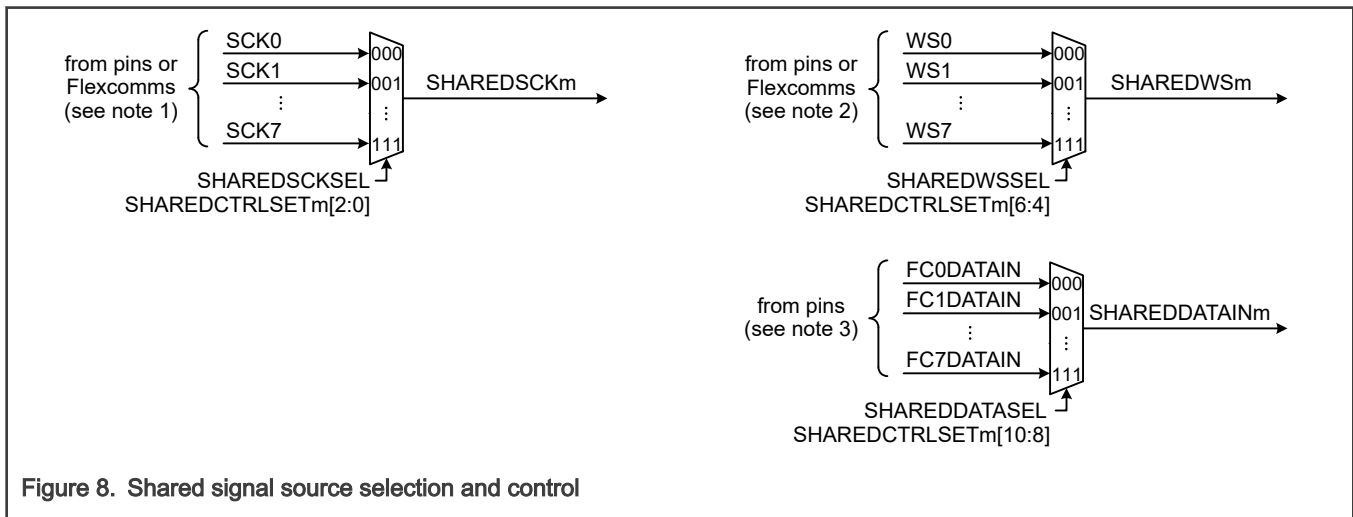
Signal sharing connections are made as register values are changed, without synchronization, and so should be done prior to the start of data streams. Also, any I2S master that is providing SCK and WS signals for shared usage should also be configured to use the shared signal. For example, if Flexcomm Interface 0 is providing SCK and WS to shared set 0, FC0CTRLSEL should select shared set 0 for SCK and WS.

The figures below show an example of representative logic for I2S signal sharing.



NOTE

These connection options are replicated for each Flexcomm Interface with I2S support.



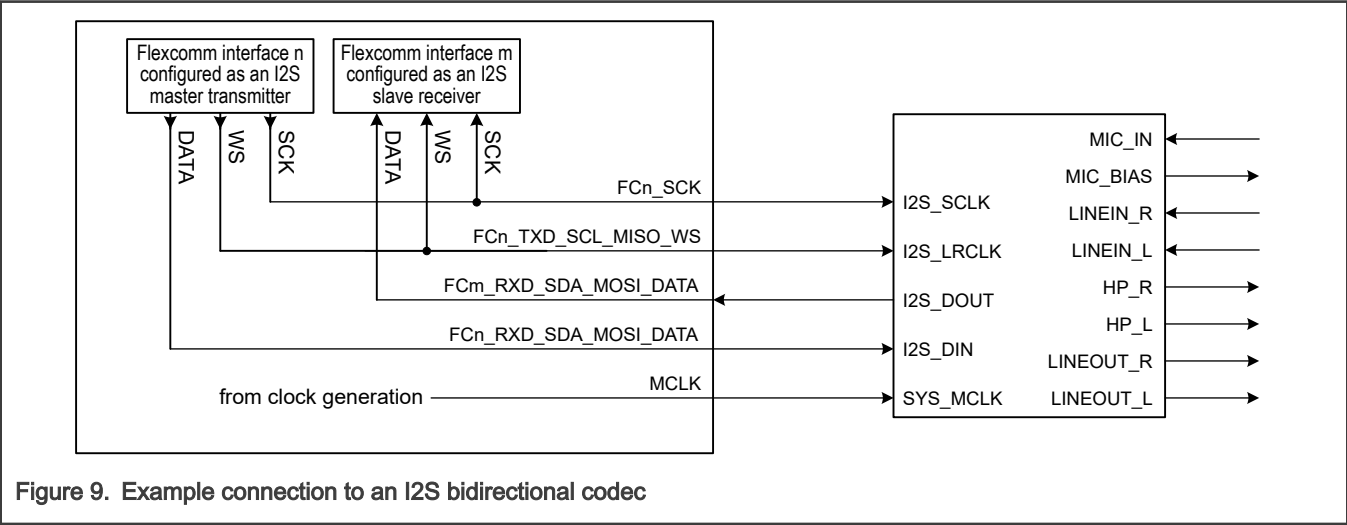
NOTE

This logic is replicated for each of the two sets of shared signals ("m", 0 through 1).

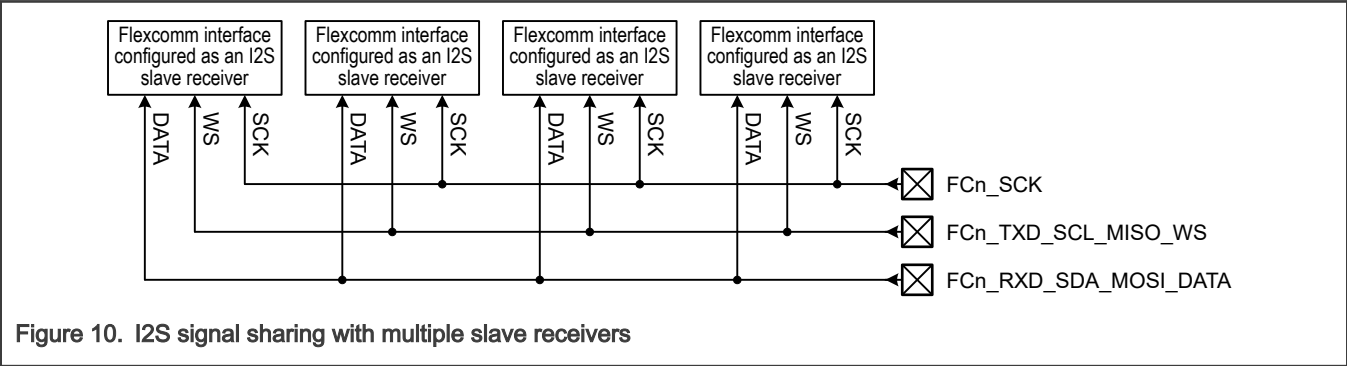
- Each SCK or WS comes either from IOPCTL (FCn_SCK) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.
- Each WS comes either from IOPCTL (FCn_TXD_SCL_MISO_WS) or from the related Flexcomm Interface if the Flexcomm is configured as an I2S master.
- Each FCnDATAIN comes from IOPCTL (FCn_RXD_SDA_MOSI_DATA).

4.4.1.1.2 I2S Signal Sharing Examples

The figure below shows an example of a simple bidirectional codec with input and output data connected to two different I2S interfaces, using signal sharing to reduce connections to a single SCK and single WS pin. In this case, one I2S interface is a master transmitter and one is a slave receiver. Data input and output cannot be shared on one pin because they are separate pins on the external codec.



The following figure shows a generic case of multiple slaves and/or receivers sharing SCK and WS, and/or DATA. It includes received data sharing with different I2S interfaces receiving data from different slots in a TDM stream. All I2S interfaces are slave receivers sharing SCK, WS, and DATA.



The following figure shows master to slave operation where one I2S interface is a master going off chip, and other on-chip I2S interfaces are slaved to it. Data could be either transmitted or received. Multiple I2S interfaces supply data to a single stream by wiring multiple pins together. All I2S interfaces are transmitters sharing SCK and WS.

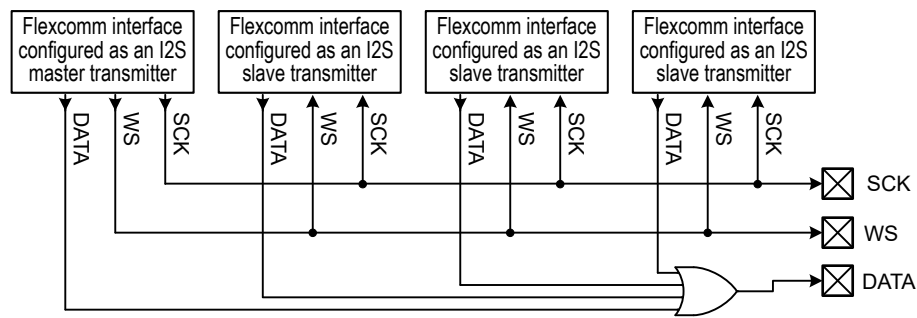


Figure 11. I2S signal sharing example showing one master and multiple slave transmitters

The following figure shows data sharing with one I2S interface transmitting onto a shared DATA line and at least one other I2S interface is receiving from the same DATA line. All I2S interfaces share SCK and WS. One I2S interface is the master, others are slaves. Two I2S interfaces are transmitters, others are receivers. Data is output from two I2S interfaces and input to two I2S interfaces. Which I2S interfaces are transmitters and receivers is arbitrary in this example. This does not necessarily mean that the transmitted data is what is being received, they could be different packets in a TDM frame. The example shows two I2S interfaces transmitting and two receiving, but it could be any combination.

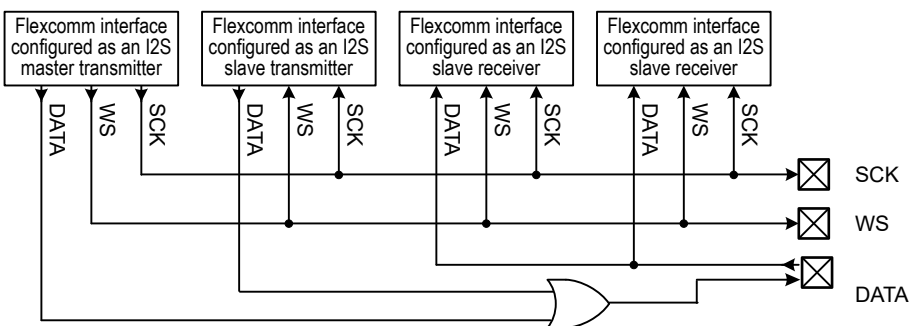
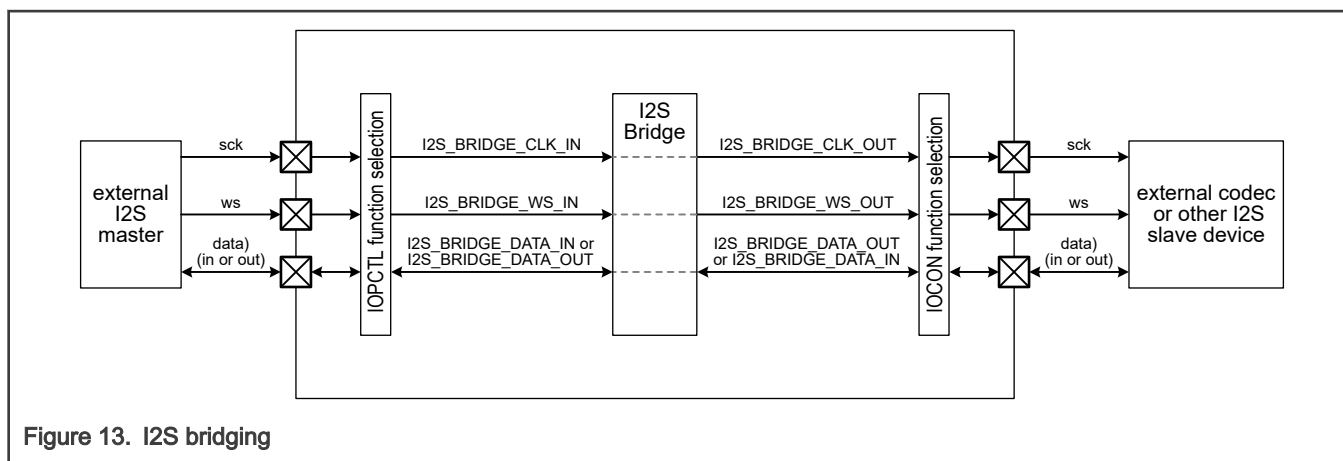


Figure 12. I2S signal sharing example showing one master with mixed transmitters and receivers

4.4.1.2 I2S Bridging

I2S Bridging allows control of pins connected to selected I2S functions and external devices to be switched between the chip and an external master. In a higher power situation, an external master controls a codec in which some other application needs are also serviced by the other master. When application needs are lower, the other master powers off and hands control of the I2S to the chip.

The I2S bridging feature is implemented essentially as a peripheral consisting of 3 wires. It can be connected in lieu of normal I2S functions via IOPCTL function selection. Any stream in progress needs to be stopped while switching to and from bridging mode. There is no synchronization, connections are simply switched in as they are changed in IOPCTL.



4.5 Memory map and register definition

This section includes the SYSCTL module memory map and detailed descriptions of all registers.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

4.5.1 SYSCTL0 register descriptions

4.5.1.1 SYSCTL0 memory map

SYSCTL0 base address: 4000_2000h

Offset	Register	Width (In bits)	Access	Reset value
4h	DSP Vector Remap (DSP_VECT_REMAP)	32	RW	0000_0000h
Ch	DSP Stall Control (DSPSTALL)	32	RW	0000_0001h
10h	AHB MAX Priority (AHBMATRIXPRIOR)	32	RW	0000_0000h
20h - 24h	AHB Buffer (AHBBRIDGEBUFFER0 - AHBBRIDGEBUFFER1)	32	RW	0000_0000h
2Ch	BOOT ROM lockout (BOOTROM_LCKOUT)	32	RW	0000_0000h
30h	M33 NMI source selection (M33NMISRCSEL)	32	RW	0000_0000h
34h	System secure tick calibration (SYSTEM_STICK_CALIB)	32	RW	0000_0000h
38h	System non-secure tick calibration (SYSTEM_NSTICK_CALIB)	32	RW	0000_0000h
60h	Product ID (PRODUCT_ID)	32	RW	0000_CAFeh
64h	Silicon Revision ID (SILICONREV_ID)	32	R	000B_000Ah
68h	JTAG ID (JTAG_ID)	32	R	See section
70h	Non-secure GPIO PSYNC (NSGPIO_PSYNC)	32	RW	0000_0000h
74h	Secure GPIO PSYNC (SGPIO_PSYNC)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
80h	Auto clock gate override 0 (AUTOCLKGATEOVERRIDE0)	32	RW	FFFF_FFFFh
84h	Auto clock gate override 1 (AUTOCLKGATEOVERRIDE1)	32	RW	FFFF_FFFFh
A0h	Clock gate override 0 (CLKGATEOVERRIDE0)	32	RW	FFFF_FFFFh
FCh	AHB SRAM access disable (AHB_SRAM_ACCESS_DISABLE)	32	RW	0000_0000h
100h	AXI SRAM access disable (AXI_SRAM_ACCESS_DISABLE)	32	RW	0000_0000h
104h	DSP SRAM access disable (DSP_SRAM_ACCESS_DISABLE)	32	RW	0000_0000h
110h	Power-Quad Memory Control (PQ_MEM_CTRL)	32	RW	0000_0000h
114h	FlexSPI0 Memory Control (FLEXSPI0_MEM_CTRL)	32	RW	0000_0000h
118h	USBHS Memory Control (USBHS_MEM_CTRL)	32	RW	0000_0000h
11Ch	USDHC0 Memory Control (USDHC0_MEM_CTRL)	32	RW	0000_0000h
120h	USDHC1 Memory Control (USDHC1_MEM_CTRL)	32	RW	0000_0000h
124h	CASPER Memory Control (CASPER_MEM_CTRL)	32	RW	0000_0003h
128h	ROM Memory Control (ROM_MEM_CTRL)	32	RW	0000_0000h
12Ch	FlexSPI1 Memory Control (FlexSPI1_MEM_CTRL)	32	RW	0000_0000h
130h	GPU Memory Control (GPU_MEM_CTRL)	32	RW	0000_0000h
134h	MIPI Memory Control (MIPI_MEM_CTRL)	32	RW	0000_0000h
138h	LCDIF Memory Control (DCN_MEM_CTRL)	32	RW	0000_0000h
13Ch	SMARTDMA Memory Control (SMARTDMA_MEM_CTRL)	32	RW	0000_0000h
400h	MIPI DSI Control (MIPI_DSI_CTRL)	32	RW	0000_0000h
40Ch	USB Clock Control (USB0CLKCTRL)	32	RW	0000_0010h
410h	USB Clock Status (USB0CLKSTAT)	32	R	0000_0003h
414h	USB PHY PLL0 lock time division (USBPHYPLL0LOCKTIMEDIV2)	32	RW	0000_CAFeh
600h	Sleep configuration 0 (PDSLEEPCFG0)	32	RW	03FE_AC80h
604h	Sleep configuration 1 (PDSLEEPCFG1)	32	RW	067F_FFFFh
608h	Sleep configuration 2 (PDSLEEPCFG2)	32	RW	FFFF_FFFEh
60Ch	Sleep configuration 3 (PDSLEEPCFG3)	32	RW	FFFF_FFFEh
610h	Run configuration 0 (PDRUNCFG0)	32	RW	03FE_AC80h
614h	Run configuration 1 (PDRUNCFG1)	32	RW	067F_FFFFh
618h	Run configuration 2 (PDRUNCFG2)	32	RW	FFFF_FFFEh
61Ch	Run configuration 3 (PDRUNCFG3)	32	RW	FFFF_FFFEh
620h	Run configuration 0 set (PDRUNCFG0_SET)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
624h	Run configuration 1 set (PDRUNCFG1_SET)	32	RW	See section
628h	Run configuration 2 set (PDRUNCFG2_SET)	32	W	See section
62Ch	Run configuration 3 set (PDRUNCFG3_SET)	32	W	See section
630h	Run configuration 0 clear (PDRUNCFG0_CLR)	32	RW	See section
634h	Run configuration 1 clear (PDRUNCFG1_CLR)	32	RW	See section
638h	Run configuration 2 clear (PDRUNCFG2_CLR)	32	RW	See section
63Ch	Run configuration 3 clear (PDRUNCFG3_CLR)	32	RW	See section
660h	PD Wake Configuration (PDWAKECFG)	32	RW	0000_0000h
680h	Start Enable 0 (STARTEN0)	32	RW	0000_0000h
684h	Start Enable 1 (STARTEN1)	32	RW	0000_0000h
688h	Start Enable 2 (STARTEN2)	32	RW	0000_0000h
6A0h	Start Enable 0 Set (STARTEN0_SET)	32	RW	See section
6A4h	Start Enable 1 Set (STARTEN1_SET)	32	RW	See section
6A8h	Start Enable 2 (STARTEN2_SET)	32	RW	0000_0000h
6C0h	Start Enable 0 clear (STARTEN0_CLR)	32	RW	See section
6C4h	Start Enable 1 clear (STARTEN1_CLR)	32	RW	See section
6C8h	Start Enable 2 (STARTEN2_CLR)	32	RW	0000_0000h
704h	FRO Safety (FROSAFETY)	32	RW	0000_0000h
710h	Main Clock Safety (MAINCLKSAFETY)	32	RW	0000_0000h
780h	Hardware Wake (HWWAKE)	32	RW	0000_0000h
E0Ch	Temperature Sensor Control (TEMPSENSORCTL)	32	RW	0000_0000h
E40h	Boot State Lock (BOOTSTATELOCK)	32	RW	0000_0000h
E50h - E6Ch	Boot State Seed (BOOTSTATESEED0 - BOOTSTATESEED7)	32	RW	0000_0000h
E70h - E8Ch	HMAC of boot state used for attestation. (BOOTSTATEHMAC0 - BOOTSTATEHMAC7)	32	RW	0000_0000h
EF0h	FLEXSPI0 Pad Control (FLEXSPI0PADCTL)	32	RW	01A5_0000h
EF4h	FLEXSPI1 Pad Control (FLEXSPI1PADCTL)	32	RW	01A5_0000h
EF8h	SDIO0 Pad Control (SDIO0PADCTL)	32	RW	01A5_0000h
EFCh	SDIO1 Pad Control (SDIO1PADCTL)	32	RW	01A5_0000h
F00h	Compound Device Identifier (CDI) (DICEHWREGn)	32	RW	0000_0000h
F50h - F5Ch	UUID (UUID0 - UUID3)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
F80h	AES Key Source Select (AESKEY_SRCSEL)	32	RW	0000_0000h
F84h	OTFAD Key Source Select (OTFADKEY_SRCSEL)	32	RW	0000_0000h
F88h	HASH Hardware Key Disable (HASHHWKEYDISABLE)	32	RW	0000_0000h
FA0h	Debug Lock Enable (DBG_LOCKEN)	32	RW	0000_000Ah
FA4h	Debug Features (DBG_FEATURES)	32	RW	0000_0055h
FA8h	Debug Features Duplicate (DBG_FEATURES_DP)	32	RW	0000_0055h
FB4h	Code Security for CPU0 (CS_PROTCPU0)	32	RW	0000_0000h
FB8h	Code Security for CPU1 (CS_PROTCPU1)	32	RW	0000_0000h
FC0h	Debug authorization scratch (DBG_AUTH_SCRATCH)	32	RW	0000_0000h
FD0h	Key block (KEY_BLOCK)	32	RW	3CC3_5AA5h

4.5.1.2 DSP Vector Remap (DSP_VECT_REMAP)

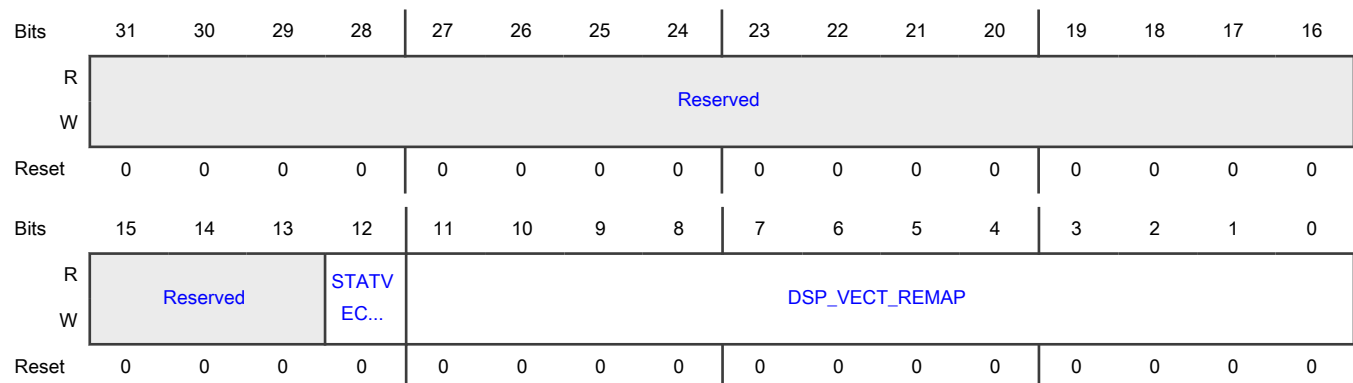
Offset

Register	Offset
DSP_VECT_REMAP	4h

Function

It selects the Fusion DSP vector base address.

Diagram



Fields

Field	Function
31-13 —	Reserved
12 STATVECSELE CT	Static Vector Select 0b - Selects the primary static vector base address on Fusion DSP (0x0000_0000) 1b - Selects the alternate static vector base address on Fusion DSP (0x0040_0000)
11-0 DSP_VECT_RE MAP	DSP_VECT_REMAP Vector Remap

4.5.1.3 DSP Stall Control (DSPSTALL)

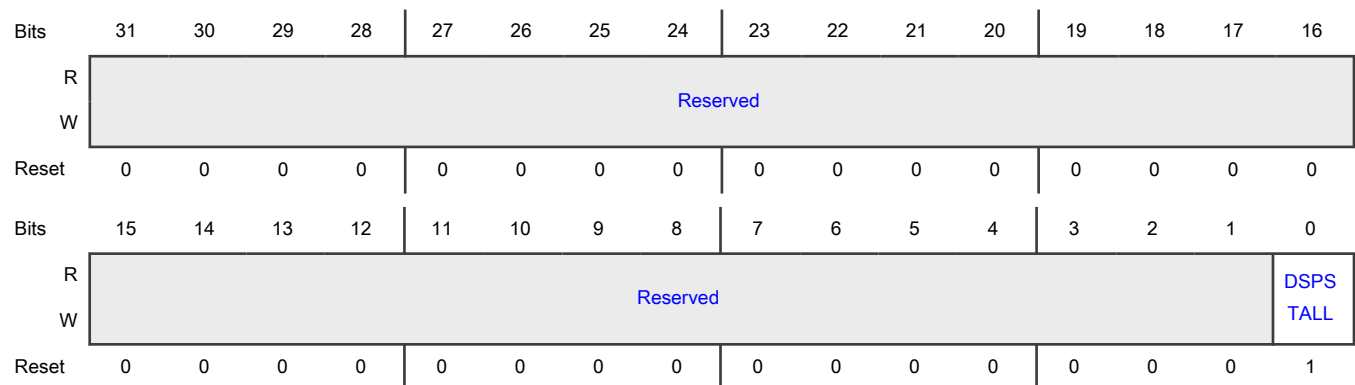
Offset

Register	Offset
DSPSTALL	Ch

Function

It controls the operation of Fusion DSP.

Diagram



Fields

Field	Function
31-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 DSPSTALL	DSPSTALL Run/Stall control 0b - Run(Normal mode) 1b - Stall mode

4.5.1.4 AHB MAX Priority (AHBMATRIXPRIOR)

Offset

Register	Offset
AHBMATRIXPRIOR	10h

Function

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed, improper settings can seriously degrade performance. Priority values are 0 = highest, 3 = lowest. When the priority is the same, the master with the lower master number is given priority. An example setting could put the Cortex-M33 Code bus as the highest priority, followed by the System bus. All other masters could share a lower priority.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								M11		M10		M9		M8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M7		M6		M5		M4		M3		M2		M1		M0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-22 M11	Master 10 Priority GPUD/LCDIF AXI/AHB Bridge0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-20 M10	Master 10 Priority Hash-AES.
19-18 M9	Master 9 Priority SDIO1.
17-16 M8	Master 8 Priority SDIO0.
15-14 M7	Master 7 Priority Smart DMA Data.
13-12 M6	Master 6 Priority Smart DMA Instruction.
11-10 M5	Master 5 Priority DMAC1.
9-8 M4	Master 4 Priority DMAC0.
7-6 M3	Master 3 Priority Fusion DSP.
5-4 M2	Master 2 Priority PowerQuad coprocessor.
3-2 M1	Master 1 Priority Cortex-M33 S-AHB (system) bus.
1-0 M0	Master 0 Priority Cortex-M33 C-AHB (code) bus.

4.5.1.5 AHB Buffer (AHBBRIDGEBUFFER0 - AHBBRIDGEBUFFER1)

Offset

Register	Offset
AHBBRIDGEBUFFER0	20h
AHBBRIDGEBUFFER1	24h

Function

AHB Buffer that bridges the AHB to APB busses. AHBBRIDGEBUFFER0 maps to peripherals on APB0 and AHBBRIDGEBUFFER1 maps to peripherals on APB1. For individual slave peripheral mapping, see APB1 (32 Peripheral Slots) in Memory Maps.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV
W	E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV	SLAV
W	E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SLAVEn	SLAVEn buffering 0b - No Buffering 1b - Buffering

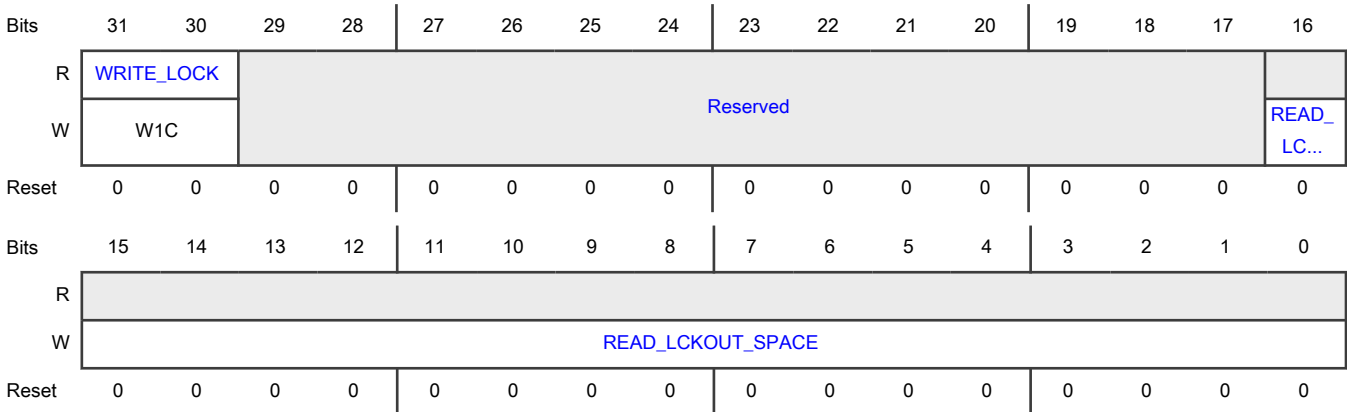
4.5.1.6 BOOT ROM lockout (BOOTROM_LCKOUT)**Offset**

Register	Offset
BOOTROM_LCKOUT	2Ch

Function

Bootcode Read Lockout is a security feature which precludes the read requests of a system defined address space of the ROM memory.

Diagram



Fields

Field	Function
31-30 WRITE_LOCK	Self Write Disable 00b - Write disable 01b - Write disable 10b - Write enable 11b - Write disable
29-17 —	Reserved
16-0 READ_LCKOUT_SPACE	Read Lockout ROM Memory space offset that will become non-accessible for read.

4.5.1.7 M33 NMI source selection (M33NMISRCSEL)

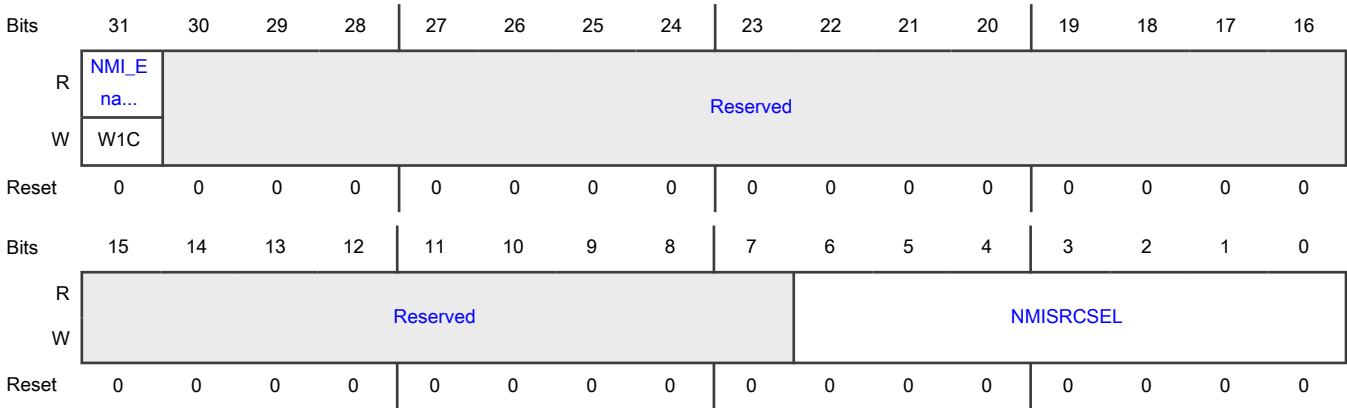
Offset

Register	Offset
M33NMISRCSEL	30h

Function

The NMI source selection register selects peripheral interrupts as source for the NMI interrupt of the Cortex-M33. The NMI source must first be disabled by writing 0 to the NMIEN bit to change the interrupt source for the NMI. The source can then be changed by updating the IRQN bits and re-enable the NMI source by setting NMIEN.

Diagram



Fields

Field	Function
31 NMI_Enable	Self Write Disable 0b - Disable NMI interrupt 1b - Enable NMI interrupt
30-7 —	Reserved
6-0 NMISRCSEL	Selects one of the M33 interrupt sources as the NMI source interrupt. See Arm M33 NVIC assignments for Interrupt Slot Numbers. NOTE NMI sources 79-127 are not usable.

4.5.1.8 System secure tick calibration (SYSTEM_STICK_CALIB)

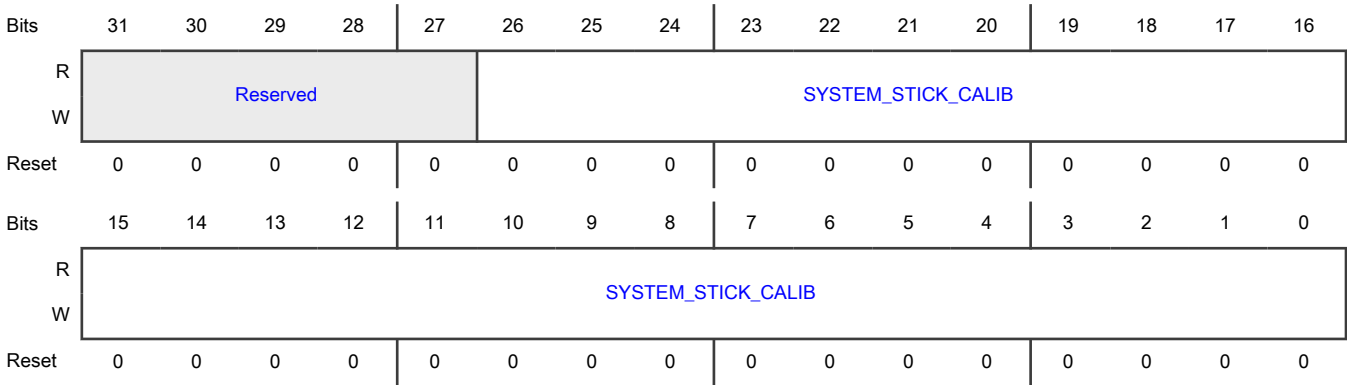
Offset

Register	Offset
SYSTEM_STICK_CALIB	34h

Function

This register allows software to set up a default value for the SYST_CALIB register for the System Tick Timer in the secure part of CPU.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-0 SYSTEM_STICK_CALIB	M33 secure tick calibration Selects the system secure tick calibration value of the M33.

4.5.1.9 System non-secure tick calibration (SYSTEM_NSTICK_CALIB)

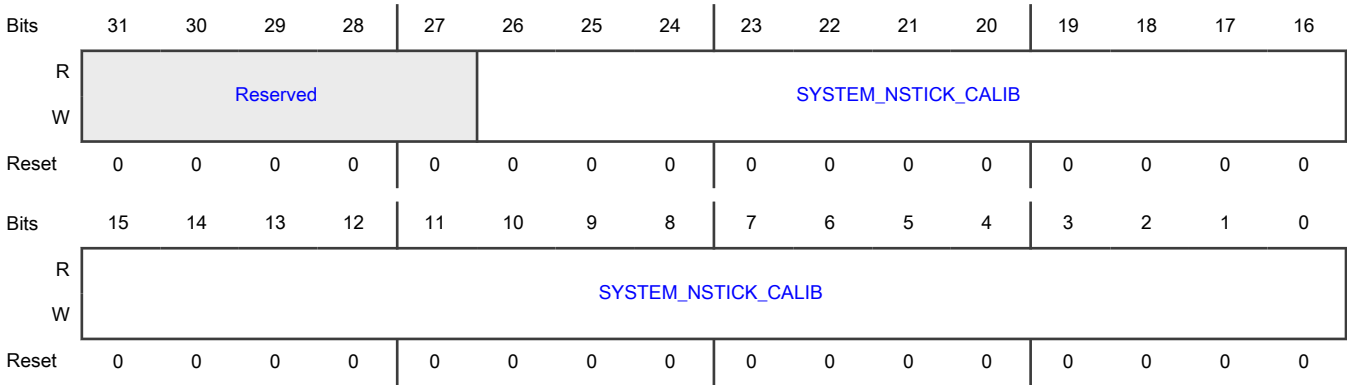
Offset

Register	Offset
SYSTEM_NSTICK_CALIB	38h

Function

This register allows software to set up a default value for the SYST_CALIB register for the System Tick Timer in the non-secure part of CPU.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-0 SYSTEM_NSTICK_CALIB	M33 non secure tick calibration Selects the system secure tick calibration value of the M33.

4.5.1.10 Product ID (PRODUCT_ID)

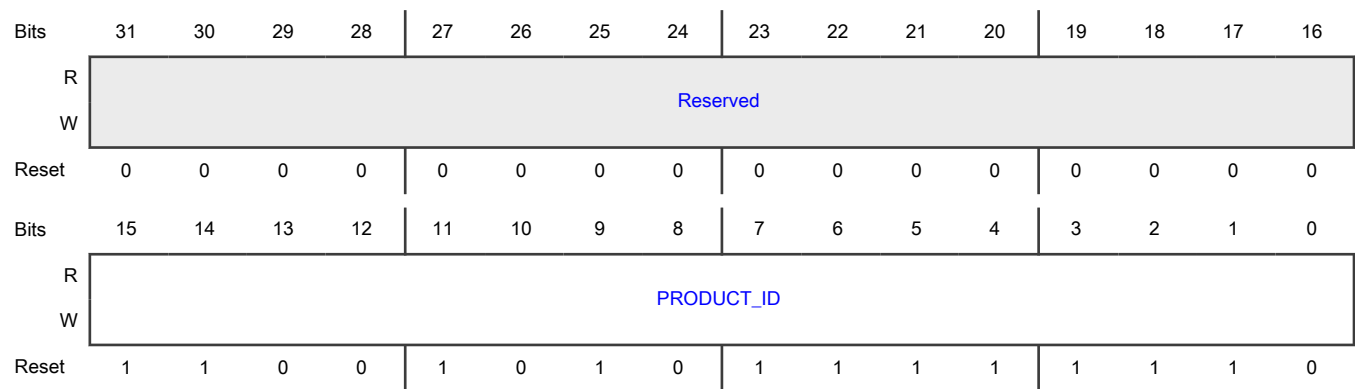
Offset

Register	Offset
PRODUCT_ID	60h

Function

The BOOTROM will copy the PRODUCT_ID programmed value found in the OTP Fuse (if the OTP valid signature is present) into this register on any reset.

The Product ID register is reset to a value of 0x0000 CAFE, however, BOOTROM will load the PRODUCT_ID from OTP.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 PRODUCT_ID	PRODUCT ID Product ID that contributes to the overall JTAG ID.

4.5.1.11 Silicon Revision ID (SILICONREV_ID)**Offset**

Register	Offset
SILICONREV_ID	64h

Function

Revision numbers for this silicon.

It is expected on the first silicon that the minor revision be 0x0. Any subsequent metal mask change/update should result in the minor revision being upgraded to the next level.

It is expected on the first silicon that the major revision be 0xA. Any subsequent full mask change/update should result in the major revision being upgraded to the next level.

Diagram

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

Fields

Field	Function
31-20 —	Reserved
19-16 MAJOR	MAJOR Silicon revision major tag. (IE, A, B, C, etc)
15-4 —	Reserved
3-0 MINOR	MINOR Silicon revision minor tag. (IE, 0, 1, 2, 3, etc)

4.5.1.12 JTAG ID (JTAG_ID)**Offset**

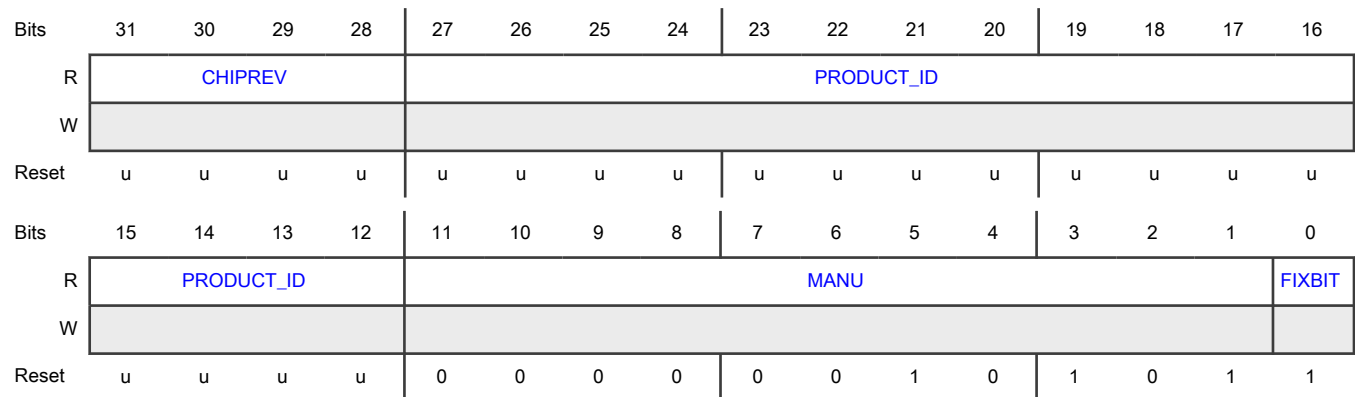
Register	Offset
JTAG_ID	68h

Function

The Product_ID is reset to a value of 0x0000 CAFE, however, BOOTROM will load the PRODUCT_ID from OTP.

The Product ID register is reset to a value of 0x0000 CAFE, however, if the OTP signature is valid and the BOOTROM takes care of its responsibilities, the value found here will be determined by what is programmed in the Product ID Word's location within the OTP.

It is expected on the first silicon that the CHIPREV revision be 0x0. Any subsequent mask change/update (Full or Partial) should result in the CHIPREV revision being upgraded to the next level.

Diagram**Fields**

Field	Function
31-28 CHIPREV	JTAG ID 4-Bit Chip Silicon Revision JTAG ID 4-Bit Chip Silicon Revision.
27-12 PRODUCT_ID	JTAG ID Product ID as defined in the Product ID register JTAG ID Product ID as defined in the Product ID register.
11-1 MANU	JTAG ID Manufacturer JTAG ID Manufacturer (NXP)
0 FIXBIT	JTAG ID fix bit. JTAG ID fix bit.

4.5.1.13 Non-secure GPIO PSYNC (NSGPIO_PSYNC)**Offset**

Register	Offset
NSGPIO_PSYNC	70h

Function

It selects the amount of synchronization stages in the Non-Secure GPIO inputs.

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 PSYNC	Synchronization Stage Setting: 0: 2-Stage Sync. 1: 1-Stage Sync.

4.5.1.14 Secure GPIO PSYNC (SGPIO_PSYNC)**Offset**

Register	Offset
SGPIO_PSYNC	74h

Function

It selects the amount of synchronization stages in the Secure GPIO inputs.

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 PSYNC	Synchronization Stage Setting: 0b - 2-Stage Sync 1b - 1-Stage Sync

4.5.1.15 Auto clock gate override 0 (AUTOCLKGATEOVERRIDE0)**Offset**

Register	Offset
AUTOCLKGATEOVERRIDE0	80h

Function

It controls enable of automatic clock gating of the corresponding logic/circuit, or override the automatic clock gating, keeping the logic/circuit continuously clocking.

NOTE

HWWAKE functionality requires AUTOCLKGATEOVERRIDE0 clock gating enabled, otherwise continuous clock will not allow entry into low power mode.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
R	Reserved																	
W																		
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R	Reserved												DMAC	DMAC	CASP	CRC_	AHB2	AHB2
W													1	0	ER	ENG...	APB1	APB0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		

Fields

Field	Function
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 DMAC1	DMAC1 0b - Enable clock gating 1b - Continuous Clocking
4 DMAC0	DMAC0 0b - Enable clock gating 1b - Continuous Clocking
3 CASPER	CASPER 0b - Enable clock gating 1b - Continuous Clocking
2 CRC_ENGINE	CRC_ENGINE 0b - Enable clock gating 1b - Continuous Clocking
1 AHB2APB1	AHB2APB1 0b - Enable clock gating 1b - Continuous Clocking
0 AHB2APB0	AHB2APB0 0b - Enable clock gating 1b - Continuous Clocking

4.5.1.16 Auto clock gate override 1 (AUTOCLKGATEOVERRIDE1)

Offset

Register	Offset
AUTOCLKGATEOVERRIDE1	84h

Function

It controls enable of automatic clock gating of the corresponding logic/circuit, or override the automatic clock gating, keeping the logic/circuit continuously clocking.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	31_...	30_...	29_...	28_...	27_...	26_...	25_...	24_...	23_...	22_...	21_...	20_...	19_...	18_...	17_...	16_...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	15_...	14_...	13_...	12_...	11_...	10_...	9_IF	8_IF	7_IF	6_IF	5_IF	4_IF	3_IF	2_IF	1_IF	0_IF
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 SRAMn_IF	SRAMn_IF 0b - Enable clock gating 1b - Continuous Clocking

4.5.1.17 Clock gate override 0 (CLKGATEOVERRIDE0)**Offset**

Register	Offset
CLKGATEOVERRIDE0	A0h

Function

Control the enable of the automatic clock gating of corresponding logic/circuit, or override this automatic clock gating, keeping the logic/circuit continuously clocking.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								PMC		ACMP	MU	ADC	USBP HY	SDIO1	SDIO0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-7 —	Reserved
6 PMC	PMC 0b - Enable clock gating 1b - Continuous Clocking
5 ACMP	ACMP 0b - Enable clock gating 1b - Continuous Clocking
4 MU	MU 0b - Enable clock gating 1b - Continuous Clocking
3 ADC	ADC 0b - Enable clock gating 1b - Continuous Clocking
2 USBPHY	USBPHY 0b - Enable clock gating 1b - Continuous Clocking
1 SDIO1	SDIO1 0b - Enable clock gating 1b - Continuous Clocking
0 SDIO0	SDIO0 0b - Enable clock gating 1b - Continuous Clocking

4.5.1.18 AHB SRAM access disable (AHB_SRAM_ACCESS_DISABLE)**Offset**

Register	Offset
AHB_SRAM_ACCESS_DISABLE	FCh

Function

Set bit to disable AHB bus access to corresponding SRAM partition. This is automatically done for all partitions when entering deep sleep mode and main_clk_shutoff is active in PDSLEEPCFG to force the RAM arbitration to select the AXI or the DSP. This needs

to be set at least 2 clocks prior to shutting off the AHB clock to the corresponding SRAM partition, otherwise the AXI or the DSP may not be able to access the RAM.

Please refer "System memory map (Arm M33 overview)" for more details on partition number and SRAM address range.

NOTE

Whenever the peripheral for a ram partition is powered-down, the corresponding bits in AHB, AXI and DSP SRAM access disable registers should also be set to prevent significant waste of power.

NOTE

When a partition is not enabled, a bus fault will not be generated. Accesses to disabled partitions should be avoided.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	031...	030...	029...	028...	027...	026...	025...	024...	023...	022...	021...	020...	019...	018...	017...	016...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	015...	014...	013...	012...	011...	010...	09_...	08_...	07_...	06_...	05_...	04_...	03_...	02_...	01_...	00_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SRAM0n_IF	Control AHB access to SRAM partition n 0b - Enable 1b - Disable

4.5.1.19 AXI SRAM access disable (AXI_SRAM_ACCESS_DISABLE)

Offset

Register	Offset
AXI_SRAM_ACCESS_DISABLE	100h

Function

Set bit to disable AXI bus access to corresponding SRAM partition. This is automatically done for all partitions when entering deep sleep mode and main_clk_shutoff is active in PDSLEEPCFG to force the RAM arbitration to select the AHB or the DSP. This needs to be set at least 2 clocks prior to shutting off the AXI clock to the corresponding SRAM partition, otherwise the AHB or the DSP may not be able to access the RAM.

NOTE

Whenever the periphery for a ram partition is powered-down, the corresponding bits in AHB, AXI and DSP SRAM access disable registers should also be set to prevent significant waste of power.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	031...	030...	029...	028...	027...	026...	025...	024...	023...	022...	021...	020...	019...	018...	017...	016...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	015...	014...	013...	012...	011...	010...	09...	08...	07...	06...	05...	04...	03...	02...	01...	00...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SRAM0n_IF	Control AXI access to SRAM partition 0b - Enable 1b - Disable

4.5.1.20 DSP SRAM access disable (DSP_SRAM_ACCESS_DISABLE)**Offset**

Register	Offset
DSP_SRAM_ACCESS_DISABLE	104h

Function

Set bit to disable Fusion DSP access to the corresponding SRAM partition. This needs to be set at least 2 clocks prior to shutting off the DSP clock, otherwise the AHB or the AXI bus may not be able to access the RAM.

NOTE

Whenever the periphery for a ram partition is powered-down, the corresponding bits in AHB, AXI and DSP SRAM access disable registers should also be set to prevent significant waste of power.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	15_...	14_...	13_...	12_...	11_...	10_...	9_IF	8_IF	7_IF	6_IF	5_IF	4_IF	3_IF	2_IF	1_IF	0_IF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 SRAMn_IF	SRAMn_IF 0b - Enable 1b - Disable

4.5.1.21 Power-Quad Memory Control (PQ_MEM_CTRL)**Offset**

Register	Offset
PQ_MEM_CTRL	110h

Function

Bits directly control the Power-Quad Memory Input Gate and Standby inputs. The bits are set only if the Power-Quad clock is disabled or if the Power-Quad Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

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

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

Fields

Field	Function
31-2 —	Reserved
1 SRAM_STDBY	SRAM Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
0 SRAM_IG	SRAM Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

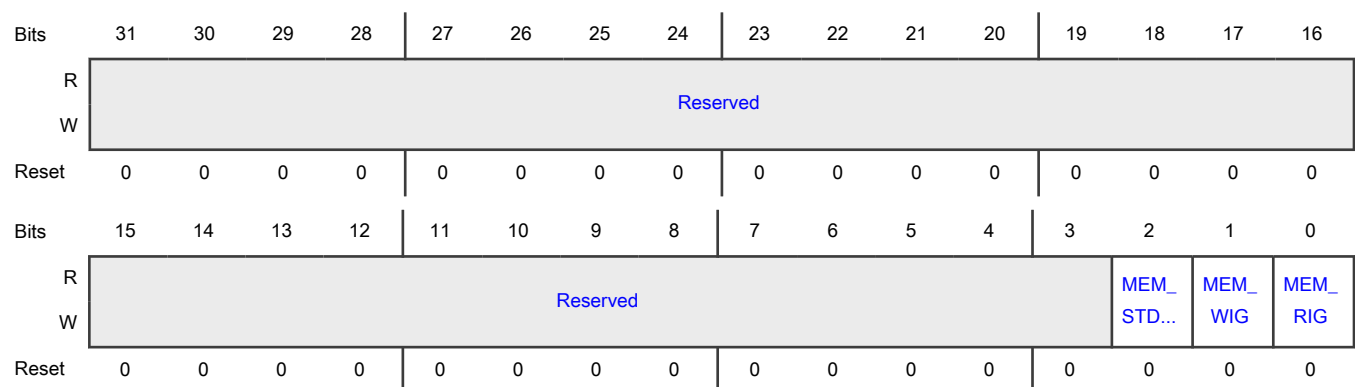
4.5.1.22 FlexSPI0 Memory Control (FLEXSPI0_MEM_CTRL)**Offset**

Register	Offset
FLEXSPI0_MEM_CTRL	114h

Function

Bits directly control the FlexSPI0 Memory Input Gate and Standby inputs. Set only if the FlexSPI0 clock is disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram**Fields**

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.23 USBHS Memory Control (USBHS_MEM_CTRL)

Offset

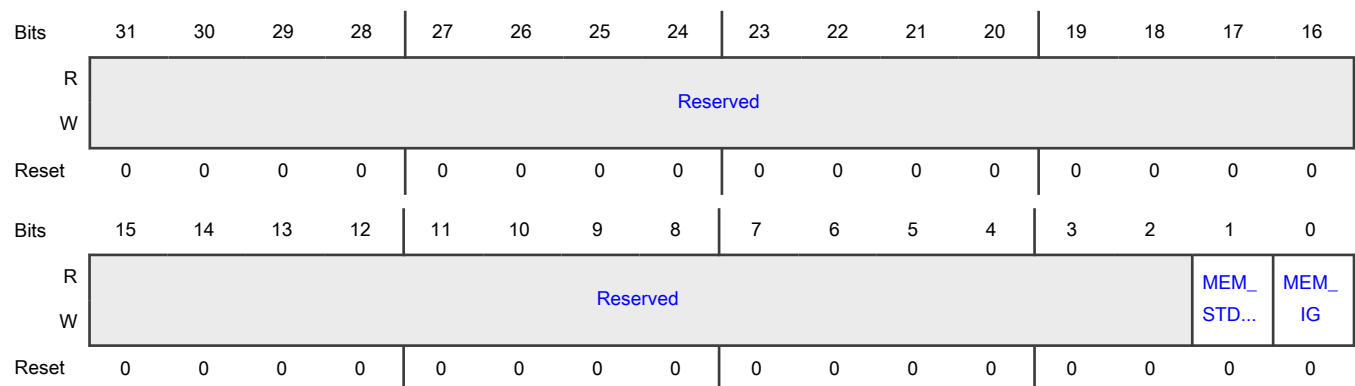
Register	Offset
USBHS_MEM_CTRL	118h

Function

Bits directly control the USB Memory Input Gate and Standby inputs. Set only if the USB Memory clock is disabled or the RAMs will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
0 MEM_IG	Memory Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable

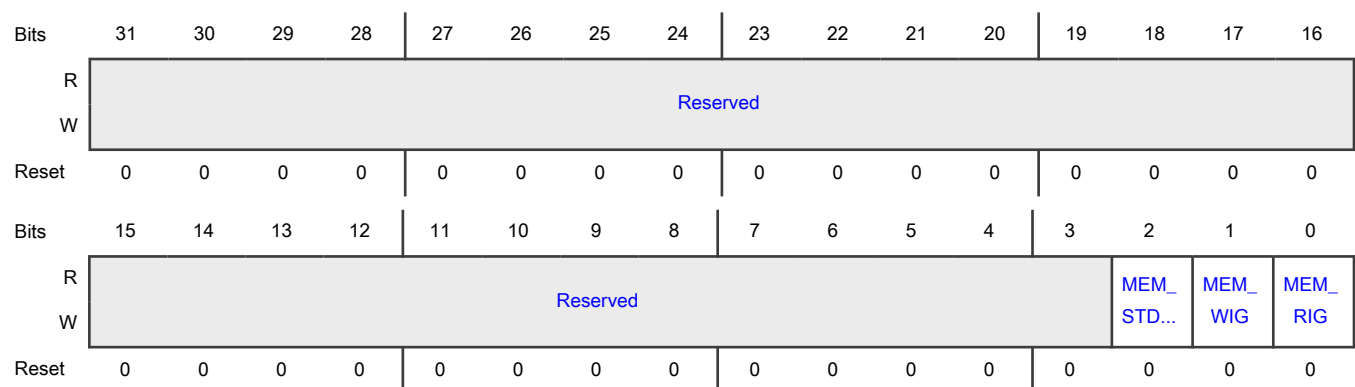
4.5.1.24 USDHC0 Memory Control (USDHC0_MEM_CTRL)**Offset**

Register	Offset
USDHC0_MEM_CTRL	11Ch

Function

Bits directly control the USDHC0 Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the USDHC0 Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram**Fields**

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.25 USDHC1 Memory Control (USDHC1_MEM_CTRL)

Offset

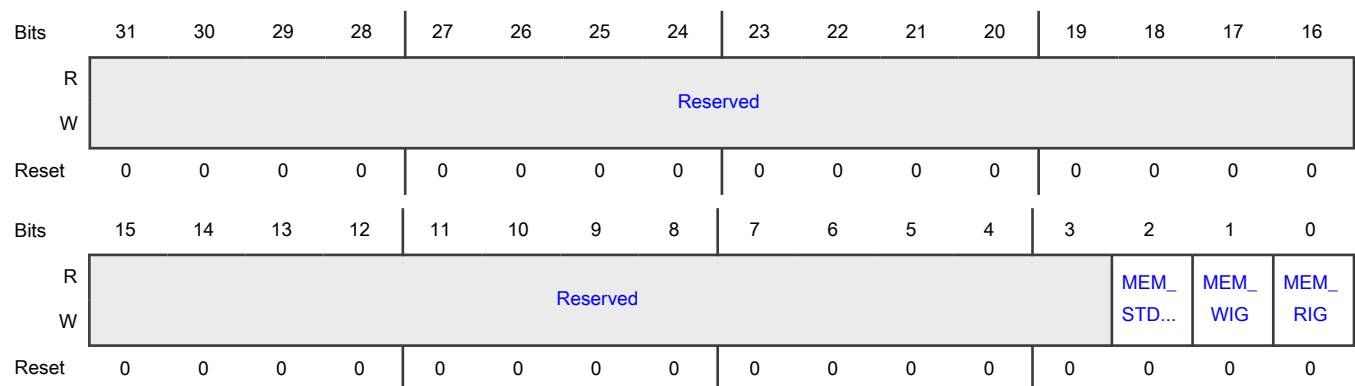
Register	Offset
USDHC1_MEM_CTRL	120h

Function

Bits directly control the USDHC1 Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the USDHC1 Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.26 CASPER Memory Control (CASPER_MEM_CTRL)

Offset

Register	Offset
CASPER_MEM_CTRL	124h

Function

Bits directly control the Casper Memory Input Gate and Standby inputs. Set only if the Casper Memory clock is disabled or the RAMs will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														MEM_ STD...	MEM_ IG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31-2 —	Reserved
1 MEM_STDBY	Auto Standby Control Disable 0b - STDBY is controlled by auto clock gating signal. 1b - STDBY Is forced low.
0 MEM_IG	Auto Input Gate Control Disable 0b - Input Gate is controlled by auto clock gating signal. 1b - Input Gate Is forced low.

4.5.1.27 ROM Memory Control (ROM_MEM_CTRL)**Offset**

Register	Offset
ROM_MEM_CTRL	128h

Function

Bits directly control the ROM Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the ROM Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												MEM_ LS	MEM_ STD...	MEM_ IG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 MEM_LS	MEM LS 0b - Enable 1b - Disable
1 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
0 MEM_IG	Memory Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.28 FlexSPI1 Memory Control (FlexSPI1_MEM_CTRL)

Offset

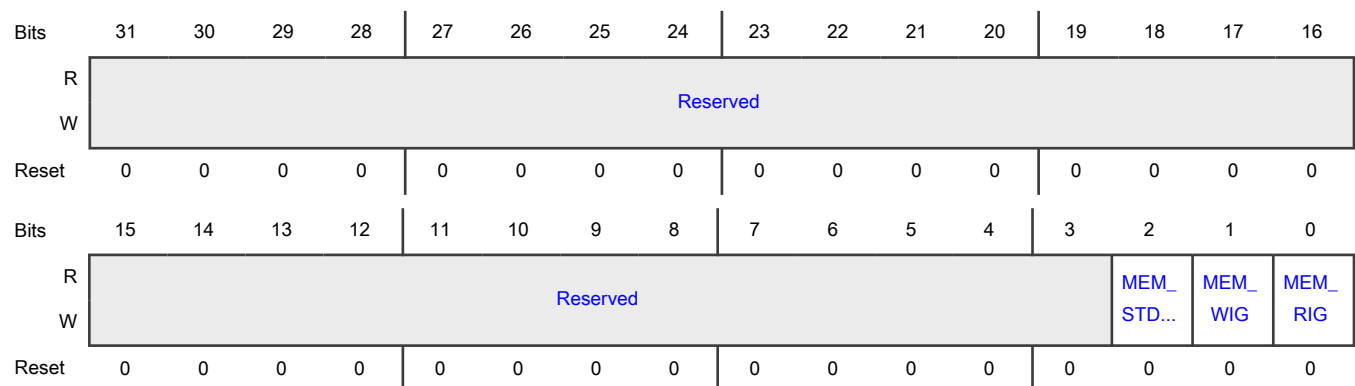
Register	Offset
FlexSPI1_MEM_CTRL	12Ch

Function

Bits directly control the FlexSPI1 Memory Input Gate and Standby inputs. Set only if the FlexSPI1 clock is disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.29 GPU Memory Control (GPU_MEM_CTRL)

Offset

Register	Offset
GPU_MEM_CTRL	130h

Function

Bits directly control the GPU Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the GPU Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												MEM_ STD...	MEM_ WIG	MEM_ RIG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-3 —	Reserved
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

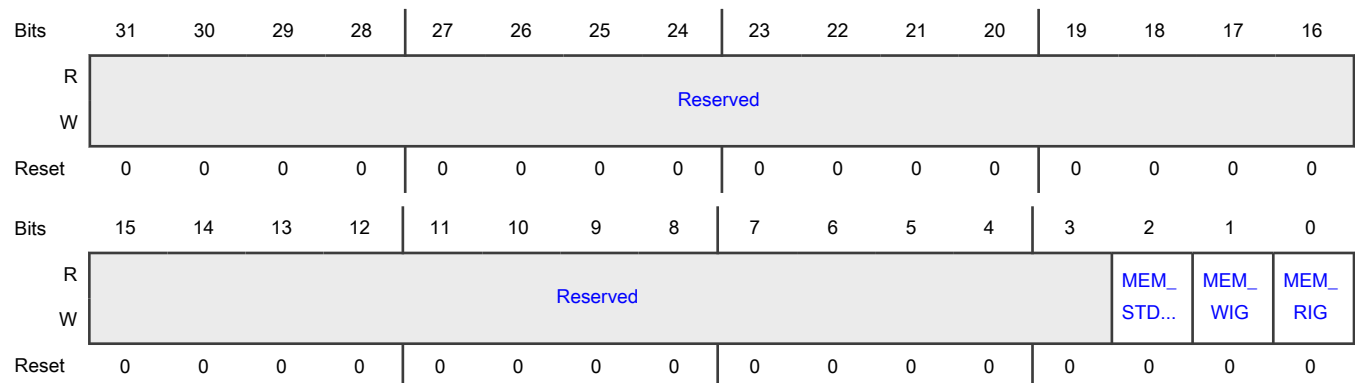
4.5.1.30 MIPI Memory Control (MIPI_MEM_CTRL)**Offset**

Register	Offset
MIPI_MEM_CTRL	134h

Function

Bits directly control the MIPI-DSI Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the MIPI Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

Fields

Field	Function
31-3 —	Reserved
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.31 LCDIF Memory Control (DCN_MEM_CTRL)

Offset

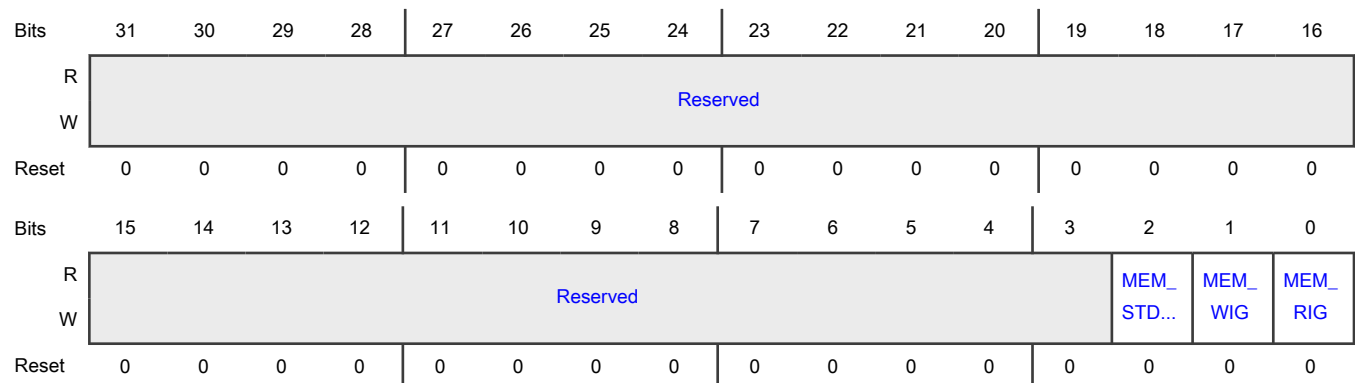
Register	Offset
DCN_MEM_CTRL	138h

Function

Bits directly control the LCDIF Memory Read Input Gate, Write Input Gate, and STDBY inputs. Set only if the LCDIF Memory clocks are disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
1 MEM_WIG	Memory Write Input Gate - Blocks the write input signals to dual-port memory 0b - Enable 1b - Disable
0 MEM_RIG	Memory Read Input Gate - Blocks the read input signals to dual-port memory 0b - Enable 1b - Disable

4.5.1.32 SMARTDMA Memory Control (SMARTDMA_MEM_CTRL)**Offset**

Register	Offset
SMARTDMA_MEM_CTRL	13Ch

Function

Bits directly control the SMARTDMA Memory Input Gate and Standby inputs. Set only if the SMARTDMA clock is disabled or the Memories will not be accessed.

Once asserted, they will reduce the dynamic and leakage power.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														MEM_ STD...	MEM_ IG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

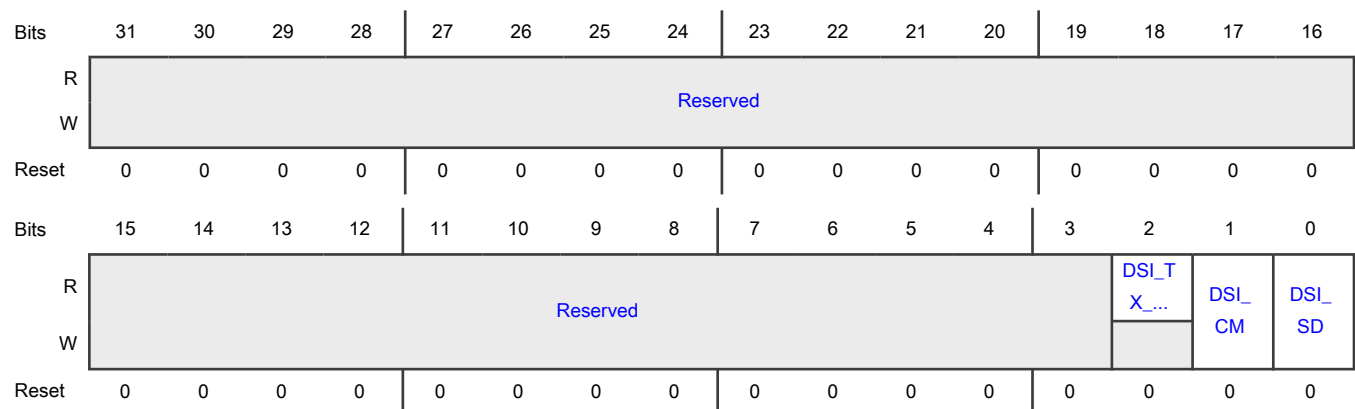
Field	Function
31-2 —	Reserved
1 MEM_STDBY	Memory Standby - Powers the driver to dual-port memory 0b - Enable 1b - Disable
0 MEM_IG	Memory Input Gate 0b - Enable 1b - Disable

4.5.1.33 MIPI DSI Control (MIPI_DSI_CTRL)**Offset**

Register	Offset
MIPI_DSI_CTRL	400h

Function

DSI_SD and DSI_CM signals are synchronized with the DPI clock coming from display controller. DSI_SD and DSI_CM signals being propagated to DSI interface by turning on the DPI mode of DC Nano.

Diagram**Fields**

Field	Function
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
2 DSI_TX_ACTIVE	DSI TX ACTIVE Asserted when the MIPI-DSI is actively transmitting data or when it has accepted a request from the user but has not yet started transmitting.
1 DSI_CM	DSI Color Mode Control. Color Mode Control for Type-4 Display only. This feature is used to configure the display on Normal Mode or Low-color mode. When bit is configured to low-color mode, display works in 8-bit mode and display power is reduced: 0b - Normal mode (full color) (default) 1b - Low color mode (8-bit)
0 DSI_SD	DSI Shutdown Control. Shutdown Control for Type-4 Display only. This feature triggers the display module to shut down the streaming video interface to reduce power consumption. 0b - Shutdown command not to be sent to the Type-4 display (default). 1b - Shutdown command to be sent to the Type-4 display

4.5.1.34 USB Clock Control (USB0CLKCTRL)

Offset

Register	Offset
USB0CLKCTRL	40Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												HS_D	POL_F	AP_FS	POL_F
W													EV_...	S_...	_H...	S_...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Fields

Field	Function
31-5 —	Reserved
4 HS_DEV_WAK EUP_N	External user wake-up signal for device mode Asserting this signal (active low) will result in exiting the low power mode; input to asynchronous control logic. 0b - Forces USB0 PHY to wake-up 1b - Normal USB0 PHY behavior
3 POL_FS_HOST _CLK	USB0 HOST need clock polarity for triggering the USB0 wake-up interrupt 0b - Falling edge of host need_clock triggers wake-up 1b - Rising edge of host need_clock triggers wake-up
2 AP_FS_HOST_ CLK	USB0 Host need clock signal control 0b - Under hardware control 1b - Forced high
1 POL_FS_DEV_ CLK	USB0 Device need clock polarity for triggering the USB0 wake-up interrupt 0b - Falling edge of device need_clock triggers wake-up 1b - Rising edge of device need_clock triggers wake-up
0 AP_FS_DEV_C LK	USB0 Device need clock signal control : 0b - Under hardware control 1b - Forced high

4.5.1.35 USB Clock Status (USB0CLKSTAT)**Offset**

Register	Offset
USB0CLKSTAT	410h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														HOST_	DEV_N
W															NE...	EE...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fields

Field	Function
31-2 —	Reserved
1 HOST_NEED_CLKST	USB0 Device Host USB0_NEEDCLK signal status 0b - Low 1b - High
0 DEV_NEED_CLKST	USB0 Device USB0_NEEDCLK signal status 0b - Low 1b - High

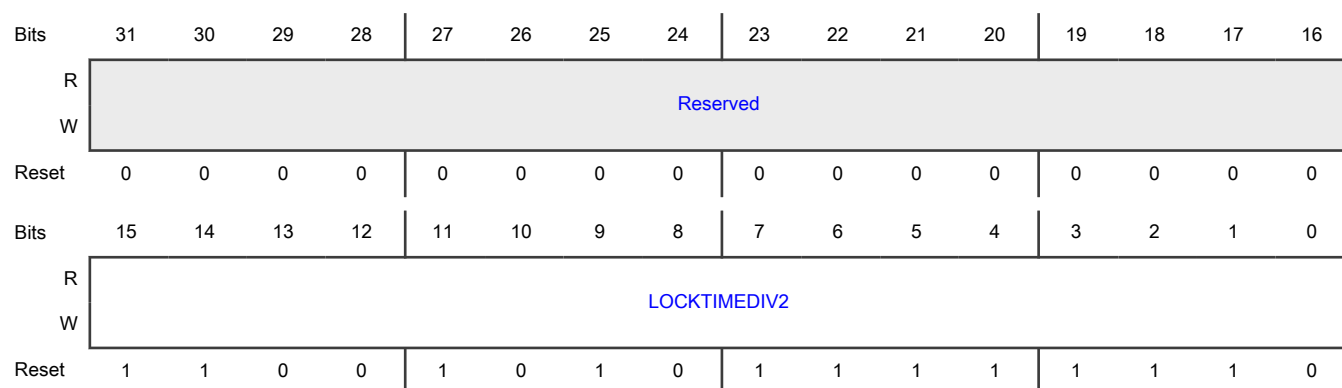
4.5.1.36 USB PHY PLL0 lock time division (USBPHYPLL0LOCKTIMEDIV2)**Offset**

Register	Offset
USBPHYPLL0LOCKTIMEDIV2	414h

Function

The BOOTROM will copy the USBPHYPLL0 worst case characterized lock time value found in the OTP Fuses into the respective fields (if the OTP valid signature is present) on any reset.

S/W needs this value in order to control the HOLDRINGOFF bit in the USBPLL0CTL0 register to know when half the lock time has expired and when the full lock has passed. Any PLL lock sequence requires the HOLDRINGOFF to be enabled ('1') during the first half of the lock time and disabled ('0') during the second half of the lock time.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 LOCKTIMEDIV 2	USBPHYPLL0 Lock Time Programmed lock time is in uS (micro-seconds) and is programmed as half the actual lock time value.

4.5.1.37 Sleep configuration 0 (PDSLEEPCFG0)**Offset**

Register	Offset
PDSLEEPCFG0	600h

Function

It controls the power to various blocks while the CPU is in the deep sleep reduced power mode. Entering reduced power modes is typically accomplished by calling the power mode entry API. It is also possible to configure the PDSLEEPCFG registers directly, set the SLEEPDEEP bit of the Cortex-M33 SCR register, then execute a WFI instruction to enter reduced power modes.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HSPA	Reserv	HSPA	Reserv	HSPA	Reserv	ACMP	PMC	ADC	ADC	ADC	AUDP	AUDP	SYSP	SYSP	FFRO
W	D_F...	ed	D_S...	ed	D_F...	ed	_PD	TEM...	TEM...	LP	PD	LLA...	LLL...	LLA...	LLL...	_PD
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RBBS	LPOS	SYSX	FBB	RBB	HVDC	LVDC	PORC	HVD1	PMCR	FRO	VDDC	DEEP	PMIC	PMIC	MAIN
W	RAM...	C_PD	TAL...	PD	PD	ORE...	ORE...	ORE...	V8_...	EF_...	CG	ORE...	_PD	MO...	MO...	CLK...
Reset	1	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0

Fields

Field	Function
31 HSPAD_FSPI1_REF_PD	Hi speed pad sleep mode 0b - High speed pad refs in normal mode 1b - High speed pad refs in sleep mode
30 —	Reserved
29 HSPAD_SDIO0_REF_PD	High Speed Pad VREF 0b - Normal mode 1b - Sleep mode
28 —	Reserved
27 HSPAD_FSPI0_REF_PD	High speed pad sleep mode 0b - High speed pad refs in normal mode 1b - High speed pad refs in sleep mode
26 —	Reserved
25 ACMP_PD	Analog comparator 0b - Enabled 1b - Powerdown
24 PMC_TEMPSN_S_PD	PMC temperature sensor 0b - Enabled 1b - Powerdown
23	ADC temperature sensor

Table continues on the next page...

Table continued from the previous page...

Field	Function
ADC_TEMPSN S_PD	0b - Enabled 1b - Powerdown
22 ADC_LP	ADC low power mode To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - Enabled 1b - Powerdown
21 ADC_PD	ADC analog functions To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - Enabled 1b - Powerdown
20 AUDPLLANA_P D	Audio PLL analog functions 0b - Enabled 1b - Powerdown
19 AUDPLLLDO_P D	Audio PLL internal regulator 0b - Enabled 1b - Powerdown
18 SYSPLLANA_P D	System PLL analog functions 0b - Enabled 1b - Powerdown
17 SYSPLLLDO_P D	System PLL internal regulator 0b - Enabled 1b - Powerdown
16 FFRO_PD	FRO 192/96 MHz internal oscillator NOTE If the FRO is enabled in Deep Sleep mode, the PMCREF_LP bit should be cleared in PDSLEEPCFG0. 0b - Enabled 1b - Powerdown
15 RBBSRAM_PD	RBBSRAM This bit should be cleared when going into deep sleep to reduce leakage.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Enables SRAM Reverse Body Bias 1b - Disables SRAM Reverse Body Bias
14 LPOSC_PD	1 MHz Low-Power oscillator 0b - Enabled 1b - Powerdown
13 SYSXTAL_PD	Main crystal oscillator 0b - Enabled 1b - Powerdown
12 FBB_PD	Forward body-bias Writes to this bit in PDRUNCFG, but not PDSLEEPCFG, can be disabled by an OTP bit. NOTE FBB_PD and RBB_PD both must not remain cleared or enabled simultaneously. 0b - Enabled 1b - Powerdown
11 RBB_PD	Reverse body-bias Writes to this bit in PDRUNCFG, but not PDSLEEPCFG, can be disabled by an OTP bit. NOTE FBB_PD and RBB_PD both must not remain cleared or enabled simultaneously. 0b - Enabled 1b - Powerdown
10 HVD CORE_PD	HVD 0b - Enabled 1b - Powerdown
9 LVD CORE_LP	LVD LVD0V85 mode 0b - HP mode 1b - LP mode
8 POR CORE_LP	LVD LD0V6 mode 0b - HP mode 1b - LP mode

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 HVD1V8_PD	HVD 0b - Enabled 1b - Powerdown
6 PMCREF_LP	Internal PMC references LP mode 0b - HP mode 1b - LP mode
5 FRO_CG	192/96 FRO Clock Gate This bit controls the first clock gating cell on the output of the FRO (before the dividers) 0b - FRO clock to the dividers is NOT gated 1b - FRO clock to the dividers is gated off
4 VDDCOREREG_LP	Vddcore regulator mode 0b - HP mode 1b - LP mode
3 DEEP_PD	Deep power-down mode This bit controls the VDDCORE supply when using the on-chip regulator. This bit is not intended to be set in PDRUNCFG, but should only be set in PDSLEEPCFG. 0b - VDDCORE supply remains on during WFI (deep_sleep mode) 1b - VDDCORE powered-off during WFI (deep_powerdown mode)
2 PMIC_MODE1	PMIC_MODE1 device pin 0b - Set mode to 0 1b - Set mode to 1
1 PMIC_MODE0	PMIC_MODE0 device pin 0b - Set mode to 0 1b - Set mode to 1
0 MAINCLK_SHU TOFF	Main clock shut off 0b - Clocks enabled 1b - Clocks disabled

4.5.1.38 Sleep configuration 1 (PDSLEEPCFG1)

Offset

Register	Offset
PDSLEEPCFG1	604h

Function

It controls the power to various modules while the CPU is in the deep sleep reduced power mode. The RAMs controlled by this register have two power controls, one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers). This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	HSPA	Reserv	ROM_	OTP_	MIPID	DSP_	Reserv	Reserv	Reserv	LCDIF	LCDIF	MIPID	MIPID	SMAR	SMAR
W	_SL...	D_S...	ed	PD	PD	SI...	PD	ed	ed	ed	_S...	_S...	SI...	SI...	TDM...	TDM...
Reset	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPU_	GPU_	CASP	Reserv	USDH	USDH	USDH	USDH	USBH	USBH	FLEXS	FLEXS	FLEXS	FLEXS	PQ_S	Reserv
W	SRA...	SRA...	ER...	ed	C1_...	C1_...	C0_...	C0_...	S_S...	S_S...	PI...	PI...	PI...	PI...	RAM...	ed
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 SRAM_SLEEP	SRAM sleep mode SRAM_SLEEP is only required if - VDD is to be dropped to 0.6V in deep_sleep mode and - At least one of the small, dedicated RAM's needs to have its data retained. NOTE If these two conditions don't apply, the SRAM_SLEEP bit should not be set. SRAM_SLEEP bit applies only to the dual-port RAMs such as all of the GPU, LCDIF, MIPI DSI, uSDHC and FLEXSPI RAM's. It does not apply to the SmartDMA RAM and is not required to retain SmartDMA RAM contents with VDD lowered to 0.6V. It is not required that the SmartDMA use the SmartDMA RAM, both SmartDMA buses have access to the 5MB of system RAM. 0b - RAM Normal mode 1b - RAM Sleep mode. Needed when vddcore can be < 0.6V to ensure contents retained. Memories not accessible in this mode.
30 HSPAD_SDIO1_REF_PD	High speed pad SDIO1 sleep mode High Speed Pad VREF 0b - Enabled 1b - Powerdown

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28 ROM_PD	ROM 0b - ROM Powered 1b - ROM not Powered
27 OTP_PD	OTP 0b - Powered 1b - Not Powered
26 MIPIDSI_PD	MIPIDSI NOTE Whenever power is removed from the MIPI DSI Host Controller, the MIPI-DPHY[PD_DPHY] must be disable before powering down and enabled after powering up based on the procedure outlined in Chip-specific MIPI DSI information. 0b - MIPI DSI not power gated 1b - MIPI DSI power gated
25 DSP_PD	DSP 0b - DSP not power gated 1b - DSP power gated
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 LCDIF_SRAM_PPD	Periphery Power for LCDIF SRAM 0b - Power down disabled or Powered ON 1b - Powerdown
20 LCDIF_SRAM_APD	Array Power for LCDIF SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 MIPIDSI_SRAM_PPD	Periphery Power for MIPIDSI SRAM 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
18 MIPIDSI_SRAM_APD	Array Power for MIPIDSI SRAM 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
17 SMARTDMA_SRAM_PPD	Periphery Power for SMARTDMA SRAM 0b - Enable 1b - Powerdown
16 SMARTDMA_SRAM_APD	Array Power for SMARTDMA SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
15 GPU_SRAM_PPD	Periphery Power for GPU SRAM 0b - Enable 1b - Powerdown
14 GPU_SRAM_APD	Array Power for GPU SRAM 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
13 CASPER_SRAM_PPD	Periphery power for Casper RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
12 —	Reserved
11 USDHC1_SRAM_PPD	Periphery power for uSDHC1 (SD/MMC/SDIO interface) RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
10	Array power for Casper RAM Array Power

Table continues on the next page...

Table continued from the previous page...

Field	Function
USDHC1_SRA M_APD	0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
9 USDHC0_SRA M_PPD	Periphery power for uSDHC0 (SD/MMC/SDIO interface) RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
8 USDHC0_SRA M_APD	Array power for uSDHC0 (SD/MMC/SDIO interface) RAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
7 USBHS_SRAM _PPD	Periphery power for USB RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
6 USBHS_SRAM _APD	Array power for USB RAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
5 FLEXSPI1_SRA M_PPD	Periphery power for FLEXSPI1 RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
4 FLEXSPI1_SRA M_APD	Array power for FLEXSPI1 RAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
3 FLEXSPI0_SRA M_PPD	Periphery power for FLEXSPI0 RAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
2	Array power for FLEXSPI0 RAM

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXSPI0_SRAM_APD	Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
1 PQ_SRAM_PP D	Peripheral power for PowerQuad RAM Peripheral Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
0 —	Reserved

4.5.1.39 Sleep configuration 2 (PDSLEEPCFG2)

Offset

Register	Offset
PDSLEEPCFG2	608h

Function

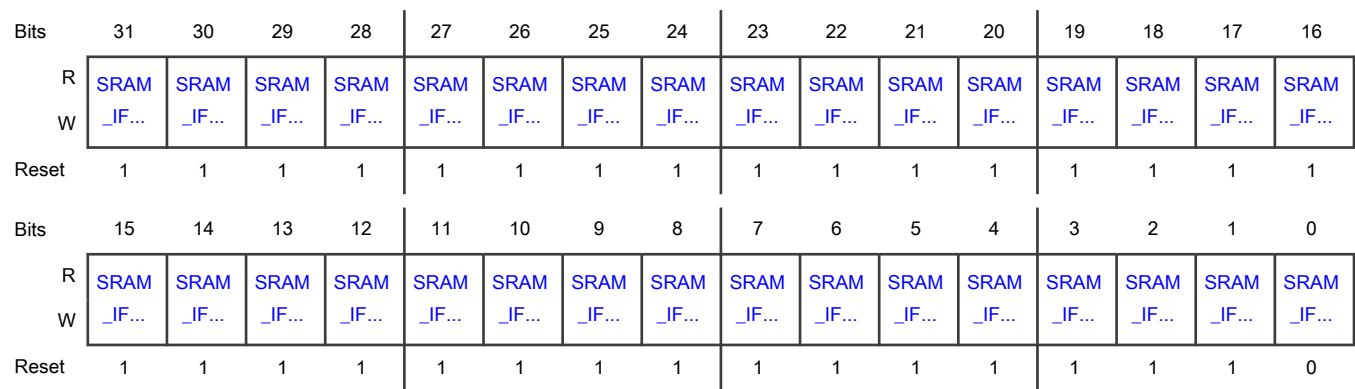
It controls the power to various modules while the CPU is in the deep sleep reduced power mode. The RAMs controlled by this register have two power controls, one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers). This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

NOTE

Whenever the periphery for a ram partition is powered-down, the corresponding bits in AHB, AXI and DSP SRAM access disable registers should also be set to prevent significant waste of power.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram**Fields**

Field	Function
31-0 SRAM_IFn_AP D	Array power for RAM interface n 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF

4.5.1.40 Sleep configuration 3 (PDSLEEPCFG3)**Offset**

Register	Offset
PDSLEEPCFG3	60Ch

Function

This register controls the power to the peripherals of the main system RAMs while the CPU is in the deep sleep reduced power mode. The RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers). This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

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

Fields

Field	Function
31-0 SRAM_IFn_PP D	Periphery power for RAM interface n Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF

4.5.1.41 Run configuration 0 (PDRUNCFG0)**Offset**

Register	Offset
PDRUNCFG0	610h

Function

It controls the power to various modules during normal operation. It is configured using a power API that handles all of the details of altering PDRUNCFG bits that require special handling. Changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

NOTE

The wakeup time can be reduced by turning-off the PLLs before going to sleep and re-enable them after wakeup.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HSPA	Reserv	HSPA	Reserv	HSPA	Reserv	ACMP	PMC	ADC	ADC	ADC	AUDP	AUDP	SYSP	SYSP	FFRO
W	D_F...	ed	D_S...	ed	D_F...	ed	_PD	TEM...	TEM...	LP	PD	LLA...	LLL...	LLA...	LLL...	_PD
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RBBS	LPOS	SYSX	FBB	RBB	HVDC	LVDC	PORC	HVD1	PMCR	FRO	VDDC	DEEP	PMIC	PMIC	MAIN
W	RAM...	C_PD	TAL...	PD	PD	ORE...	ORE...	ORE...	V8_...	EF_...	CG	ORE...	_PD	MO...	MO...	CLK...
Reset	1	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0

Fields

Field	Function
31 HSPAD_FSPI1_REF_PD	Hi speed pad sleep mode 0b - High speed pad refs in normal mode 1b - High speed pad refs in sleep mode
30 —	Reserved
29 HSPAD_SDIO0_REF_PD	High Speed Pad VREF 0b - Normal mode 1b - Sleep mode
28 —	Reserved
27 HSPAD_FSPI0_REF_PD	Hi-speed pad sleep mode 0b - High speed pad refs in normal mode 1b - High speed pad refs in sleep mode
26 —	Reserved
25 ACMP_PD	Analog comparator 0b - Enabled 1b - Powerdown
24 PMC_TEMPSN_S_PD	PMC temperature sensor 0b - Enabled 1b - Powerdown
23	ADC temperature sensor

Table continues on the next page...

Table continued from the previous page...

Field	Function
ADC_TEMPSN S_PD	0b - Enabled 1b - Powerdown
22 ADC_LP	ADC low power mode To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - Enabled 1b - Powerdown
21 ADC_PD	ADC analog functions 0b - Enabled 1b - Powerdown
20 AUDPLLANA_P D	Audio PLL analog functions 0b - Enabled 1b - Powerdown
19 AUDPLLLDO_P D	Audio PLL internal regulator 0b - Enabled 1b - Powerdown
18 SYSPLLANA_P D	System PLL analog functions 0b - Enabled 1b - Powerdown
17 SYSPLLLDO_P D	System PLL internal regulator 0b - Enabled 1b - Powerdown
16 FFRO_PD	FFRO 192/96 MHz internal oscillator NOTE If the FRO is enabled in Deep Sleep mode, the PMCREF_LP bit should be cleared in PDSLEEPCFG0. 0b - Enabled 1b - Powerdown
15 RBBSRAM_PD	Reverse body-bias SRAM 0b - Enables SRAM Reverse Body Bias 1b - Disables SRAM Reverse Body Bias
14	1 MHz Low-Power oscillator

Table continues on the next page...

Table continued from the previous page...

Field	Function
LPOSC_PD	0b - Enabled 1b - Powerdown
13 SYSXTAL_PD	Main crystal oscillator 0b - Enabled 1b - Powerdown
12 FBB_PD	Forward body-bias Writes to this bit in PDRUNCFG, but not PDSLEEPCFG, can be disabled by an OTP bit. NOTE FBB_PD and RBB_PD both must not remain cleared or enabled simultaneously. 0b - Enabled 1b - Powerdown
11 RBB_PD	Reverse body-bias Writes to this bit in PDRUNCFG, but not PDSLEEPCFG, can be disabled by an OTP bit. NOTE FBB_PD and RBB_PD both must not remain cleared or enabled simultaneously. 0b - Enabled 1b - Powerdown
10 HVD CORE_PD	HVD 0b - Enabled 1b - Powerdown
9 LVD CORE_LP	LVD LVD0V85 mode 0b - HP mode 1b - LP mode
8 POR CORE_LP	LVD LD0V6 mode 0b - HP mode 1b - LP mode
7 HVD1V8_PD	HVD 0b - Enabled 1b - Powerdown

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 PMCREF_LP	Internal PMC references LP mode 0b - HP mode 1b - LP mode
5 FRO_CG	192/96 FRO Clock Gate This bit controls the first clock gating cell on the output of the FRO (before the dividers) 0b - FRO clock to the dividers is NOT gated 1b - FRO clock to the dividers is gated off
4 VDDCOREREG_LP	Vddcore regulator mode 0b - HP mode 1b - LP mode
3 DEEP_PD	Deep power-down mode This bit controls the VDDCORE supply when using the on-chip regulator. This bit is not intended to be set in PDRUNCFG, but should only be set in PDSLEEPCFG. 0b - VDDCORE supply remains on during WFI (deep_sleep mode) 1b - VDDCORE powered-off during WFI (deep_powerdown mode)
2 PMIC_MODE1	PMIC_MODE1 device pin 0b - Set mode to 0 1b - Set mode to 1
1 PMIC_MODE0	PMIC_MODE0 device pin 0b - Set mode to 0 1b - Set mode to 1
0 MAINCLK_SHU TOFF	Main clock shut off 0b - Clocks enabled 1b - Clocks gated

4.5.1.42 Run configuration 1 (PDRUNCFG1)

Offset

Register	Offset
PDRUNCFG1	614h

Function

It controls the power to various memory related functions during normal operation. The RAMs controlled by this register have two power controls: one for the actual memory array, and one for the periphery (support circuitry such as line drivers

and sense amplifiers). This provides the option of saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a RAM saves more power, but the contents are lost. Changes to the PDRUNCFGn registers should generally be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	HSPA	Reserv	ROM_	OTP_	MIPID	DSP_	Reserv	Reserv	Reserv	LCDIF	LCDIF	MIPID	MIPID	SMAR	SMAR
W	_SL...	D_S...	ed	PD	PD	SI...	PD	ed	ed	ed	_S...	_S...	SI...	SI...	TDM...	TDM...
Reset	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPU_	GPU_	CASP	Reserv	USDH	USDH	USDH	USDH	USBH	USBH	FLEXS	FLEXS	FLEXS	FLEXS	PQ_S	Reserv
W	SRA...	SRA...	ER...	ed	C1_...	C1_...	C0_...	C0_...	S_S...	S_S...	PI...	PI...	PI...	PI...	RAM...	ed
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 SRAM_SLEEP	SRAM sleep mode SRAM_SLEEP is only required if - VDD is to be dropped to 0.6V in deep_sleep mode and - At least one of the small, dedicated RAM's needs to have its data retained. NOTE If these two conditions don't apply, the SRAM_SLEEP bit should not be set. SRAM_SLEEP bit applies only to the dual-port RAMs such as all of the GPU, LCDIF, MIPI DSI, uSDHC and FLEXSPI RAM's. It does not apply to the SmartDMA RAM and is not required to retain SmartDMA RAM contents with VDD lowered to 0.6V. It is not required that the SmartDMA use the SmartDMA RAM, both SmartDMA buses have access to the 5MB of system RAM. 0b - RAM Normal mode 1b - RAM Sleep mode. Needed when vddcore can be < 0.6V to ensure contents retained. Memories not accessible in this mode.
30 HSPAD_SDIO1_REF_PD	High speed pad SDIO1 sleep mode High Speed Pad VREF 0b - Enabled 1b - Powerdown

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28 ROM_PD	Array periphery power for ROM 0b - ROM Powered 1b - ROM not Powered
27 OTP_PD	OTP 0b - Powered 1b - Not Powered
26 MIPIDSI_PD	MIPIDSI NOTE This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations. 0b - MIPI DSI not power gated 1b - MIPI DSI power gated
25 DSP_PD	DSP NOTE The DSP module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations. 0b - DSP not power gated 1b - DSP power gated
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 LCDIF_SRAM_PPD	LCDIF SRAM Periphery Power 0b - Power down disabled or Powered ON 1b - Powerdown
20	LCDIF SRAM Array Power

Table continues on the next page...

Table continued from the previous page...

Field	Function
LCDIF_SRAM_APD	0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
19 MIPIDSI_SRAM_PPD	MIPIDSI SRAM Periphery Power 0b - Enable 1b - Powerdown
18 MIPIDSI_SRAM_APD	MIPIDSI SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
17 SMARTDMA_SRAM_PPD	SMARTDMA SRAM Periphery Power NOTE This bit can only be set to 1 if SMARTDMA is not used. If SmartDMA is using only the shared RAM then this bit needs to remain cleared (set to 0). 0b - Enable 1b - Powerdown
16 SMARTDMA_SRAM_APD	SMARTDMA SRAM Array Power NOTE This bit can only be set to 1 if SMARTDMA is not used. If SmartDMA is using only the shared RAM then this bit needs to remain cleared (set to 0). 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
15 GPU_SRAM_PD	GPU Periphery Power 0b - Enable 1b - Powerdown
14 GPU_SRAM_APD	GPU Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
13 CASPER_SRAM_PPD	CASPER SRAM Periphery Power 0b - Enable 1b - Powerdown
12 —	Reserved
11	USDHC1 SRAM Periphery Power

Table continues on the next page...

Table continued from the previous page...

Field	Function
USDHC1_SRA M_PPD	0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
10 USDHC1_SRA M_APD	USDHC1 SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
9 USDHC0_SRA M_PPD	USDHC0 SRAM_ Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
8 USDHC0_SRA M_APD	USDHC0 SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
7 USBHS_SRAM _PPD	USBHS SRAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
6 USBHS_SRAM _APD	USBHS SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
5 FLEXSPI1_SRA M_PPD	FLEXSPI1 SRAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
4 FLEXSPI1_SRA M_APD	FLEXSPI1 SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
3 FLEXSPI0_SRA M_PPD	FLEXSPI0 SRAM Periphery Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
2 FLEXSPI0_SRA M_APD	FLEXSPI0 SRAM Array Power 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
1 PQ_SRAM_PP D	Power Quad SRAM Periphery Power

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This bit can only be set to 1 if PowerQuad is not used. If PowerQuad is using only the shared RAM or FLEXSPI1 then this bit needs to remain cleared (set to 0). 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF
0	Reserved
—	

4.5.1.43 Run configuration 2 (PDRUNCFG2)

Offset

Register	Offset
PDRUNCFG2	618h

Function

This register controls the power to the memory array of the main system RAMs during normal operation. The RAMs controlled by this register have two power controls, one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers). This enables saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost. Only the first SRAM partition is powered up after reset, so only that SRAM data is preserved through reset. Changes to the PDRUNCFGn registers should be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

NOTE

Whenever the periphery for a ram partition is powered-down, the corresponding bits in AHB, AXI and DSP SRAM access disable registers should also be set to prevent significant waste of power.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

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

Fields

Field	Function
31-0	Array Power for RAM interface n
SRAM_IFn_AP	0b - Power down disabled or Powered ON
D	1b - Power down enabled or Powered OFF

4.5.1.44 Run configuration 3 (PDRUNCFG3)**Offset**

Register	Offset
PDRUNCFG3	61Ch

Function

This register controls the power to the memory array of the main system RAMs during normal operation. The RAMs controlled by this register have two power controls, one for the actual memory array, and one for the periphery (support circuitry such as line drivers and sense amplifiers). This enables saving power by turning off the periphery of one or more RAMs while retaining the contents of those RAMs for later use. Turning off both portions of a particular RAM saves more power, but the contents are lost. Only the first SRAM partition is powered up after reset, so only that SRAM data is preserved through reset. Changes to the PDRUNCFGn registers should be accomplished by writing to the PDRUNCFGn_SET and/or PDRUNCFGn_CLR registers. This avoids the possibility of an interrupt changing the value of a PDRUNCFG register after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. Reserved bits must not be changed by user software.

NOTE

Once software is running, any combination of SRAM control bits may be modified by multiple write operations to the PDRUNCFG, but have no effect until the changes are activated by the PMC control register[APPLYCFG] bit.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
W	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...	_IF...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Fields

Field	Function
31-0 SRAM_IFn_PP D	Periphery power for RAM interface n. 0b - Power down disabled or Powered ON 1b - Power down enabled or Powered OFF

4.5.1.45 Run configuration 0 set (PDRUNCFG0_SET)

Offset

Register	Offset
PDRUNCFG0_SET	620h

Function

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserv		Reserv		Reserv										
W	HSPA	D_F...	HSPA	D_S...	HSPA	D_F...	ACMP	PMC_	ADC_	ADC_	ADC_	AUDP	AUDP	SYSP	SYSP	FFRO
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RBBS	LPOS	SYSX	FBB_	RBB_	HVDC	LVDC	PORC	HVD1	PMCR	FRO_	VDDC	DEEP	PMIC_	PMIC_	MAIN
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 HSPAD_FSPI1_ REF_PD	High speed pad FSPI1 sleep mode 0b - No effect 1b - Sets the PDRUNCFG0 Bit
30 —	Reserved
29 HSPAD_SDIO0_ REF_PD	High speed pad sleep mode 0b - No effect 1b - Sets the PDRUNCFG0 Bit
28 —	Reserved
27 HSPAD_FSPI0_ REF_PD	High speed pad FSPI0 voltage detect sleep mode 0b - No effect 1b - Sets the PDRUNCFG0 Bit
26 —	Reserved
25 ACMP_PD	Analog comparator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
24 PMC_TEMPSN S_PD	PMC temperature sensor 0b - No effect 1b - Sets the PDRUNCFG0 Bit
23 ADC_TEMPSN S_PD	ADC temperature sensor 0b - No effect 1b - Sets the PDRUNCFG0 Bit
22 ADC_LP	ADC low power mode To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - No effect 1b - Sets the PDRUNCFG0 Bit
21 ADC_PD	ADC analog functions 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the PDRUNCFG0 Bit
20 AUDPLLANA_PD	Audio PLL analog functions 0b - No effect 1b - Sets the PDRUNCFG0 Bit
19 AUDPLLLDO_PD	Audio PLL internal regulator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
18 SYSPLLANA_PD	System PLL analog functions 0b - No effect 1b - Sets the PDRUNCFG0 Bit
17 SYSPLLLDO_PD	System PLL internal regulator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
16 FFRO_PD	FFRO 19296 MHz internal oscillator NOTE If the FRO is enabled in Deep Sleep mode, the PMCREF_LP bit should be cleared in PDSLEEPCFG0. 0b - No effect 1b - Sets the PDRUNCFG0 Bit
15 RBBSRAM_PD	Reverse body-bias SRAM 0b - No effect 1b - Sets the PDRUNCFG0 Bit
14 LPOSC_PD	1 MHz Low-Power oscillator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
13 SYSXTAL_PD	Main crystal oscillator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
12 FBB_PD	Forward body-bias 0b - No effect 1b - Sets the PDRUNCFG0 Bit
11	Reverse body-bias

Table continues on the next page...

Table continued from the previous page...

Field	Function
RBB_PD	0b - No effect 1b - Sets the PDRUNCFG0 Bit
10 HVD CORE_PD	HVD 0b - No effect 1b - Sets the PDRUNCFG0 Bit
9 LVD CORE_LP	LVD 0b - No effect 1b - Sets the PDRUNCFG0 Bit
8 POR CORE_LP	Internal PMC references LP mode 0b - No effect 1b - Sets the PDRUNCFG0 Bit
7 HVD1V8_PD	HVD 0b - No effect 1b - Sets the PDRUNCFG0 Bit
6 PMCREG_LP	Internal PMC references LP mode 0b - No effect 1b - Sets the PDRUNCFG0 Bit
5 FRO.CG	192/96 FRO Clock Gate 0b - No effect 1b - Sets the PDRUNCFG0 Bit
4 VDDCOREREG_LP	Vddcore regulator mode when using on-chip regulator 0b - No effect 1b - Sets the PDRUNCFG0 Bit
3 DEEP_PD	Deep power-down mode This bit controls the VDDCORE supply when using the on-chip regulator. This bit is not intended to be set in PDRUNCFG, but should only be set in PDSLEEPCFG. 0b - VDDCORE supply remains on during WFI (deep_sleep mode) 1b - VDDCORE powered-off during WFI (deep_powerdown mode)
2 PMIC_MODE1	PMIC_MODE1 pin 0b - No effect 1b - Sets the PDRUNCFG0 Bit
1	PMIC_MODE0 pin

Table continues on the next page...

Table continued from the previous page...

Field	Function
PMIC_MODE0	0b - No effect 1b - Sets the PDRUNCFG0 Bit
0 MAINCLK_SHU TOFF	Main clock shut off 0b - No effect 1b - Sets the PDRUNCFG0 Bit

4.5.1.46 Run configuration 1 set (PDRUNCFG1_SET)

Offset

Register	Offset
PDRUNCFG1_SET	624h

Function

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			Reserv					Reserv	Reserv	Reserv						
W	SRAM _SL...	HSPA D_S...		ROM_ PD	OTP_ PD	MIPID SI...	DSP_ PD				LCDIF _S...	LCDIF _S...	MIPID SI...	MIPID SI...	SMAR TDM...	SMAR TDM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				Reserv												
W	GPU_ SRA...	GPU_ SRA...	CASP ER...		USDH C1_...	USDH C1_...	USDH C0_...	USDH C0_...	USBH S_S...	USBH S_S...	FLEXS PI...	FLEXS PI...	FLEXS PI...	FLEXS PI...	PQ_S RAM...	PQ_S RAM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 SRAM_SLEEP	SRAM sleep mode SRAM_SLEEP is only required if - VDD is to be dropped to 0.6V in deep_sleep mode and - At least one of the small, dedicated RAM's needs to have its data retained.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE If these two conditions don't apply, the SRAM_SLEEP bit should not be set. SRAM_SLEEP bit applies only to the dual-port RAMs such as all of the GPU, LCDIF, MIPI DSI, uSDHC and FLEXSPI RAM's. It does not apply to the SmartDMA RAM and is not required to retain SmartDMA RAM contents with VDD lowered to 0.6V. It is not required that the SmartDMA use the SmartDMA RAM, both SmartDMA buses have access to the 5MB of system RAM. 0b - No effect 1b - Sets the PDRUNCFG1 Bit
30 HSPAD_SDIO1 _REF_PD	High speed pad sleep mode 0b - No effect 1b - Sets the PDRUNCFG1 Bit
29 —	Reserved
28 ROM_PD	Array and periphery power for ROM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
27 OTP_PD	Array and periphery power for OTP 0b - No effect 1b - Sets the PDRUNCFG1 Bit
26 MIPIDSI_PD	Array and periphery power for MIPIDSI 0b - No effect 1b - Sets the PDRUNCFG1 Bit
25 DSP_PD	Array and periphery power for DSP 0b - No effect 1b - Sets the PDRUNCFG1 Bit
24 —	Reserved
23 —	Reserved
22 —	Reserved
21	Periphery power for LCDIF

Table continues on the next page...

Table continued from the previous page...

Field	Function
LCDIF_SRAM_PPD	0b - No effect 1b - Sets the PDRUNCFG1 Bit
20 LCDIF_SRAM_APD	Array power for LCDIF 0b - No effect 1b - Sets the PDRUNCFG1 Bit
19 MIPIDSI_SRAM_PPD	Periphery power for MIPIDSI SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
18 MIPIDSI_SRAM_APD	Array power for MIPIDSI SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
17 SMARTDMA_SRAM_PPD	Periphery power for SMARTDMA SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
16 SMARTDMA_SRAM_APD	Array power for SMARTDMA SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
15 GPU_SRAM_PPD	Periphery power for GPU SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
14 GPU_SRAM_APD	Array power for GPU SRAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
13 CASPER_SRAM_PPD	Periphery power for Casper RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
12 —	Reserved
11 USDHC1_SRAM_PPD	Periphery power for uSDHC1 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 USDHC1_SRA M_APD	Array power for uSDHC1 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
9 USDHC0_SRA M_PPD	Periphery power for uSDHC0 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
8 USDHC0_SRA M_APD	Array power for uSDHC0 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
7 USBHS_SRAM _PPD	Periphery power for USB RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
6 USBHS_SRAM _APD	Array power for USB RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
5 FLEXSPI1_SRA M_PPD	Periphery power for FLEXSPI1 0b - No effect 1b - Sets the PDRUNCFG1 Bit
4 FLEXSPI1_SRA M_APD	Array power for FLEXSPI1 0b - No effect 1b - Sets the PDRUNCFG1 Bit
3 FLEXSPI0_SRA M_PPD	Periphery power for FLEXSPI0 0b - No effect 1b - Sets the PDRUNCFG1 Bit
2 FLEXSPI0_SRA M_APD	Array power for FLEXSPI0 0b - No effect 1b - Sets the PDRUNCFG1 Bit
1 PQ_SRAM_PP D	Periphery power for PowerQuad RAM 0b - No effect 1b - Sets the PDRUNCFG1 Bit
0	Array power for PowerQuad RAM

Table continues on the next page...

Table continued from the previous page...

Field	Function
PQ_SRAM_AP D	0b - No effect 1b - Sets the PDRUNCFG1 Bit

4.5.1.47 Run configuration 2 set (PDRUNCFG2_SET)

Offset

Register	Offset
PDRUNCFG2_SET	628h

Function

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SRAM_IFn_AP D	Array power for SRAM interface n. 0b - No effect 1b - Sets the PDRUNCFG2 Bit

4.5.1.48 Run configuration 3 set (PDRUNCFG3_SET)

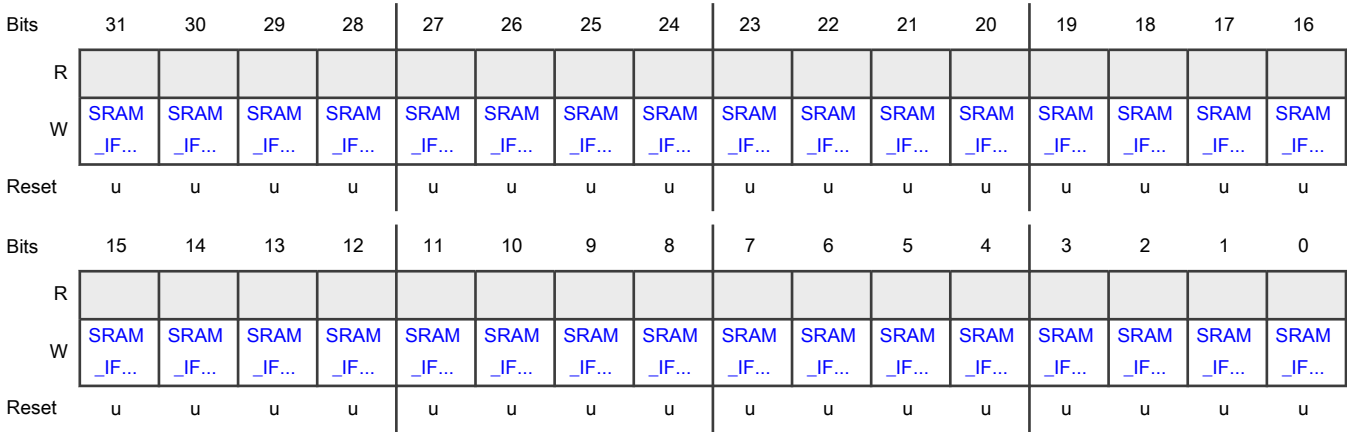
Offset

Register	Offset
PDRUNCFG3_SET	62Ch

Function

Writing a 1 to a bit position in this register sets the corresponding position in PDRUNCFG3.

Diagram



Fields

Field	Function
31-0 SRAM_IFn_PP D	Peripheral power for RAM interface n. 0b - No effect 1b - Sets the PDRUNCFG3 Bit

4.5.1.49 Run configuration 0 clear (PDRUNCFG0_CLR)

Offset

Register	Offset
PDRUNCFG0_CLR	630h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserv		Reserv		Reserv										
W	HSPA D_F...		HSPA D_S...		HSPA D_F...		ACMP _PD	PMC_ TEM...	ADC_ TEM...	ADC_ LP	ADC_ PD	AUDP LLA...	AUDP LLL...	SYSP LLA...	SYSP LLL...	FFRO _PD
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RBBS RAM...	LPOS C_PD	SYSX TAL...	FBB_ PD	RBB_ PD	HVDC ORE...	LVDC ORE...	PORC ORE...	HVD1 V8_...	PMCR EF_...	FRO_ CG	VDDC ORE...	DEEP _PD	PMIC_ MO...	PMIC_ MO...	MAIN CLK...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 HSPAD_FSPI1_ REF_PD	High speed pad FSPIO1 sleep mode 0b - No effect 1b - Clears the PDRUNCFG0 Bit
30 —	Reserved
29 HSPAD_SDIO0_ REF_PD	High speed pad sleep mode 0b - No effect 1b - Clears the PDRUNCFG0 Bit
28 —	Reserved
27 HSPAD_FSPI0_ REF_PD	High speed pad FSPIO0 sleep mode 0b - No effect 1b - Clears the PDRUNCFG0 Bit
26 —	Reserved
25 ACMP_PD	Analog comparator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
24 PMC_TEMPSN S_PD	PMC temperature sensor 0b - No effect 1b - Clears the PDRUNCFG0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 ADC_TEMPSN S_PD	ADC temperature sensor 0b - No effect 1b - Clears the PDRUNCFG0 Bit
22 ADC_LP	ADC low power mode To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - No effect 1b - Clears the PDRUNCFG0 Bit
21 ADC_PD	ADC analog functions To minimize power consumption in deep_sleep or deep_powerdown mode both ADC_LP and ADC_PD bits must be set. 0b - No effect 1b - Clears the PDRUNCFG0 Bit
20 AUDPLLANA_P D	Audio PLL analog functions 0b - No effect 1b - Clears the PDRUNCFG0 Bit
19 AUDPLLLDO_P D	Audio PLL internal regulator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
18 SYSPLLANA_P D	System PLL analog functions 0b - No effect 1b - Clears the PDRUNCFG0 Bit
17 SYSPLLLDO_P D	System PLL internal regulator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
16 FFRO_PD	FRO 16 MHz internal oscillator NOTE If the FRO is enabled in Deep Sleep mode, the PMCREF_LP bit should be cleared in PDSLEEPCFG0. 0b - No effect 1b - Clears the PDRUNCFG0 Bit
15	Reverse body-bias SRAM

Table continues on the next page...

Table continued from the previous page...

Field	Function
RBBSRAM_PD	0b - No effect 1b - Clears the PDRUNCFG0 Bit
14 LPOSC_PD	1 MHz Low-Power oscillator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
13 SYSXTAL_PD	Main crystal oscillator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
12 FBB_PD	Forward body-bias 0b - No effect 1b - Clears the PDRUNCFG0 Bit
11 RBB_PD	Reverse body-bias 0b - No effect 1b - Clears the PDRUNCFG0 Bit
10 HVD CORE_PD	HVD 0b - No effect 1b - Clears the PDRUNCFG0 Bit
9 LVD CORE_LP	LVD 0b - No effect 1b - Clears the PDRUNCFG0 Bit
8 POR CORE_LP	LVD 0b - No effect 1b - Clears the PDRUNCFG0 Bit
7 HVD1V8_PD	HVD 0b - No effect 1b - Clears the PDRUNCFG0 Bit
6 PMCREF_LP	Internal PMC references LP mode 0b - No effect 1b - Clears the PDRUNCFG0 Bit
5 FRO_CG	192/96 FRO Clock Gate 0b - No effect 1b - Clears the PDRUNCFG0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 VDDCOREREG _LP	Vddcore regulator mode when using on-chip regulator 0b - No effect 1b - Clears the PDRUNCFG0 Bit
3 DEEP_PD	Deep power-down mode This bit controls the VDDCORE supply when using the on-chip regulator. This bit is not intended to be set in PDRUNCFG, but should only be set in PDSLEEPCFG. 0b - VDDCORE supply remains on during WFI (deep_sleep mode) 1b - VDDCORE powered-off during WFI (deep_powerdown mode)
2 PMIC_MODE1	PMIC_MODE1 device pin 0b - No effect 1b - Clears the PDRUNCFG0 Bit
1 PMIC_MODE0	PMIC_MODE0 device pin 0b - No effect 1b - Clears the PDRUNCFG0 Bit
0 MAINCLK_SHU TOFF	Main clock shut off 0b - No effect 1b - Clears the PDRUNCFG0 Bit

4.5.1.50 Run configuration 1 clear (PDRUNCFG1_CLR)

Offset

Register	Offset
PDRUNCFG1_CLR	634h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			Reserv					Reserv	Reserv	Reserv						
W	SRAM _SL...	HSPA D_S...		ROM_ PD	OTP_ PD	MIPID SI...	DSP_ PD				LCDIF _S...	LCDIF _S...	MIPID SI...	MIPID SI...	SMAR TDM...	SMAR TDM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				Reserv												
W	GPU_ SRA...	GPU_ SRA...	CASP ER...		USDH C1_...	USDH C1_...	USDH C0_...	USDH C0_...	USBH S_S...	USBH S_S...	FLEXS PI...	FLEXS PI...	FLEXS PI...	FLEXS PI...	PQ_S RAM...	PQ_S RAM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 SRAM_SLEEP	SRAM sleep mode SRAM_SLEEP is only required if - VDD is to be dropped to 0.6V in deep_sleep mode and - At least one of the small, dedicated RAM's needs to have its data retained. NOTE If these two conditions don't apply, the SRAM_SLEEP bit should not be set. SRAM_SLEEP bit applies only to the dual-port RAMs such as all of the GPU, LCDIF, MIPI DSI, uSDHC and FLEXSPI RAM's. It does not apply to the SmartDMA RAM and is not required to retain SmartDMA RAM contents with VDD lowered to 0.6V. It is not required that the SmartDMA use the SmartDMA RAM, both SmartDMA buses have access to the 5MB of system RAM. 0b - No effect 1b - Clears the PDRUNCFG1 Bit
30 HSPAD_SDIO1_REF_PD	High speed pad sleep mode 0b - No effect 1b - Clears the PDRUNCFG1 Bit
29 —	Reserved
28 ROM_PD	Array and periphery power for ROM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
27 OTP_PD	Array and periphery power for OTP 0b - No effect 1b - Clears the PDRUNCFG1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
26 MIPIDSI_PD	Array and periphery power for MIPIDSI 0b - No effect 1b - Clears the PDRUNCFG1 Bit
25 DSP_PD	Array and periphery power for DSP 0b - No effect 1b - Clears the PDRUNCFG1 Bit
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 LCDIF_SRAM_PPD	Periphery power for LCDIF 0b - No effect 1b - Clears the PDRUNCFG1 Bit
20 LCDIF_SRAM_APD	Array power for LCDIF 0b - No effect 1b - Clears the PDRUNCFG1 Bit
19 MIPIDSI_SRAM_PPD	Periphery power for MIPIDSI SRAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
18 MIPIDSI_SRAM_APD	Array power for MIPIDSI SRAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
17 SMARTDMA_SRAM_PPD	Periphery power for SMARTDMA SRAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
16 SMARTDMA_SRAM_APD	Array power for SMARTDMA SRAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
15	Periphery power for GPU RAM

Table continues on the next page...

Table continued from the previous page...

Field	Function
GPU_SRAM_P PD	0b - No effect 1b - Clears the PDRUNCFG1 Bit
14 GPU_SRAM_A PD	Array power for GPU SRAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
13 CASPER_SRA M_PPD	Periphery power for Casper RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
12 —	Reserved
11 USDHC1_SRA M_PPD	Periphery power for uSDHC1 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
10 USDHC1_SRA M_APD	Array power for uSDHC1 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
9 USDHC0_SRA M_PPD	Periphery power for uSDHC0 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
8 USDHC0_SRA M_APD	Array power for uSDHC0 (SD/MMC/SDIO interface) RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
7 USBHS_SRAM _PPD	Periphery power for USB RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
6 USBHS_SRAM _APD	Array power for USB RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
5 FLEXSPI1_SRA M_PPD	Periphery power for FlexSPI1 0b - No effect 1b - Clears the PDRUNCFG1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 FLEXSPI1_SRA M_APD	Array power for FlexSPI1 0b - No effect 1b - Clears the PDRUNCFG1 Bit
3 FLEXSPI0_SRA M_PPD	Periphery power for FlexSPI0 0b - No effect 1b - Clears the PDRUNCFG1 Bit
2 FLEXSPI0_SRA M_APD	Array power for FlexSPI0 0b - No effect 1b - Clears the PDRUNCFG1 Bit
1 PQ_SRAM_PP D	Periphery power for PowerQuad RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit
0 PQ_SRAM_AP D	Array power for PowerQuad RAM 0b - No effect 1b - Clears the PDRUNCFG1 Bit

4.5.1.51 Run configuration 2 clear (PDRUNCFG2_CLR)

Offset

Register	Offset
PDRUNCFG2_CLR	638h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SRAM_IFn_AP D	Array power for RAM interface n 0b - No effect 1b - Clears the PDRUNCFG2 Bit

4.5.1.52 Run configuration 3 clear (PDRUNCFG3_CLR)

Offset

Register	Offset
PDRUNCFG3_CLR	63Ch

Function

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG3.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...	SRAM _IF...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SRAM_IFn_PP D	Periphery power for RAM interface n 0b - No effect 1b - Clears the PDRUNCFG3 Bit

4.5.1.53 PD Wake Configuration (PDWAKECFG)

Offset

Register	Offset
PDWAKECFG	660h

Function

This register is used in a Deep Sleep wakeup event. Depending on the register configuration, system will have PDRUNCFG applied or PDSLEEPCFG copied over to PDRUNCFG. Since these registers are in the vddcore domain these bits have no effect in deep powerdown mode or on wakeup from deep powerdown mode. They go to their reset value and the RBB_PD and FBB_PD bits in PDRUNCFG0 are set to 1 (their reset value) on wakeup from deep powerdown.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								OTPP DKE...		DSPP DKE...		MIPIP DK...	RBBS RAM...	FBBK EEP...	RBBK EEP...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-6 —	Reserved
5 OTPPDKEEPS T	OTP_PD value on wakeup 0b - Use value of OTP_PD in PDRUNCFG on wakeup 1b - Copy PDSLEEPCFG OTP_PD value to PDRUNCFG OTP_PD on wakeup to keep OTP state
4	DSP_PD value on wakeup

Table continues on the next page...

Table continued from the previous page...

Field	Function
DSPDPDKEEPS T	0b - Use value of DSP_PD in PDRUNCFG on wakeup 1b - Copy PDSLEEPCFG DSP_PD value to PDRUNCFG DSP_PD on wakeup to keep DSP state
3 MIPIPDKEEPS T	MIPI_PD value on wakeup 0b - Use value of MIPI_PD in PDRUNCFG on wakeup 1b - Copy PDSLEEPCFG MIPI_PD value to PDRUNCFG MIPI_PD on wakeup to keep MIPI state
2 RBBSRAMKEE PST	RBB SRAM mode on wakeup 0b - Use value of RBBSRAM_PD in PDRUNCFG on wakeup 1b - Copy PDSLEEPCFG RBBSRAM_PD value to PDRUNCFG RBBSRAM_PD on wakeup to keep RBB state
1 FBBKEEPST	FBB mode on wakeup 0b - Use value of FBB_PD in PDRUNCFG on wakeup 1b - Copy PDSLEEPCFG FBB_PD value to PDRUNCFG FBB_PD on wakeup to keep FBB state
0 RBBKEEPST	RBB mode on wakeup 0b - Use value of RBB_PD in PDRUNCFG on wakeup. 1b - Copy PDSLEEPCFG RBB_PD value to PDRUNCFG RBB_PD on wakeup to keep RBB state.

4.5.1.54 Start Enable 0 (STARTEN0)

Offset

Register	Offset
STARTEN0	680h

Function

This register enables an interrupt for wake-up from deep-sleep mode. Many peripherals serve the wake-up function. The peripheral must be sufficiently enabled to cause interrupts in the reduced power mode.

NOTE

Changes to the STARTEN0 registers be accomplished by using the related STARTEN_SET0 and STARTEN_CLR0 registers. This avoids any unintentional setting or clearing of other bits. The corresponding interrupts in the NVIC should be enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	RNG	PMC	HWVAD0	SECUREVIOLATION	HYPERSVISOR	Reserved	DMIC0	ACMP	Reserved	ADC0	FLEXCOM0	FLEXCOM1	FLEXCOM2	FLEXCOM3	FLEXCOM4	FLEXCOM5
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	FLEXCOM6	FLEXCOM7	CT32BIT3	SCT0	CT32BIT1	CT32BIT0	MRT0	UTICK0	GPIOIN0	GPIOIN1	GPIOIN2	GPIOIN3	GPIOIN4	GPIOIN5	DMAC0	WDT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 RNG	RNG wake-up. 0b - Disabled 1b - Enabled
30 PMC	PMC wake-up. 0b - Disabled 1b - Enabled
29 HWVAD0	Hardware VAD wake-up. 0b - Disabled 1b - Enabled
28 SECUREVIOLATION	Secure Violation wake-up. 0b - Disabled 1b - Enabled
27 HYPERVISOR	Hypervisor interrupt wake-up. 0b - Disabled 1b - Enabled
26 —	Reserved
25 DMIC0	DMIC wake-up. 0b - Disabled 1b - Enabled
24 ACMP	ACMP wake-up. 0b - Disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enabled
23 —	Reserved
22 ADC0	ADC wake-up. 0b - Disabled 1b - Enabled
21 FLEXCOMM15	Flexcomm 15 (PMIC I2C) peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
20 FLEXCOMM14	Flexcomm 14 (High Speed SPI) peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
19 FLEXCOMM5	Flexcomm 5 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
18 FLEXCOMM4	Flexcomm 4 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
17 FLEXCOMM3	Flexcomm 3 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
16 FLEXCOMM2	Flexcomm 2 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
15 FLEXCOMM1	Flexcomm 1 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
14 FLEXCOMM0	Flexcomm 0 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
13	CTIMER 3 wake-up

Table continues on the next page...

Table continued from the previous page...

Field	Function
CT32BIT3	0b - Disabled 1b - Enabled
12 SCT0	SCTimer/PWM wake-up. 0b - Disabled 1b - Enabled
11 CT32BIT1	CTIMER 1 wake-up 0b - Disabled 1b - Enabled
10 CT32BIT0	CTIMER 0 wake-up 0b - Disabled 1b - Enabled
9 MRT0	MRT wake-up. 0b - Disabled 1b - Enabled
8 UTICK0	UTICK wake-up. 0b - Disabled 1b - Enabled
7 GPIO_INT0_IR Q3	GPIO pin interrupt 3 wake-up. 0b - Disabled 1b - Enabled
6 GPIO_INT0_IR Q2	GPIO pin interrupt 2 wake-up. 0b - Disabled 1b - Enabled
5 GPIO_INT0_IR Q1	GPIO pin interrupt 1 wake-up. 0b - Disabled 1b - Enabled
4 GPIO_INT0_IR Q0	GPIO pin interrupt 0 wake-up. 0b - Disabled 1b - Enabled
3 GPIO_INTB	Non-secure GPIO interrupt B wake-up. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 GPIO_INTA	Non-secure GPIO interrupt A wake-up. 0b - Disabled 1b - Enabled
1 DMAC0	DMA controller 0 wake-up. 0b - Disabled 1b - Enabled
0 WDT0	Watchdog timer 0 wake-up. 0b - Disabled 1b - Enabled

4.5.1.55 Start Enable 1 (STARTEN1)

Offset

Register	Offset
STARTEN1	684h

Function

This register enables an interrupt for wake-up from deep-sleep mode. Many peripherals serve the wake-up function. The peripheral must be sufficiently enabled to cause interrupts in the reduced power mode.

NOTE

Changes to the STARTEN1 registers be accomplished by using the related STARTEN_SET1 and STARTEN_CLR1 registers. This avoids any unintentional setting or clearing of other bits. The corresponding interrupts in the NVIC should be enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FLEX	FLEX	FLEX	FLEX	SHA	PMIC	CASP	POWE	PUF	DMAC	USB_	Reserv	USB0_	Reserved		SGPIO
W	COM...	COM...	COM...	COM...			ER	RQU...		1	PHY...	ed	NE...			_J...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SGPIO			FLEX	FLEX	FLEX	OS_E	CT32B	CT32B	GPIO_	GPIO_	GPIO_	GPIO_	MU	DSP_T	RTC_L
W	_J...	SDIO1	SDIO0	COM...	COM...	PI	VEN...	IT4	IT2	IN...	IN...	IN...	IN...		IE...	IT...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 FLEXCOMM11	FLEXCOMM 11 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
30 FLEXCOMM10	FLEXCOMM 10 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
29 FLEXCOMM9	FLEXCOMM 9 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
28 FLEXCOMM8	FLEXCOMM 8 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
27 SHA	Hash-AES wake-up. 0b - Disabled 1b - Enabled
26 PMIC	Wake-up from on-chip PMC or off-chip PMIC. 0b - Disabled 1b - Enabled
25 CASPER	CASPER co-processor wake-up. 0b - Disabled 1b - Enabled
24 POWERQUAD	POWERQUAD co-processor wake-up. 0b - Disabled 1b - Enabled
23 PUF	PUF wake-up. 0b - Disabled 1b - Enabled
22 DMAC1	DMA controller 1 wake-up. 0b - Disabled 1b - Enabled
21 USB_PHYDCD	USB PHY DCD interrupt wake-up

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled
20 —	Reserved
19 USB0_NEEDCLK	USB activity wake-up. 0b - Disabled 1b - Enabled
18-17 —	Reserved
16 SGPIO_INTB	Secure GPIO interrupt B wake-up. 0b - Disabled 1b - Enabled
15 SGPIO_INTA	Secure GPIO interrupt A wake-up. 0b - Disabled 1b - Enabled
14 SDIO1	SDIO0 wake-up. 0b - Disabled 1b - Enabled
13 SDIO0	SDIO0 wake-up. 0b - Disabled 1b - Enabled
12 FLEXCOMM7	FLEXCOMM 7 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
11 FLEXCOMM6	FLEXCOMM 6 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
10 FLEXSPI	Quad/octal SPI wake-up. 0b - Disabled 1b - Enabled
9	OS Event Timer wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Function
OS_EVENT_TIMER_WU	0b - Disabled 1b - Enabled
8 CT32BIT4	CTIMER 4 wake-up 0b - Disabled 1b - Enabled
7 CT32BIT2	CTIMER 2 wake-up 0b - Disabled 1b - Enabled
6 GPIO_INT0_IR_Q7	GPIO pin interrupt 7 wake-up. 0b - Disabled 1b - Enabled
5 GPIO_INT0_IR_Q6	GPIO pin interrupt 6 wake-up. 0b - Disabled 1b - Enabled
4 GPIO_INT0_IR_Q5	GPIO pin interrupt 5 wake-up. 0b - Disabled 1b - Enabled
3 GPIO_INT0_IR_Q4	GPIO pin interrupt 4 wake-up. 0b - Disabled 1b - Enabled
2 MU	Message Unit wake-up. 0b - Disabled 1b - Enabled
1 DSP_TIE_EXP_STATE1	DSP wake-up. 0b - No effect 1b - Sets the corresponding STARTEN1 bit
0 RTC_LITE0_WAKEUP	RTC wake-up. 0b - Disabled 1b - Enabled

4.5.1.56 Start Enable 2 (STARTEN2)

Offset

Register	Offset
STARTEN2	688h

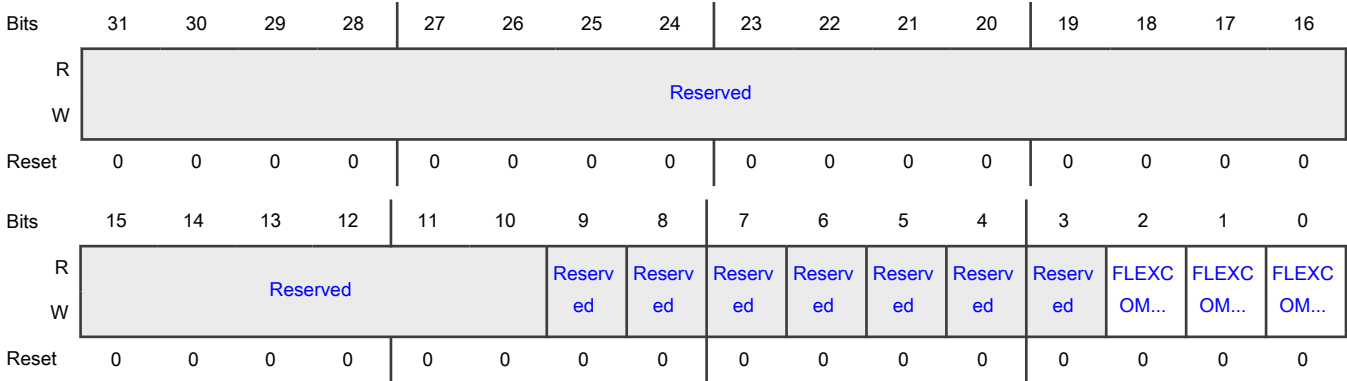
Function

This register enables an interrupt for wake-up from deep-sleep mode. Many peripherals serve the wake-up function. The peripheral must be sufficiently enabled to cause interrupts in the reduced power mode.

NOTE

Changes to the STARTEN2 registers be accomplished by using the related STARTEN_SET2 and STARTEN_CLR2 registers. This avoids any unintentional setting or clearing of other bits. The corresponding interrupts in the NVIC should be enabled.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 FLEXCOMM16	FlexComm 16 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
1 FLEXCOMM13	FlexComm 13 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled
0 FLEXCOMM12	FlexComm 12 peripheral interrupt wake-up. 0b - Disabled 1b - Enabled

4.5.1.57 Start Enable 0 Set (STARTEN0_SET)

Offset

Register	Offset
STARTEN0_SET	6A0h

Function

Writing a 1 to a bit position sets the corresponding position in STARTEN0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RNG	PMC	HWVAD0	SECUREVIOLATION	HYPERSVISOR	Reserved	DMIC0	ACMP	Reserved	ADC0	FLEXCOM0	FLEXCOM1	Reserved	Reserved	Reserved	Reserved
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved	CT32BIT3	SCT0	CT32BIT1	CT32BIT0	MRT0	UTICK0	GPIOIN0	GPIOIN1	GPIOIN2	GPIOIN3	GPIOIN4	GPIOIN5	DMAC0	WDT0
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 RNG	RNG wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
30 PMC	PMC wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
29 HWVAD0	Hardware VAD wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
28 SECUREVIOLATION	Secure Violation wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
27 HYPERVISOR	Hypervisor interrupt wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
26 —	Reserved
25 DMIC0	DMIC wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
24 ACMP	ACMP wake-up. 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the STARTEN0 Bit
23 —	Reserved
22 ADC0	ADC wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
21 FLEXCOMM15	FlexComm 15 (PMIC I2C) peripheral interrupt wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
20 FLEXCOMM14	FlexComm 14 (High Speed SPI) peripheral interrupt wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
19-14 —	FlexComm n peripheral interrupt wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
13 CT32BIT3	CTIMER 3 wake-up 0b - No effect 1b - Sets the STARTEN0 Bit
12 SCT0	SCTimer/PWM wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
11 CT32BIT1	CTIMER 1 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
10 CT32BIT0	CTIMER 0 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
9 MRT0	MRT wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
8	UTICK wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Function
UTICK0	0b - No effect 1b - Sets the STARTEN0 Bit
7 GPIO_INT0_IR Q3	GPIO pin interrupt 3 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
6 GPIO_INT0_IR Q2	GPIO pin interrupt 2 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
5 GPIO_INT0_IR Q1	GPIO pin interrupt 1 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
4 GPIO_INT0_IR Q0	GPIO pin interrupt 0 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
3 GPIO_INTB	Non-secure GPIO interrupt B wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
2 GPIO_INTA	Non-secure GPIO interrupt A wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
1 DMAC0	DMA controller 0 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
0 WDT0	Watchdog timer 0 wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit

4.5.1.58 Start Enable 1 Set (STARTEN1_SET)

Offset

Register	Offset
STARTEN1_SET	6A4h

Function

Writing a 1 to a bit position sets the corresponding position in STARTEN1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FLEX	FLEX	FLEX	FLEX	SHA	PMIC	CASP	POWE	PUF	DMAC	USB_	Reserv	USB0_	Reserved		SGPIO
W	COM...	COM...	COM...	COM...			ER	RQU...		1	PHY...	ed	NE...			_I...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SGPIO	SDIO1	SDIO0	FLEX	FLEX	FLEXS	OS_E	CT32B	CT32B	GPIO_	GPIO_	GPIO_	GPIO_	MU	DSP_T	RTC_L
W	_I...			COM...	COM...	PI	VEN...	IT4	IT2	IN...	IN...	IN...	IN...		IE...	IT...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 FLEXCOMM11	FLEXCOMM 11 peripheral interrupt wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
30 FLEXCOMM10	FLEXCOMM 10 peripheral interrupt wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
29 FLEXCOMM9	FLEXCOMM 9 peripheral interrupt wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
28 FLEXCOMM8	FLEXCOMM 8 peripheral interrupt wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
27 SHA	Hash-AES wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
26 PMIC	Wake-up from on-chip PMC or off-chip PMIC. 0b - No Effect 1b - Sets the STARTEN1 Bit
25 CASPER	CASPER co-processor wake-up. 0b - No Effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the STARTEN1 Bit
24 POWERQUAD	POWERQUAD co-processor wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
23 PUF	PUF wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
22 DMAC1	DMA controller 1 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
21 USB_PHYDCD	USB PHY DCD interrupt wake-up 0b - No Effect 1b - Sets the STARTEN1 Bit
20 —	Reserved
19 USB0_NEEDCLK	USB activity wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
18-17 —	Reserved
16 SGPIO_INTB	Secure GPIO interrupt B wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
15 SGPIO_INTA	Secure GPIO interrupt A wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
14 SDIO1	SDIO01 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
13 SDIO0	SDIO0 wake-up. 0b - No Effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the STARTEN1 Bit
12 FLEXCOMM7	FLEXCOMM7 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
11 FLEXCOMM6	FLEXCOMM6 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
10 FLEXSPI	Quad/octal SPI wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
9 OS_EVENT_TIMER_WU	OS Event Timer wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
8 CT32BIT4	CTIMER 4 wake-up 0b - No Effect 1b - Sets the STARTEN1 Bit
7 CT32BIT2	CTIMER 2 wake-up 0b - No Effect 1b - Sets the STARTEN1 Bit
6 GPIO_INT0_IR_Q7	GPIO pin interrupt 7 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
5 GPIO_INT0_IR_Q6	GPIO pin interrupt 6 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
4 GPIO_INT0_IR_Q5	GPIO pin interrupt 5 wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
3 GPIO_INT0_IR_Q4	Message Unit wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 MU	Message Unit wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit
1 DSP_TIE_EXP STATE1	DSP wake-up 0b - Disabled 1b - Enabled
0 RTC_LITE0_W AKEUP	RTC wake-up. 0b - No Effect 1b - Sets the STARTEN1 Bit

4.5.1.59 Start Enable 2 (STARTEN2_SET)

Offset

Register	Offset
STARTEN2_SET	6A8h

Function

Writing a 1 to a bit position in this register sets the corresponding position in STARTEN2 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserv	Reserv	Reserv	Reserv	Reserv	FLEXC	FLEXC	FLEXC
W									ed	ed	ed	ed	ed	OM...	OM...	OM...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-10 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 FLEXCOMM16	FlexComm16 interrupt wake-up 0b - No effect 1b - Sets the STARTEN2 bit
1 FLEXCOMM13	FlexComm 13 interrupt wake-up 0b - No effect 1b - Sets the STARTEN2 bit
0 FLEXCOMM12	FlexComm 12 interrupt wake-up 0b - No effect 1b - Sets the STARTEN2 bit

4.5.1.60 Start Enable 0 clear (STARTEN0_CLR)

Offset

Register	Offset
STARTEN0_CLR	6C0h

Function

Writing a 1 to a bit position in STARTEN0_CLR clears the corresponding position in STARTEN0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RNG	PMC	HWVA D0	SECU REV...	HYPE RVI...	Reserv ed	DMIC0	ACMP	Reserv ed	ADC0	FLEX COM...	FLEX COM...	Reserv ed	Reserv ed	Reserv ed	Reserv ed
W	W1C	W1C	W1C	W1C	W1C		W1C	W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	Reserv ed	CT32B IT3	SCT0	CT32B IT1	CT32B IT0	MRT0	UTICK 0	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	DMAC 0	WDT0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 RNG	RNG wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
30 PMC	PMC wake-up. 0b - No effect 1b - Sets the STARTEN0 Bit
29 HWVAD0	Hardware VAD wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
28 SECUREVIOLA TION	Secure Violation wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
27 HYPERVISOR	Hypervisor interrupt wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
26 —	Reserved
25 DMIC0	DMIC wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
24	ACMP wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ACMP	0b - No effect 1b - Clears the STARTEN0 Bit
23 —	Reserved
22 ADC0	ADC wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
21 FLEXCOMM15	FlexComm 15 (PMIC I2C) peripheral interrupt wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
20 FLEXCOMM14	FlexComm 14 (High Speed SPI) peripheral interrupt wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
19-14 —	FlexComm n peripheral interrupt wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
13 CT32BIT3	CTIMER 3 wake-up 0b - No effect 1b - Clears the STARTEN0 Bit
12 SCT0	SCTimer/PWM wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
11 CT32BIT1	CTIMER 1 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
10 CT32BIT0	CTIMER 0 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
9 MRT0	MRT wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 UTICK0	UTICK wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
7 GPIO_INT0_IR Q3	GPIO pin interrupt 3 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
6 GPIO_INT0_IR Q2	GPIO pin interrupt 2 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
5 GPIO_INT0_IR Q1	GPIO pin interrupt 1 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
4 GPIO_INT0_IR Q0	GPIO pin interrupt 0 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
3 GPIO_INTB	Non-secure GPIO interrupt B wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
2 GPIO_INTA	Non-secure GPIO interrupt A wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
1 DMAC0	DMA controller 0 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit
0 WDT0	Watchdog timer 0 wake-up. 0b - No effect 1b - Clears the STARTEN0 Bit

4.5.1.61 Start Enable 1 clear (STARTEN1_CLR)

Offset

Register	Offset
STARTEN1_CLR	6C4h

Function

Writing a 1 to a bit position in STARTEN0_CLR clears the corresponding position in STARTEN1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	SHA	PMIC	CASP ER	POWE RQU...	PUF	DMAC 1	USB_ PHY...	Reserv ed	USB0_ NE...	Reserved		SGPIO _I...
W					W1C	W1C	W1C	W1C	W1C	W1C	W1C		W1C			W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SGPIO _I...	SDIO1	SDIO0	FLEX COM...	FLEX COM...	FLEXS PI	OS_E VEN...	CT32B IT4	CT32B IT2	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	MU	Reserv ed	RTC_L IT...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C		W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 FLEXCOMM11	FLEXCOMM 11 peripheral interrupt wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
30 FLEXCOMM10	FLEXCOMM 10 peripheral interrupt wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
29 FLEXCOMM9	FLEXCOMM 9 peripheral interrupt wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
28 FLEXCOMM8	FLEXCOMM 8 peripheral interrupt wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
27 SHA	Hash-AES wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
26 PMIC	Wake-up from on-chip PMC or off-chip PMIC. 0b - No Effect 1b - Clears the STARTEN1 Bit
25	CASPER co-processor wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Function
CASPER	0b - No Effect 1b - Clears the STARTEN1 Bit
24 POWERQUAD	POWERQUAD co-processor wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
23 PUF	PUF wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
22 DMAC1	DMA controller 1 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
21 USB_PHYDCD	USB PHY DCD interrupt wake-up 0b - No Effect 1b - Clears the STARTEN1 Bit
20 —	Reserved
19 USB0_NEEDCLK	USB activity wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
18-17 —	Reserved
16 SGPIO_INTB	Secure GPIO interrupt B wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
15 SGPIO_INTA	Secure GPIO interrupt A wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
14 SDIO1	SDIO01 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
13	SDIO0 wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Function
SDIO0	0b - No Effect 1b - Clears the STARTEN1 Bit
12 FLEXCOMM7	FLEXCOMM7 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
11 FLEXCOMM6	FLEXCOMM6 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
10 FLEXSPI	Quad/octal SPI wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
9 OS_EVENT_TIMER_WU	OS Event Timer wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
8 CT32BIT4	CTIMER 4 wake-up 0b - No Effect 1b - Clears the STARTEN1 Bit
7 CT32BIT2	CTIMER 2 wake-up 0b - No Effect 1b - Clears the STARTEN1 Bit
6 GPIO_INT0_IRQ7	GPIO pin interrupt 7 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
5 GPIO_INT0_IRQ6	GPIO pin interrupt 6 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
4 GPIO_INT0_IRQ5	GPIO pin interrupt 5 wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
3 GPIO_INT0_IRQ4	Message Unit wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 MU	Message Unit wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit
1 —	Reserved
0 RTC_LITE0_W AKEUP	RTC wake-up. 0b - No Effect 1b - Clears the STARTEN1 Bit

4.5.1.62 Start Enable 2 (STARTEN2_CLR)

Offset

Register	Offset
STARTEN2_CLR	6C8h

Function

Writing a 1 to a bit position in this register clears the corresponding position in STARTEN2 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserv	Reserv	Reserv	Reserv	Reserv	FLEXC	FLEXC	FLEXC
W									ed	ed	ed	ed	ed	OM...	OM...	OM...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	W1C	W1C	W1C

Fields

Field	Function
31-10 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 FLEXCOMM16	FlexComm 16 interrupt wake-up 0b - No effect 1b - Clears the STARTEN2 bit
1 FLEXCOMM13	FlexComm 13 interrupt wake-up 0b - No effect 1b - Clears the STARTEN2 bit
0 FLEXCOMM12	FlexComm 12 interrupt wake-up 0b - No effect 1b - Clears the STARTEN2 bit

4.5.1.63 FRO Safety (FROSAFETY)

Offset

Register	Offset
FROSAFETY	704h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FROSAFETY															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FROSAFETY															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 FROSAFETY	FRO Safety

4.5.1.64 Main Clock Safety (MAINCLKSAFETY)**Offset**

Register	Offset
MAINCLKSAFETY	710h

Function

This register controls the Main Clock turn on delay from Deep Sleep wake up.

NOTE

The values written to this register should not be changed. The main clock turn on delay is calculated by the SDK Power Library.

Diagram

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

Fields

Field	Function
31-16 —	Reserved
15-0 DELAY	Main Clock turn on delay for Deep Sleep wake up

4.5.1.65 Hardware Wake (HWWAKE)

Offset

Register	Offset
HWWAKE	780h

Function

The primary use of the hardware Wake-up control register is to provide the possibility for some peripherals to have DMA service during deep-sleep mode without waking up entire device. These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts.

When a peripheral that is able to operate (at least in some modes) during deep-sleep reaches its programmed FIFO threshold, it can cause bus clocks to be temporarily enabled. During that time, DMA can move data from the peripheral to memory or from memory to the peripheral. When DMA completes, full deep-sleep mode will be resumed.

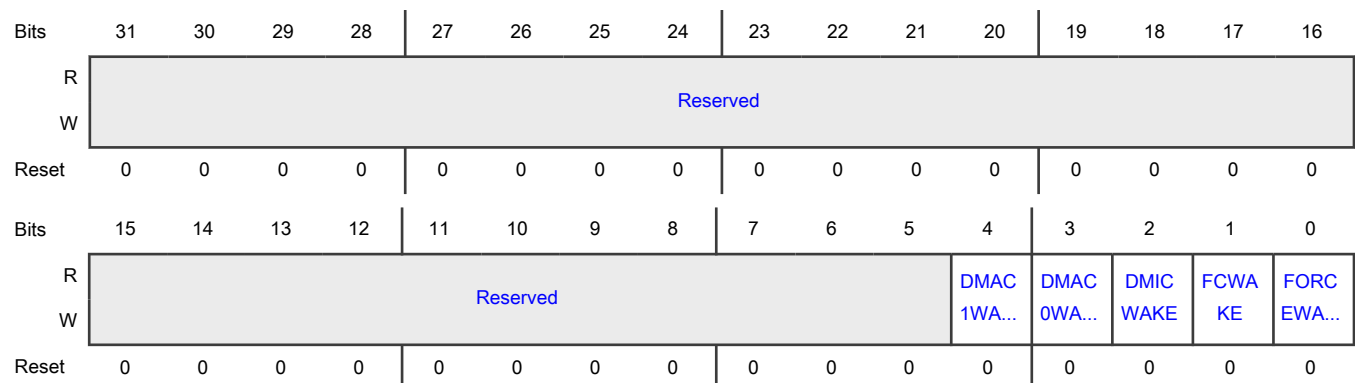
See section, Entering Deep Sleep Mode in Power Management Controller (PMC) chapter.

NOTE

HWWAKE functionality requires AUTOCLKGATEOVERRIDE0 clock gating enabled, otherwise continuous clock will not allow entry into low power mode.

See "Entering Deep Sleep Mode" section in Power Management Controller chapter for more details.

Diagram



Fields

Field	Function
31-5 —	Reserved
4 DMAC1WAKE	Wake for DMAC1. When 1, DMAC1 being busy will cause peripheral clocking to remain running until DMAC1 completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMAC1 has completed its related activity.
3 DMAC0WAKE	Wake for DMAC0. When 1, DMAC0 being busy will cause peripheral clocking to remain running until DMAC0 completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMAC0 has completed its related activity.
2 DMICWAKE	Wake for Digital Microphone. When 1, the digital microphone input FIFO reaching the level specified by TRIGLVL of either channel will cause peripheral clocking to be enabled temporarily while the DMIC FIFO level is at or above TRIGLVL. This allows DMA to become active to move data out of the DMIC FIFO. Generally, WAKEDMA should also be enabled when WAKEDMIC is enabled.
1 FCWAKE	Wake for FlexComm Interfaces. When 1, any Flexcomm Interface FIFO reaching the level specified by its own FIFO level will cause peripheral clocking to be enabled temporarily while a Flexcomm Interface FIFO level is at or beyond the specified level. This allows DMA to become active to move data out of the Flexcomm Interface FIFO. Generally, WAKEDMA should also be enabled when FCWAKE is enabled. This feature uses the TXLVL flag in the FIFOSTAT register if the bit WAKETX in the FIFOCFG register is set, or the RXLVL flag in FIFOSTAT if WAKERX in FIFOCFG is set.
0 FORCEWAKE	Force peripheral clocking to stay on during deep-sleep mode. When 1, clocking to peripherals is prevented from being shut down when the CPU enters deep-sleep mode. This is intended to allow a coprocessor to continue operating while the main CPU(s) are shut down.

4.5.1.66 Temperature Sensor Control (TEMPSENSORCTL)**Offset**

Register	Offset
TEMPSENSORCTL	E0Ch

Function

This register is used to select temperature sensor used by ADC.

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 TSSRC	Temperature Sensor Source 0b - ADC Built-in Temperature Sensor 1b - PMC Temperature Sensor

4.5.1.67 Boot State Lock (BOOTSTATELOCK)**Offset**

Register	Offset
BOOTSTATELOCK	E40h

Function

It stores the lockable boot state. The bits are only settable, cannot be cleared except reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															BOOT STA...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-2 —	Reserved
1 BOOTSTATEH MACLOCK	Boot State HMA Lockout 0b - BOOTSTATEHMAC[0:7] can be changed 1b - BOOTSTATEHMAC[0:7] cannot be changed
0 BOOTSTATES EEDLOCK	Boot State Seed Lockout 0b - BOOTSTATESEED[0:7] can be changed 1b - BOOTSTATESEED[0:7] cannot be changed

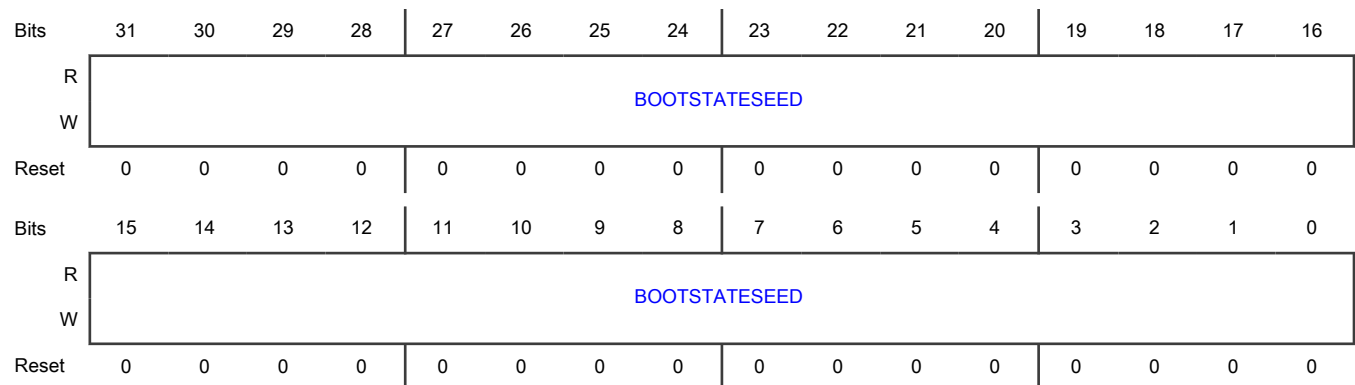
4.5.1.68 Boot State Seed (BOOTSTATESEED0 - BOOTSTATESEED7)**Offset**

For a = 0 to 7:

Register	Offset
BOOTSTATESEEDa	E50h + (a × 4h)

Function

It stores the lockable boot state.

Diagram**Fields**

Field	Function
31-0 BOOTSTATES EED	BOOTSTATESEED[0:7] A 256-bit random number set by boot ROM on each restart. This number is used as key for computing HMAC of the boot state.

4.5.1.69 HMAC of boot state used for attestation. (BOOTSTATEHMAC0 - BOOTSTATEHMAC7)

Offset

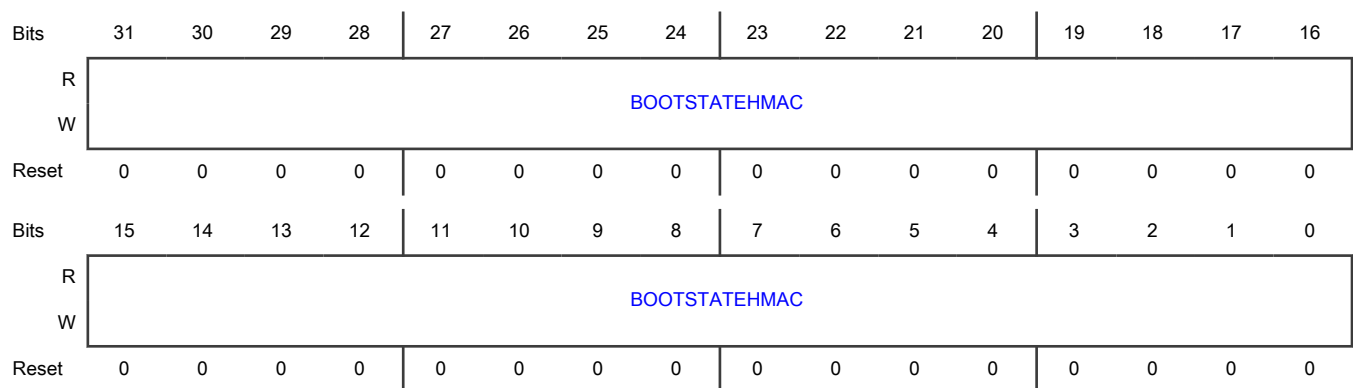
For a = 0 to 7:

Register	Offset
BOOTSTATEHMACa	E70h + (a × 4h)

Function

It stores the lockable boot state.

Diagram



Fields

Field	Function
31-0	BOOTSTATEHMAC[0:7]
BOOTSTATEHMAC	The boot state is defined as SHA2-256 bit digest of following data in order - - User image including certificate block - Boot configuration and security epoch (OTP words 95 to 104, 128 to 147). Optionally include key store data (OTP words 106 to 127).

4.5.1.70 FLEXSPI0 Pad Control (FLEXSPI0PADCTL)

Offset

Register	Offset
FLEXSPI0PADCTL	EF0h

Function

This register controls the compensation circuit for the FlexSPI0 IOs. This compensation circuit enables those IOs to operate within expected performance even having process, voltage and temperature variations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							COMP OK	NASRCP				NASRCN			
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				COMP EN	COMP TQ	FREE ZE	FASTF RZ	RASRCP_3_0				RASRCN_3_0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24 COMPOK	FLEXSPI0 Pad Compensation Circuit Status
23-20 NASRCP	FLEXSPI0 Pad Compensation Circuit Status
19-16 NASRCN	FLEXSPI0 Pad Compensation Circuit Status
15-12 —	Reserved
11 COMPEN	Drives FLEXSPI0 Pad Compensation Circuit
10 COMPTQ	Drives FLEXSPI0 Pad Compensation Circuit
9 FREEZE	Drives FLEXSPI0 Pad Compensation Circuit
8 FASTFRZ	Drives FLEXSPI0 Pad Compensation Circuit
7-4 RASRCP_3_0	Drives FLEXSPI0 Pad Compensation Circuit

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 RASRCN_3_0	Drives FLEXSPI0 Pad Compensation Circuit

4.5.1.71 FLEXSPI1 Pad Control (FLEXSPI1PADCTL)

Offset

Register	Offset
FLEXSPI1PADCTL	EF4h

Function

This register controls the compensation circuit for the FlexSPI0 IOs. This compensation circuit enables those IOs to operate within expected performance even having process, voltage and temperature variations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								COMP OK	NASRCP_3_0				NASRCN_3_0		
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				COMP EN	COMP TQ	FREE ZE	FASTF RZ	RASRCP_3_0				RASRCN_3_0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24 COMPOK	FLEXSPI1 Pad Compensation Circuit Status
23-20 NASRCP_3_0	FLEXSPI1 Pad Compensation Circuit Status
19-16	FLEXSPI1 Pad Compensation Circuit Status

Table continues on the next page...

Table continued from the previous page...

Field	Function
NASRCN_3_0	
15-12 —	Reserved
11 COMPEN	Drives FLEXSPI1 Pad Compensation Circuit
10 COMPTQ	Drives FLEXSPI1 Pad Compensation Circuit
9 FREEZE	Drives FLEXSPI1 Pad Compensation Circuit
8 FASTFRZ	Drives FLEXSPI1 Pad Compensation Circuit
7-4 RASRCP_3_0	Drives FLEXSPI1 Pad Compensation Circuit
3-0 RASRCN_3_0	Drives FLEXSPI1 Pad Compensation Circuit

4.5.1.72 SDIO0 Pad Control (SDIO0PADCTL)

Offset

Register	Offset
SDIO0PADCTL	EF8h

Function

This register controls the compensation circuit for the SDIO IOs. This compensation circuit enables those IOs to operate within expected performance even having process, voltage and temperature variations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							COMP OK	NASRCP				NASRCN			
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				COMP EN	COMP TQ	FREE ZE	FASTF RZ	RASRCP				RASRCN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24 COMPOK	SDIO0 Pad Compensation Circuit Status
23-20 NASRCP	SDIO0 Pad Compensation Circuit Status
19-16 NASRCN	SDIO0 Pad Compensation Circuit Status
15-12 —	Reserved
11 COMPEN	Drives SDIO0 Pad Compensation Circuit
10 COMPTQ	Drives SDIO0 Pad Compensation Circuit
9 FREEZE	Drives SDIO0 Pad Compensation Circuit
8 FASTFRZ	Drives SDIO0 Pad Compensation Circuit
7-4 RASRCP	Drives SDIO0 Pad Compensation Circuit

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 RASRCN	Drives SDIO0 Pad Compensation Circuit

4.5.1.73 SDIO1 Pad Control (SDIO1PADCTL)

Offset

Register	Offset
SDIO1PADCTL	EFCh

Function

This register controls the compensation circuit for the SDIO IOs. This compensation circuit enables those IOs to operate within expected performance even having process, voltage and temperature variations.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							COMP OK	NASRCP				NASRCN			
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				COMP EN	COMP TQ	FREE ZE	FASTF RZ	RASRCP				RASRCN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-25 —	Reserved
24 COMPOK	SDIO1 Pad Compensation Circuit Status
23-20 NASRCP	SDIO1 Pad Compensation Circuit Status
19-16	SDIO1 Pad Compensation Circuit Status

Table continues on the next page...

Table continued from the previous page...

Field	Function
NASRCN	
15-12 —	Reserved
11 COMPEN	Drives SDIO1 Pad Compensation Circuit
10 COMPTQ	Drives SDIO1 Pad Compensation Circuit
9 FREEZE	Drives SDIO1 Pad Compensation Circuit
8 FASTFRZ	Drives SDIO1 Pad Compensation Circuit
7-4 RASRCP	Drives SDIO1 Pad Compensation Circuit
3-0 RASRCN	Drives SDIO1 Pad Compensation Circuit

4.5.1.74 Compound Device Identifier (CDI) (DICEHWREGn)

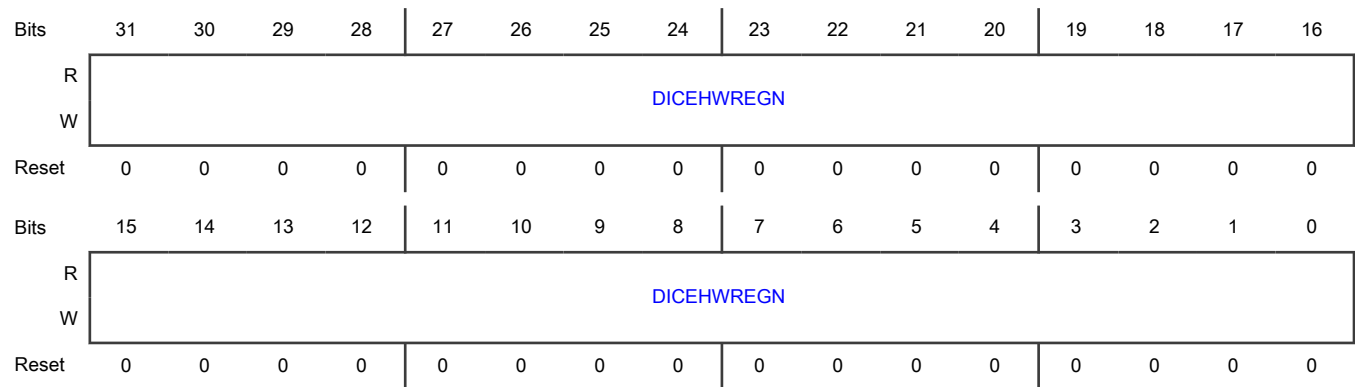
Offset

Register	Offset
DICEHWREGn	F00h

Function

This 256 bit identifier is derived as per Trusted Computing Group's (TCG) Device Identifier Composition Engine (DICE) specification (<https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/>), from a 256-bit Unique Device Secret (UDS) and the digest (a condensed cryptographic representation) of OEM's root of trust code (aka first mutable user code). The CDI is unique not only to the device, but to the combination of the device, the cryptographic identity of the device's First Mutable Code and, optionally, secure boot configuration data stored in OTP.

One of the possible uses of the Compound Device Identifier is to attest to the trustworthiness of an embedded device. The CDI could also be used to generate implicit device identity, an asymmetric device key pair that serves as a long-term identity for the device. Since the hash of OEM root of trust code is used to derive the CDI, any tampering of this critical code (injection of malware), will result in a new Device Identity and an unrecoverable loss of the existing Device Identity and any derived secrets.

Diagram**Fields**

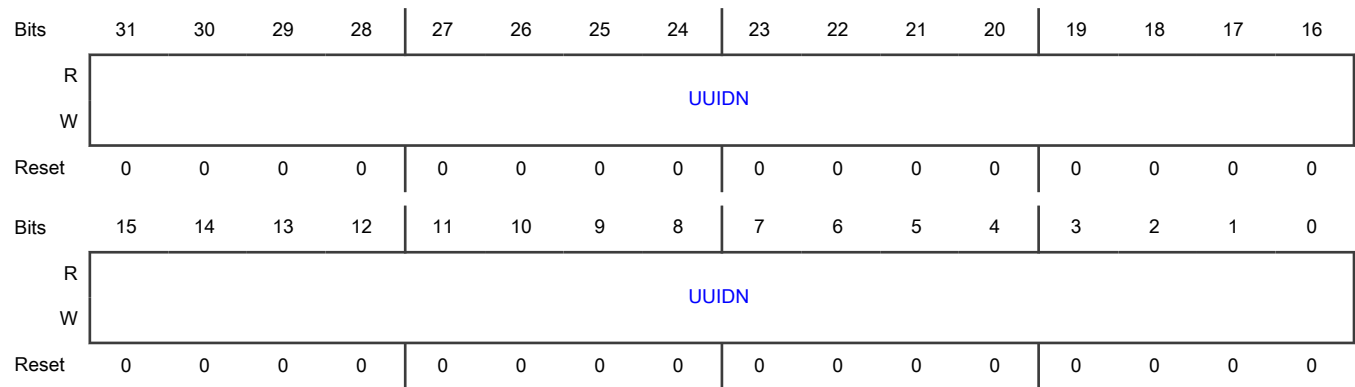
Field	Function
31-0 DICEHWREGN	DICE General Purpose 32-Bit Data Register

4.5.1.75 UUID (UUID0 - UUID3)**Offset**

Register	Offset
UUID0	F50h
UUID1	F54h
UUID2	F58h
UUID3	F5Ch

Function

This register contains 128 bits long Universally Unique Identifier(UUID), also known as GUIDs (Globally Unique Identifier). The UUID is generated as per IETF specification RFC4122 version 5 (<https://tools.ietf.org/html/rfc4122>), to ensure uniqueness across space and time.

Diagram**Fields**

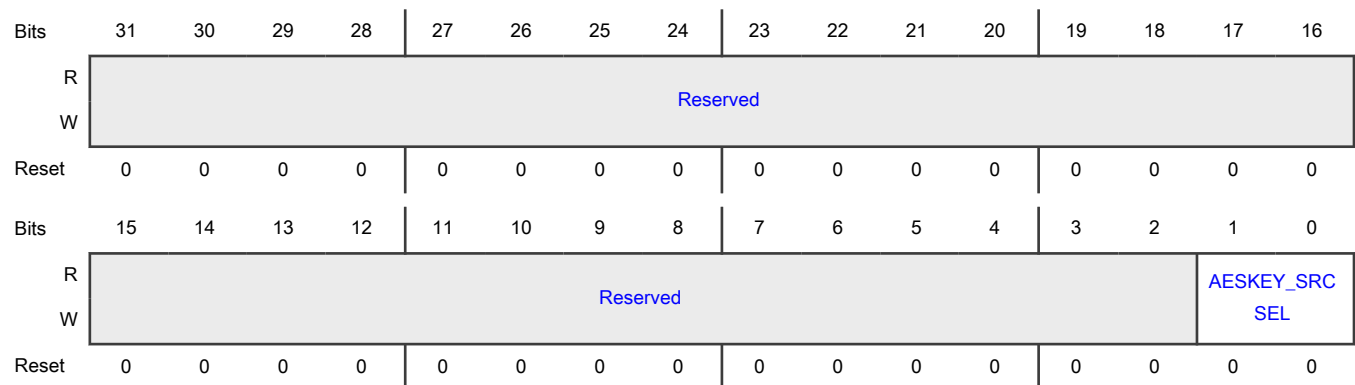
Field	Function
31-0 UUIDN	UUIDn 32-Bit Data Register

4.5.1.76 AES Key Source Select (AESKEY_SRCSEL)**Offset**

Register	Offset
AESKEY_SRCSEL	F80h

Function

This register selects the key source for the AES 256 module.

Diagram

Fields

Field	Function
31-2 —	Reserved
1-0 AESKEY_SRCSEL	AES Key Source Select 00b - PUF 01b - PUF 10b - OTP 11b - PUF

4.5.1.77 OTFAD Key Source Select (OTFADKEY_SRCSEL)**Offset**

Register	Offset
OTFADKEY_SRCSEL	F84h

Function

This register selects an OTFAD key.

Diagram

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

Fields

Field	Function
31-2 —	Reserved
1-0	OTFAD Key Source Select

Table continues on the next page...

Table continued from the previous page...

Field	Function
OTFADKEY_S RCSEL	00b - PUF 01b - PUF 10b - OTP 11b - PUF

4.5.1.78 HASH Hardware Key Disable (HASHHWKEYDISABLE)

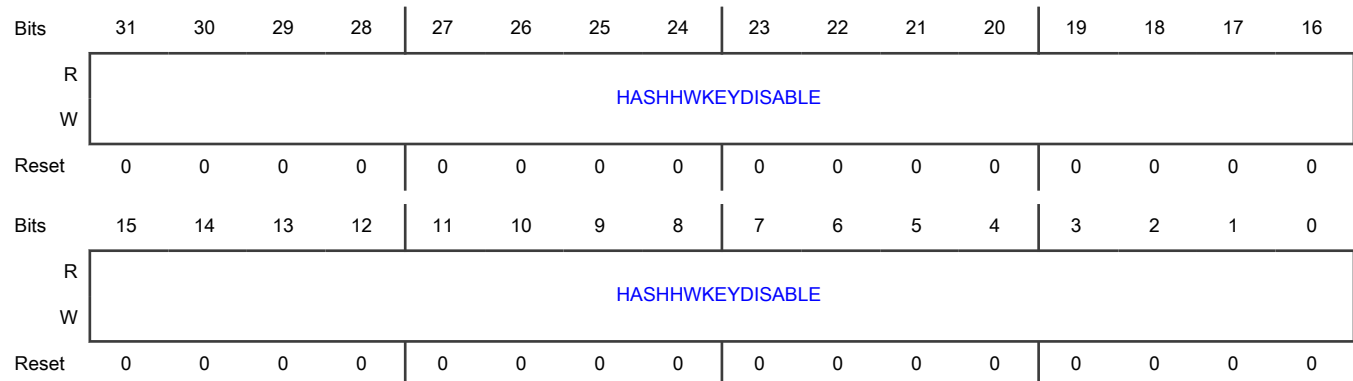
Offset

Register	Offset
HASHHWKEYDISABLE	F88h

Function

This register control the access to AES keys delivered through secret HW bus from PUF and OTP to AES engine.

Diagram



Fields

Field	Function
31-0 HASHHWKEYD ISABLE	HASH Hardware Key Disable If value = 0xC33CA55A, then software running on CM33 can use the secret keys. For all other values, secret keys are only usable when software on CM33 is running at security level. In other words, if value = 0xC33CA55A, then HASH CRYPTCFG register bit 7 (AESSECRET) is changeable when CM33 is running at security level less than 3. Security level programmed for HASHCRYPT engine in Peripheral Protection Control (PPC) register in secure AHB bus control restricts access to HASHCRYPT registers.

4.5.1.79 Debug Lock Enable (DBG_LOCKEN)

Offset

Register	Offset
DBG_LOCKEN	FA0h

Function

NOTE

The DBG_LOCKEN register can be read at any time. If any value is written other than the reset value of 0x0000 000A, the register will be locked from further writes but can still be read. Once the DBG_LOCKEN register is locked, it will disable the write capability to the DBG_FEATURES, DBG_FEATURES_DP, CS_PROTTEST, CS_PROTCPU0, CS_PROTCPU1, and DBG_AUTH_SCRATCH registers.

Diagram

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

Fields

Field	Function
31-4 —	Reserved
3-0 DBG_LOCKEN	Debug Write Lock the following registers DBG_FEATURES DBG_FEATURES_DP CS_PROTTEST CS_PROTCPU0 CS_PROTCPU1 DBG_AUTH_SCRATCH 1010: Write Enabled (Unlocked) Any other value other than 1010: Write Disabled (Locked)

4.5.1.80 Debug Features (DBG_FEATURES)

Offset

Register	Offset
DBG_FEATURES	FA4h

Function

This register controls Cortex-M33 (CPU0) debug features. Invasive debug, non-invasive debug, secure invasive debug, secure non-invasive debug are described in Arm documentation. To enable any of the debug features, the DBG_FEATURES register setting for the respective debug feature needs to be set to '10'. In addition, the respective bits in the DBG_FEATURES_DP register must be set to '10'.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								SPNIDEN1		SPIDEN1		NIDEN1		DBGEN1	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Fields

Field	Function
31-8 —	Reserved
7-6 SPNIDEN1	CM33 SPNIDEN Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled
5-4 SPIDEN1	CM33 SPID Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 NIDEN1	CM33 NID Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled
1-0 DBGEN1	CM33 Debug Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled

4.5.1.81 Debug Features Duplicate (DBG_FEATURES_DP)

Offset

Register	Offset
DBG_FEATURES_DP	FA8h

Function

This register controls Cortex-M33 (CPU0) debug features. It is a duplicate of the Debug Features register. To enable any of the debug features, the DBG_FEATURES register setting for the respective debug feature needs to be set to '10'. In addition, the respective bits in the DBG_FEATURES_DP register must be set to '10'.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								SPNIDEN1		SPIDEN1		NIDEN1		DBGEN1	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Fields

Field	Function
31-8 —	Reserved
7-6 SPNIDEN1	CM33 SPNIDEN Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled
5-4 SPIDEN1	CM33 SPID Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled
3-2 NIDEN1	CM33 NID Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled
1-0 DBGEN1	CM33 Debug Enable Control 00b - Disabled 01b - Disabled 10b - Enabled 11b - Disabled

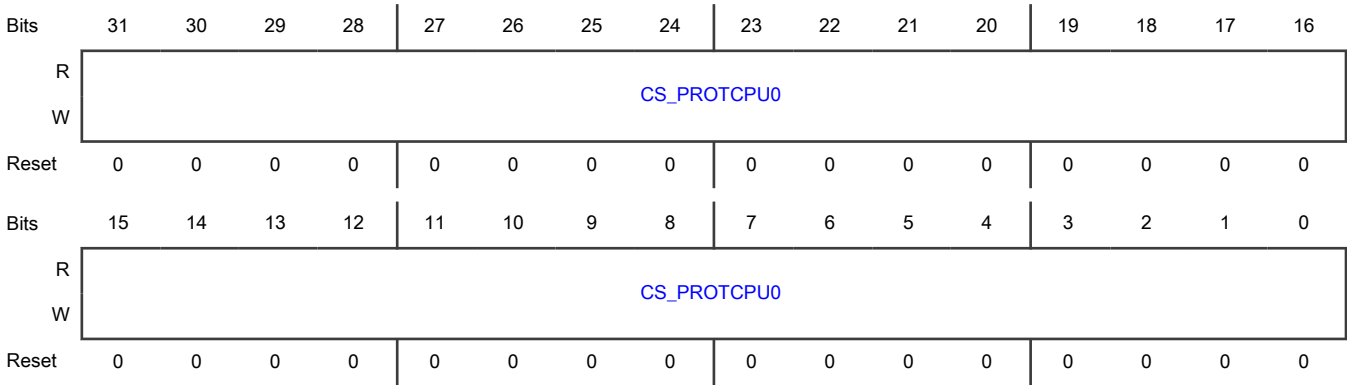
4.5.1.82 Code Security for CPU0 (CS_PROTCPU0)**Offset**

Register	Offset
CS_PROTCPU0	FB4h

Function

This register is used by the Boot Code during DEBUG authentication mechanism to enable debug access port for CPU0 (Cortex-M33).

Diagram



Fields

Field	Function
31-0 CS_PROTCPU 0	Controls M33 AP Enable Magic Value: 0x1234 5678

4.5.1.83 Code Security for CPU1 (CS_PROTCPU1)

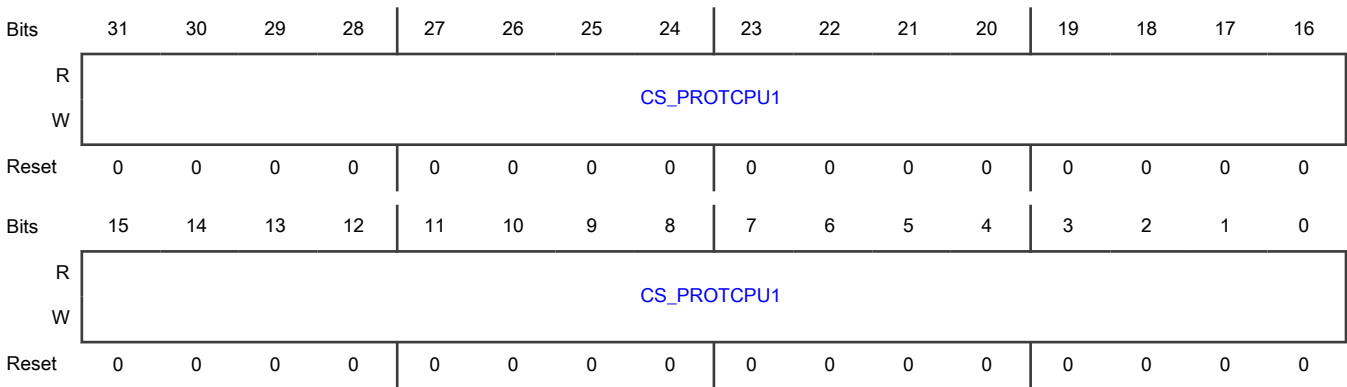
Offset

Register	Offset
CS_PROTCPU1	FB8h

Function

This register is used by the Boot Code during DEBUG authentication mechanism to enable debug access port for CPU1(DSP).

Diagram

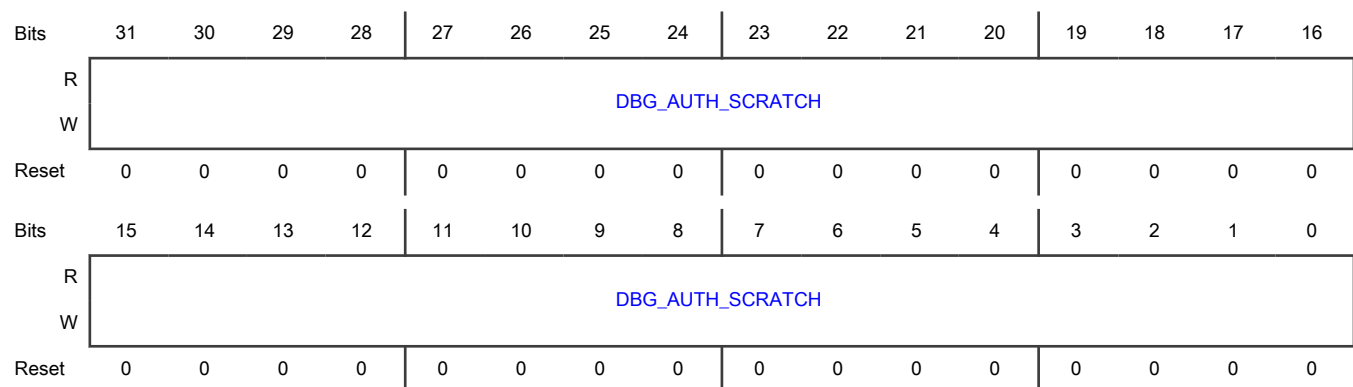


Fields

Field	Function
31-0 CS_PROTCPU 1	Controls AP Enable Magic Value: 0x1234 5678

4.5.1.84 Debug authorization scratch (DBG_AUTH_SCRATCH)**Offset**

Register	Offset
DBG_AUTH_SCRATCH	FC0h

Diagram**Fields**

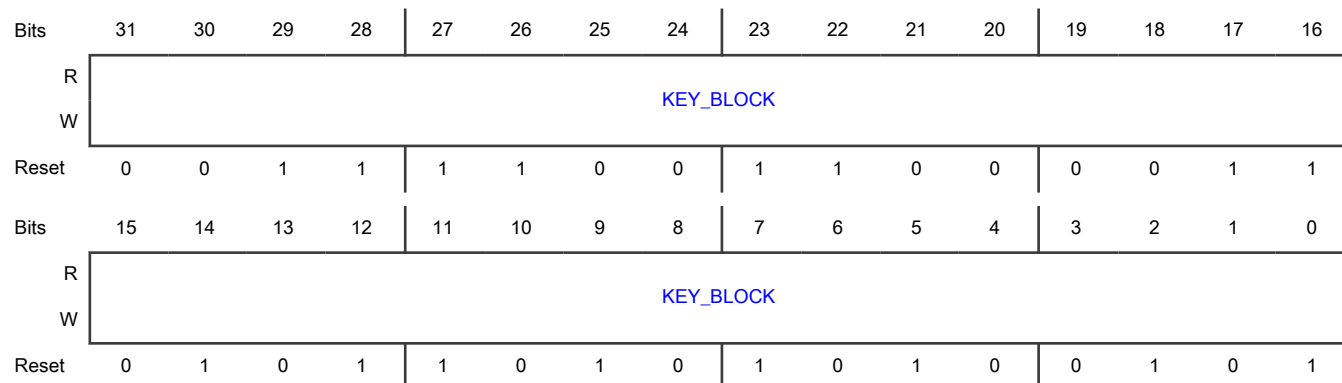
Field	Function
31-0 DBG_AUTH_S CRATCH	Debug authorization scratch register for S/W.

4.5.1.85 Key block (KEY_BLOCK)**Offset**

Register	Offset
KEY_BLOCK	FD0h

Function

The KEY_BLOCK register can be read at any time. If any value is written other than the reset value of 0x3CC3_5AA5, the register will be locked from further writes but can still be read.

Diagram**Fields**

Field	Function
31-0 KEY_BLOCK	PUF key and data output Any value other than 0x3CC35AA5 will block the PUF key and data output. This includes the software key/data output and the hardware key/data output to AES and OTFAD. NOTE If the OTP bit (syscon_fa_mode_en) is set, the PUF key/data output will also be blocked (same effect as the KEY_BLOCK register value not equal to 0x3CC35AA5)

4.5.2 SYSCTL1 register descriptions**4.5.2.1 SYSCTL1 memory map**

SYSCTL1 base address: 4002_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Update Clock Lockout (UPDATELCKOUT)	32	RW	0000_0000h
10h	MCLK direction control (MCLKPINDIR)	32	RW	0000_0000h
30h	DSP NMI source selection (DSPNMISRCSEL)	32	RW	0000_0005h
40h - 74h	Flexcomm control selection (FC0CTRLSEL - FC13CTRLSEL)	32	RW	0000_0000h
80h - 84h	Shared control set (SHAREDCTRLSET0 - SHAREDCTRLSET1)	32	RW	0000_0000h
200h	RX Event Pulse Generator (RXEVPULSEGEN)	32	RW	0000_0000h

4.5.2.2 Update Clock Lockout (UPDATELCKOUT)

Offset

Register	Offset
UPDATELCKOUT	0h

Function

It prevents write access to all other registers in the module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															UPDA TEL...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 UPDATELCKOUT	Update Clock Lockout 0b - Normal Mode 1b - Protected Mode. Cannot be written to. Currently this register does not lock anything

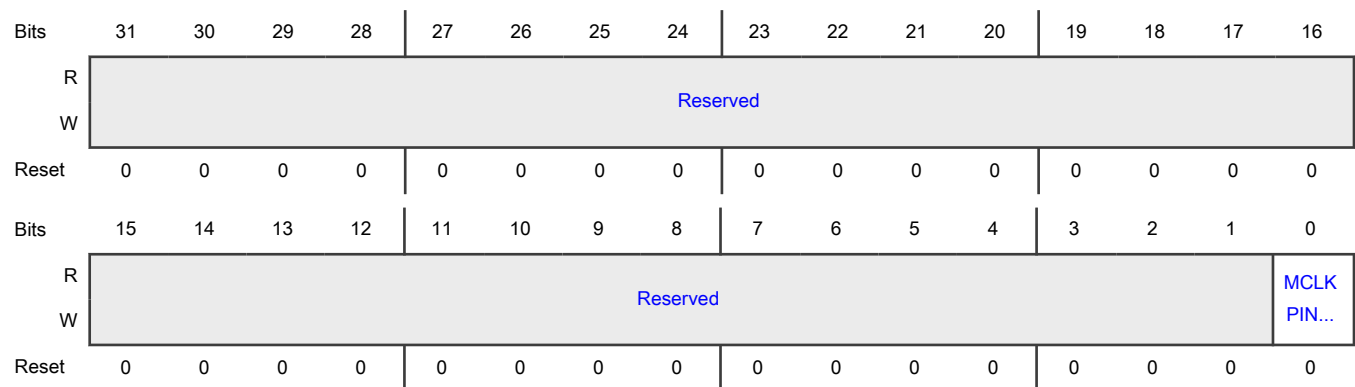
4.5.2.3 MCLK direction control (MCLKPINDIR)

Offset

Register	Offset
MCLKPINDIR	10h

Function

This register controls the direction of the pin associated with MCLK when MCLK is the selected function on that pin.

Diagram**Fields**

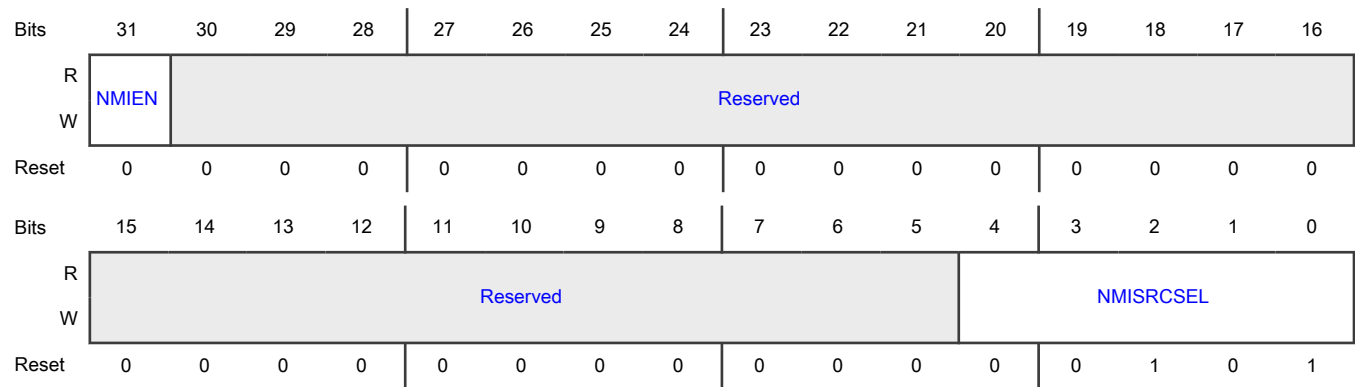
Field	Function
31-1 —	Reserved
0 MCLKPINDIR	MCLK direction control Selects the Pin Direction of MCLK: 0b - I2S MCLK is in input direction 1b - I2S MCLK is in the output direction

4.5.2.4 DSP NMI source selection (DSPNMISRCSEL)**Offset**

Register	Offset
DSPNMISRCSEL	30h

Function

It selects one of the DSP interrupt sources as the NMI source. To change the interrupt source for the NMI, the interrupt must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the NMISRCSEL bits and re-enable the interrupt by setting NMIEN.

Diagram**Fields**

Field	Function
31 NMIEN	NMI Enable 0b - Disable NMI Interrupt 1b - Enable NMI Interrupt
30-5 —	Reserved
4-0 NMISRCSEL	DSP NMI source selection Select one of the DSP interrupt sources as the NMI source. See DSP Interrupt Slot Table for Interrupt Slot Numbers. Writing a value of 0, 1, 2, 3 or 4 has no effect.

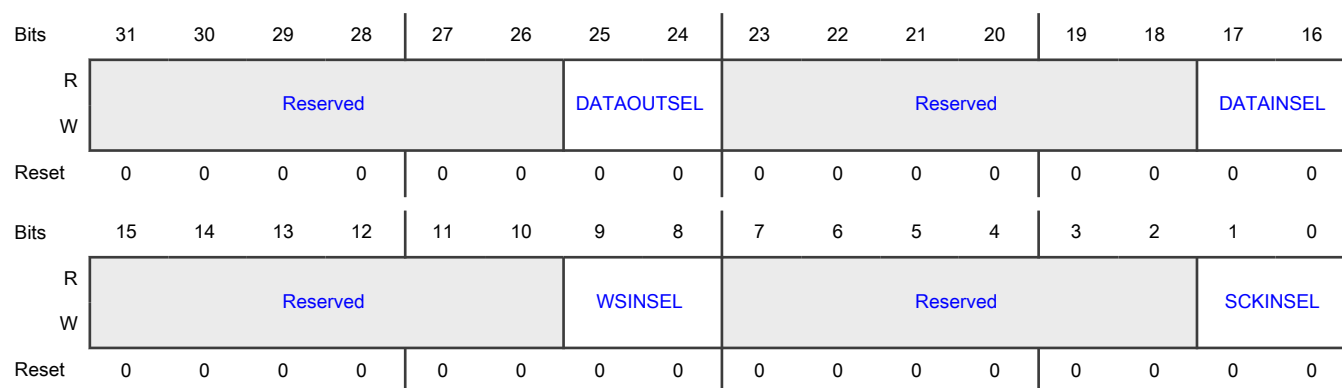
4.5.2.5 Flexcomm control selection (FC0CTRLSEL - FC13CTRLSEL)**Offset**

For a = 0 to 13:

Register	Offset
FCaCTRLSEL	40h + (a × 4h)

Function

These registers select the SCK, WS, DATA input, and DATA output signal source for each Flexcomm Interface. They allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. Please refer [I2S Signal Sharing](#).

Diagram**Fields**

Field	Function
31-26 —	Reserved
25-24 DATAOUTSEL	DATA OUT Select 00b - Original FLEXCOMM I2S signals 01b - Shared Set0 I2S signals 10b - Shared Set1 I2S signals 11b - Reserved
23-18 —	Reserved
17-16 DATAINSEL	DATA IN Select 00b - Original FLEXCOMM I2S signals 01b - Shared Set0 I2S signals 10b - Shared Set1 I2S signals 11b - Reserved
15-10 —	Reserved
9-8 WSINSEL	SCK IN Select 00b - Original FLEXCOMM I2S signals 01b - Shared Set0 I2S signals 10b - Shared Set1 I2S signals 11b - Reserved
7-2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
1-0 SCKINSEL	SCK IN Select 00b - Original FLEXCOMM I2S signals 01b - Shared Set0 I2S signals 10b - Shared Set1 I2S signals 11b - Reserved

4.5.2.6 Shared control set (SHAREDCTRLSET0 - SHAREDCTRLSET1)

Offset

Register	Offset
SHAREDCTRLSET0	80h
SHAREDCTRLSET1	84h

Function

These registers allow more than one on-chip I2S interface to be connected to some of the same pins without external board wiring. Please refer [I2S Signal Sharing](#).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								FC7D	FC6D	FC5D	FC4D	FC3D	FC2D	FC1D	FC0D
W									ATA...	ATA...	ATA...	ATA...	ATA...	ATA...	ATA...	ATA...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				SHAREDCTRLSEL				Reserv	SHAREDWSEL				Reserv	SHAREDSCSEL	
W									ed					ed		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24	Reserved
—	
23	FLEXCOMM7 DATAOUT Output Enable:

Table continues on the next page...

Table continued from the previous page...

Field	Function
FC7DATAOUT EN	0b - Input 1b - Output
22 FC6DATAOUT EN	FLEXCOMM6 DATAOUT Output Enable: 0b - Input 1b - Output
21 FC5DATAOUT EN	FLEXCOMM5 DATAOUT Output Enable: 0b - Input 1b - Output
20 FC4DATAOUT EN	FLEXCOMM4 DATAOUT Output Enable: 0b - Input 1b - Output
19 FC3DATAOUT EN	FLEXCOMM3 DATAOUT Output Enable: 0b - Input 1b - Output
18 FC2DATAOUT EN	FLEXCOMM2 DATAOUT Output Enable: 0b - Input 1b - Output
17 FC1DATAOUT EN	FLEXCOMM1 DATAOUT Output Enable: 0b - Input 1b - Output
16 FC0DATAOUT EN	FLEXCOMM0 DATAOUT Output Enable: 0b - Input 1b - Output
15-11 —	Reserved
10-8 SHARED DATA SEL	Shared DATA Select: 000b - FLEXCOMM0 001b - FLEXCOMM1 010b - FLEXCOMM2 011b - FLEXCOMM3 100b - FLEXCOMM4

Table continues on the next page...

Table continued from the previous page...

Field	Function
	101b - FLEXCOMM5 110b - FLEXCOMM6 111b - FLEXCOMM7
7 —	Reserved
6-4 SHAREDWSSEL	Shared WS Select: 000b - FLEXCOMM0 001b - FLEXCOMM1 010b - FLEXCOMM2 011b - FLEXCOMM3 100b - FLEXCOMM4 101b - FLEXCOMM5 110b - FLEXCOMM6 111b - FLEXCOMM7
3 —	Reserved
2-0 SHAREDSCSEL	Shared SCK Select 000b - FLEXCOMM0 001b - FLEXCOMM1 010b - FLEXCOMM2 011b - FLEXCOMM3 100b - FLEXCOMM4 101b - FLEXCOMM5 110b - FLEXCOMM6 111b - FLEXCOMM7

4.5.2.7 RX Event Pulse Generator (RXEVPULSEGEN)

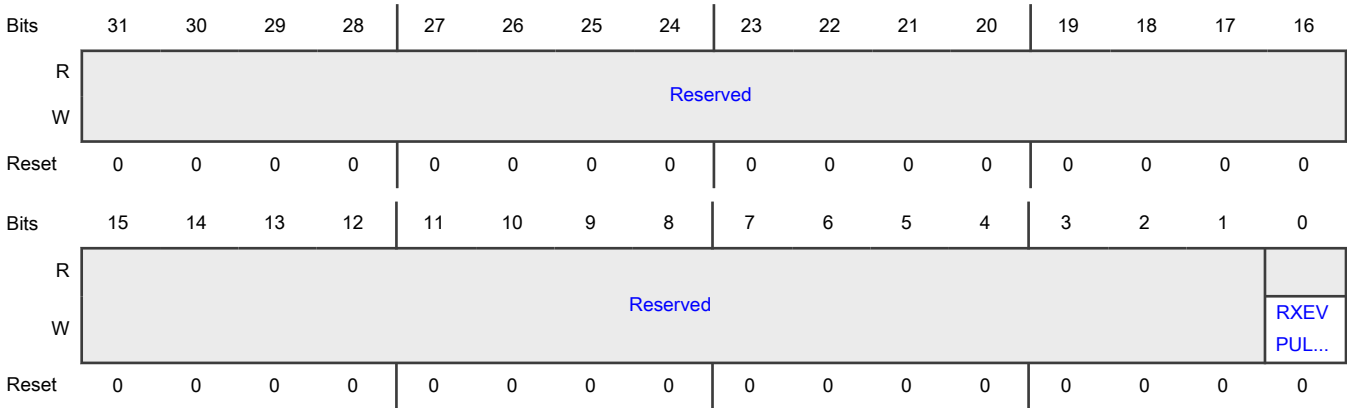
Offset

Register	Offset
RXEVPULSEGEN	200h

Function

This register allows generating a pulse on the CM33 RXEV input that will wake it up if it is stopped in a WFE instruction.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 RXEVPULSEGEN	RX Event Pulse Generator Writing a 1 to this register will create a logic 1 pulse for one APB bus clock. It is automatically cleared. 0b - No effect 1b - Pulse RXEV High for one PSCLK cycle

Chapter 5

Reset Controller (RSTCTL)

5.1 Chip-specific RSTCTL information

Table 14. Reference links to related information

Topic	Related module	Reference
Full description	RSTCTL	RSTCTL
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

This section includes chip-specific information for RSTCTL controller.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

5.1.1 Module instances

This device has two instances of the RSTCTL module, RSTCTL0 and RSTCTL1.

5.2 Overview

The RSTCTL module manages resets for the chip. It also provides registers for software to:

- see the source of the latest reset event
- reset individual peripheral modules

5.3 Functional description

The main system reset will be triggered by the following:

- Power-on reset
- External reset input
- PMC Low-voltage detect Reset

SYSRESETREQ resets most blocks, both CPUs, and other non-debug components. A timeout of the DSP Watchdog Timer will ONLY reset the Fusion CPU. It will also assert an interrupt to the M33 CPU. Nothing else in the system should be directly impacted by this WDT. On either a system reset or a DSP WDT Timeout, the Fusion DSP CPU should be held in reset until released by a software command issued by the Control Processor (M33). The DSP main clock will also be off on reset. All blocks should be reset by chip reset except where explicitly excluded. Software Reset register bits provided in the RSTCTL for most peripherals.

5.4 Signals

This table describes the module signals.

Signal	I/O	Description
RESETN	I	External Reset input. A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and the boot code to execute. Wakes up the part from deep power-down mode.
SD1_RESET_N	O	SD/MMC1 card hardware reset
SD0_RESET_N	O	SD/MMC0 card hardware reset

5.5 Memory map and register definition

This section includes the RSTCTL module memory map and detailed descriptions of all registers.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

5.5.1 Reset Controller 0 register descriptions

This section includes the memory map and detailed descriptions of registers for Reset Controllers 0.

5.5.1.1 Reset Controller 0 Registers memory map

RSTCTL0 base address: 4000_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	System Reset Status Register (SYSRSTSTAT)	32	RW	0000_0001h
10h	Peripheral Reset Control Register 0 (PRSTCTL0)	32	RW	7DF5_1F0Ah
14h	Peripheral Reset Control Register 1 (PRSTCTL1)	32	RW	0101_800Ch
18h	Peripheral Reset Control Register 2 (PRSTCTL2)	32	RW	1C00_0001h
40h	Peripheral Reset Control Register 0 SET (PRSTCTL0_SET)	32	RW	See section
44h	Peripheral Reset Control Register 1 SET (PRSTCTL1_SET)	32	RW	See section
48h	Peripheral Reset Control Register 2 SET (PRSTCTL2_SET)	32	RW	See section
70h	Peripheral Reset Control Register 0 CLR (PRSTCTL0_CLR)	32	RW	See section
74h	Peripheral Reset Control Register 1 CLR (PRSTCTL1_CLR)	32	RW	See section
78h	Peripheral Reset Control Register 2 CLR (PRSTCTL2_CLR)	32	RW	See section

5.5.1.2 System Reset Status Register (SYSRSTSTAT)

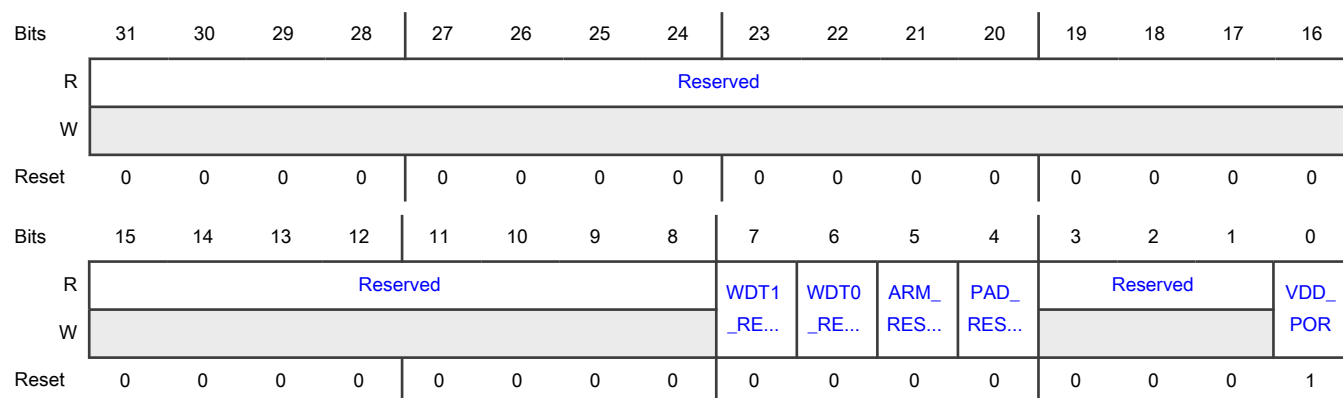
Offset

Register	Offset
SYSRSTSTAT	0h

Function

The register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external RESET pin - remains asserted after the POR signal is negated, then its bit is set to detected.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 WDT1_RESET	WatchDog Timer 1 reset was detected NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No WDT1 reset event detected 1b - WDT1 reset event detected. Write '1' to clear this bit
6 WDT0_RESET	WatchDog Timer 0 reset was detected NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No WDT0 reset event detected 1b - WDT0 reset event detected. Write '1' to clear this bit
5 ARM_RESET	ARM reset was detected Write 1 to clear this bit. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No ARM reset event is detected 1b - ARM reset was detected. Write '1' to clear this bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 PAD_RESET	RESETN pin reset was detected Write 1 to clear this bit. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No RESETN pin event is detected 1b - RESETN pin event was detected. Write '1' to clear this bit
3-1 —	Reserved
0 VDD_POR	VDD CORE Power-On Reset (POR) was detected Write 1 to clear this bit. NOTE During power on, this bit will be set to '1'. 0b - No VDD CORE POR event is detected 1b - VDD CORE POR event was detected

5.5.1.3 Peripheral Reset Control Register 0 (PRSTCTL0)

Offset

Register	Offset
PRSTCTL0	10h

Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL0 registers be accomplished by using the related PRSTCTL0_SET and PRSTCTL0_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed	SMAR	MIPI_ DS...	MIPI_ DS...	DISPL AY...	GPU	Reserv ed	SCT	USBH S_S...	USBH S_H...	USBH S_D...	USBH S_P...	Reserv ed	FLEXS PI1	Reserv ed	FLEXS PI...
W		TDMA														
Reset	0	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserv ed	Reserved		RNG	PUF	HASH CRY...	CASP ER	POWE RQU...	Reserved				Reserv ed	AXI_S WI...	Reserv ed	DSP	Reserv ed
W																	
Reset	0	0	0	1	1	1	1	1	0	0	0	0	1	0	1	0	

Fields

Field	Function
31 —	Reserved
30 SMARTDMA	SMARTDMA Event/Algorithm handler reset control 0b - Clear Reset 1b - Set Reset
29 MIPI_DSI_PHY	MIPI DSI PHY reset control NOTE This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations. 0b - Clear Reset 1b - Set Reset
28 MIPI_DSI_CON TROLLER	MIPI Digital serial Interface controller reset control NOTE This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations. 0b - Clear Reset 1b - Set Reset
27 DISPLAY_CON TROLLER	LCDIF Display Controller reset control 0b - Clear Reset 1b - Set Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
26 GPU	GPU reset control 0b - Clear Reset 1b - Set Reset
25 —	Reserved
24 SCT	SCTimer reset control 0b - Clear Reset 1b - Set Reset
23 USBHS_SRAM	USB RAM reset control 0b - Clear Reset 1b - Set Reset
22 USBHS_HOST	USB HOST reset control 0b - Clear Reset 1b - Set Reset
21 USBHS_DEVICE	USB HS Device reset control 0b - Clear Reset 1b - Set Reset
20 USBHS_PHY	USB PHY reset control 0b - Clear Reset 1b - Set Reset
19 —	Reserved
18 FLEXSPI1	FLEXSPI1 reset control 0b - Clear Reset 1b - Set Reset
17 —	Reserved
16 FLEXSPI0_OTFAD	FLEXSPI0 and OTFAD reset control 0b - Clear Reset 1b - Set Reset
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14-13 —	Reserved
12 RNG	RNG reset control 0b - Clear Reset 1b - Set Reset
11 PUF	PUF reset control 0b - Clear Reset 1b - Set Reset
10 HASHCRYPT	Hash-Crypt reset control 0b - Clear Reset 1b - Set Reset
9 CASPER	CASPER reset control 0b - Clear Reset 1b - Set Reset
8 POWERQUAD	POWERQUAD reset control 0b - Clear Reset 1b - Set Reset
7-5 —	Reserved
4 —	Reserved
3 AXI_SWITCH	AXI Switch reset control 0b - Clear Reset 1b - Set Reset
2 —	Reserved
1 DSP	Fusion F1 DSP reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations. 0b - Clear Reset 1b - Set Reset
0	Reserved
—	

5.5.1.4 Peripheral Reset Control Register 1 (PRSTCTL1)

Offset

Register	Offset
PRSTCTL1	14h

Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL1 register be accomplished by using the related PRSTCTL1_SET and PRSTCTL1_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
R	Reserved								SHSG PIO0	Reserved								ADC0
W																		
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1		
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R	ACMP	Reserved											SDIO1	SDIO0	Reserved			
W	0																	
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0		

Fields

Field	Function
31-25 —	Reserved
24 SHSGPIO0	Secure GPIO 0 reset control 0b - Clear Reset 1b - Set Reset
23-17 —	Reserved
16 ADC0	Analog-to-Digital converter reset control 0b - Clear Reset 1b - Set Reset
15 ACMP0	Analog comparator reset control 0b - Clear Reset 1b - Set Reset
14-4 —	Reserved
3 SDIO1	SDIO1 reset control 0b - Clear Reset 1b - Set Reset
2 SDIO0	SDIO0 reset control 0b - Clear Reset 1b - Set Reset
1-0 —	Reserved

5.5.1.5 Peripheral Reset Control Register 2 (PRSTCTL2)**Offset**

Register	Offset
PRSTCTL2	18h

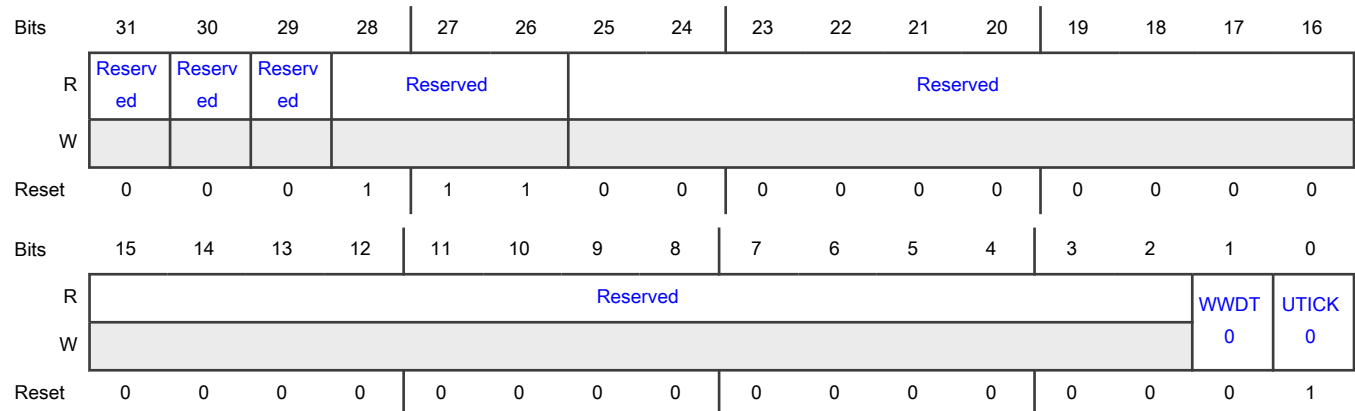
Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL2

registers be accomplished by using the related PRSTCTL2_SET and PRSTCTL2_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram**Fields**

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28-26 —	Reserved
25-2 —	Reserved
1 WWDT0	Watchdog timer reset control WDT reset 0b - Clear Reset 1b - Set Reset
0 UTICK0	Micro-tick timer reset control 0b - Clear Reset 1b - Set Reset

5.5.1.6 Peripheral Reset Control Register 0 SET (PRSTCTL0_SET)

Offset

Register	Offset
PRSTCTL0_SET	40h

Function

Writing a 1 to a bit position in PRSTCTL0_SET sets the corresponding position in PRSTCTL0. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed						Reserv ed						Reserv ed		Reserv ed	
W		SMAR TDMA	MIPI_ DS...	MIPI_ DS...	DISPL AY...	GPU		SCT	USBH S_S...	USBH S_H...	USBH S_D...	USBH S_P...		FLEXS PI1		FLEXS PI...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	Reserved							Reserved				Reserv ed		Reserv ed	Reserv ed
W				RNG	PUF	HASH CRY...	CASP ER	POWE RQU...					AXI_S WI...		DSP	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 —	Reserved
30 SMARTDMA	SMARTDMA Event/Algorithm handler reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
29 MIPI_DSI_PHY	MIPI DSI PHY reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
28 MIPI_DSI_CON TROLLER	MIPI DSI controller reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
27	LCDIF DISPLAY CONTROLLER reset set

Table continues on the next page...

Table continued from the previous page...

Field	Function
DISPLAY_CON TROLLER	0b - No Effect 1b - Sets the PRSTCTL0 Bit
26 GPU	GPU reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
25 —	Reserved
24 SCT	SCTimer reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
23 USBHS_SRAM	USBHS SRAM reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
22 USBHS_HOST	USB HOST reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
21 USBHS_DEVIC E	USB Device reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
20 USBHS_PHY	USB PHY reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
19 —	Reserved
18 FLEXSPI1	FLEXSPI1 reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
17 —	Reserved
16	FLEXSPI0 and OTFAD reset set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXSPI0_OTF AD	0b - No Effect 1b - Sets the PRSTCTL0 Bit
15 —	Reserved
14-13 —	Reserved
12 RNG	RNG reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
11 PUF	PUF reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
10 HASHCRYPT	HASHCRYPT reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
9 CASPER	CASPER reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
8 POWERQUAD	POWERQUAD reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
7-5 —	Reserved
4 —	Reserved
3 AXI_SWITCH	AXI_SWITCH reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 DSP	Fusion_ DSP reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
0 —	Reserved

5.5.1.7 Peripheral Reset Control Register 1 SET (PRSTCTL1_SET)

Offset

Register	Offset
PRSTCTL1_SET	44h

Function

Writing a 1 to a bit position in PRSTCTL1_SET sets the corresponding position in PRSTCTL1. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved																
W									SHSG PIO0	Reserved							ADC0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R		Reserved													Reserved		
W	ACMP 0												SDIO1	SDIO0			
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	

Fields

Field	Function
31-25 —	Reserved
24 SHSGPIO0	SHSGPIO0 reset set 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the PRSTCTL1 Bit
23-17 —	Reserved
16 ADC0	ADC0 reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
15 ACMP0	ACMP0 reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
14-4 —	Reserved
3 SDIO1	SDIO1 reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
2 SDIO0	SDIO0 reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
1-0 —	Reserved

5.5.1.8 Peripheral Reset Control Register 2 SET (PRSTCTL2_SET)

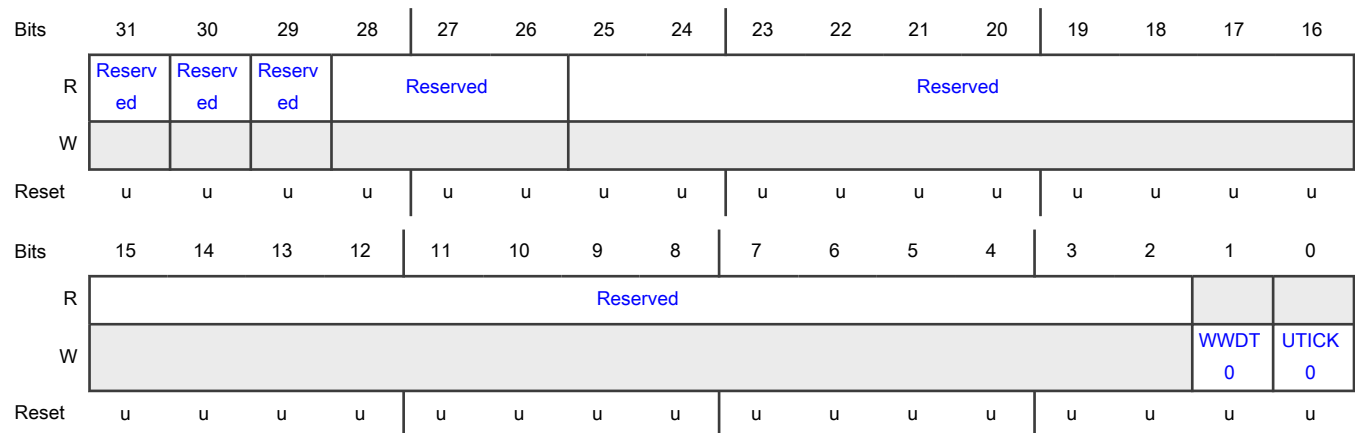
Offset

Register	Offset
PRSTCTL2_SET	48h

Function

Writing a 1 to a bit position in PRSTCTL2_SET sets the corresponding position in PRSTCTL2. This is a write-only register.

Diagram



Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28-26 —	Reserved
25-2 —	Reserved
1 WWDT0	WWDT0 reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
0 UTICK0	Micro-tick timer 0 reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit

5.5.1.9 Peripheral Reset Control Register 0 CLR (PRSTCTL0_CLR)

Offset

Register	Offset
PRSTCTL0_CLR	70h

Function

Writing a 1 to a bit position in PRSTCTL0_CLR clears the corresponding position in PRSTCTL0. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed						Reserv ed						Reserv ed		Reserv ed	
W		SMAR TDMA	MIPI_ DS...	MIPI_ DS...	DISPL AY...	GPU		SCT	USBH S_S...	USBH S_H...	USBH S_D...	USBH S_P...		FLEXS PI1		FLEXS PI...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	Reserved							Reserved				Reserv ed		Reserv ed	Reserv ed
W				RNG	PUF	HASH CRY...	CASP ER	POWE RQU...					AXI_S WI...		DSP	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 —	Reserved
30 SMARTDMA	SMARTDMA Event/Algorithm handler reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
29 MIPI_DSI_PHY	MIPI DSI PHY reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
28 MIPI_DSI_CON TROLLER	MIPI DSI controller reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
27	LCDIF DISPLAY CONTROLLER reset clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
DISPLAY_CONTROLLER	0b - No effect 1b - Clears the PRSTCTL0 Bit
26 GPU	GPU reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
25 —	Reserved
24 SCT	SCT reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
23 USBHS_SRAM	USBHS SRAM reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
22 USBHS_HOST	USB HOST reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
21 USBHS_DEVICE	USB DEVICE reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
20 USBHS_PHY	USB PHY reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
19 —	Reserved
18 FLEXSPI1	FLEXSPI1 reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
17 —	Reserved
16	FLEXSPI0 and OTFAD reset clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXSPI0_OTF AD	0b - No effect 1b - Clears the PRSTCTL0 Bit
15 —	Reserved
14-13 —	Reserved
12 RNG	RNG reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
11 PUF	PUF reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
10 HASHCRYPT	HASHCRYPT reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
9 CASPER	CASPER reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
8 POWERQUAD	POWERQUAD reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
7-5 —	Reserved
4 —	Reserved
3 AXI_SWITCH	AXI_SWITCH reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 DSP	Fusion_ F1 DSP reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
0 —	Reserved

5.5.1.10 Peripheral Reset Control Register 1 CLR (PRSTCTL1_CLR)

Offset

Register	Offset
PRSTCTL1_CLR	74h

Function

Writing a 1 to a bit position in PRSTCTL1_CLR clears the corresponding position in PRSTCTL1. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved																
W									SHSG PIO0	Reserved							ADC0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R		Reserved													Reserved		
W	ACMP 0												SDIO1	SDIO0			
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	

Fields

Field	Function
31-25 —	Reserved
24 SHSGPIO0	Secure HSGPIO0 reset clear 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Clears the PRSTCTL1 Bit
23-17 —	Reserved
16 ADC0	ADC0 reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
15 ACMP0	ACMP0 reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
14-4 —	Reserved
3 SDIO1	SDIO1 reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
2 SDIO0	SDIO0 reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
1-0 —	Reserved

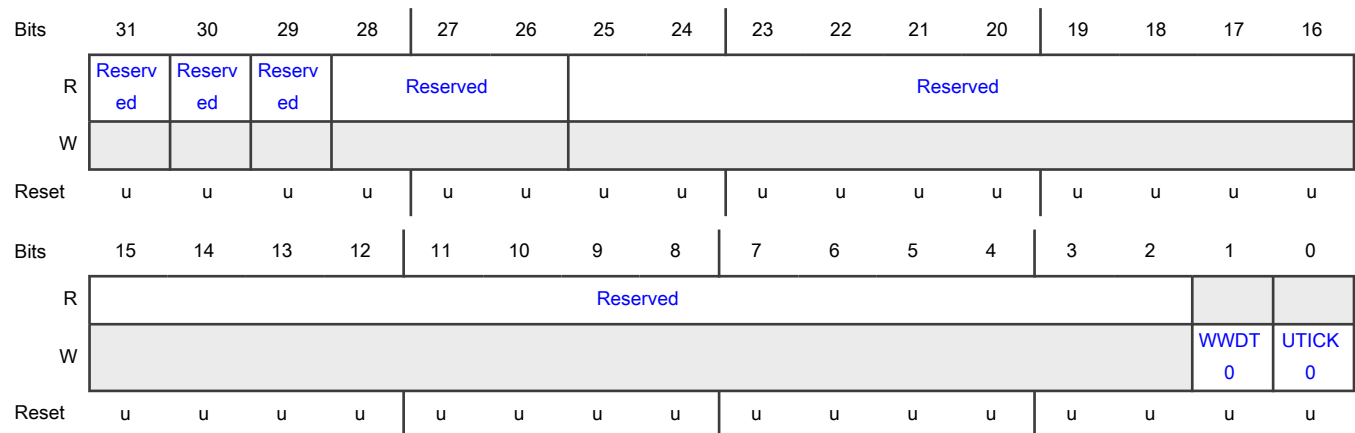
5.5.1.11 Peripheral Reset Control Register 2 CLR (PRSTCTL2_CLR)

Offset

Register	Offset
PRSTCTL2_CLR	78h

Function

Writing a 1 to a bit position in PRSTCTL2_CLR clears the corresponding position in PRSTCTL2. This is a write-only register.

Diagram**Fields**

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28-26 —	Reserved
25-2 —	Reserved
1 WWDT0	WWDT0 reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit
0 UTICK0	Micro-tick timer 0 reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit

5.5.2 Reset Controller 1 register descriptions

This section includes the memory map and detailed descriptions of registers for Reset Controllers 1.

5.5.2.1 Reset Controller 1 Registers memory map

RSTCTL1 base address: 4002_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	System Reset Status Register (SYSRSTSTAT)	32	R	0000_0001h
10h	Peripheral Reset Control Register 0 (PRSTCTL0)	32	RW	01C0_FF00h
14h	Peripheral Reset Control Register 1 (PRSTCTL1)	32	RW	B181_00FFh
18h	Peripheral Reset Control Register 2 (PRSTCTL2)	32	RW	C001_011Fh
40h	Peripheral Reset Control Register 0 SET (PRSTCTL0_SET)	32	RW	See section
44h	Peripheral Reset Control Register 1 SET (PRSTCTL1_SET)	32	RW	See section
48h	Peripheral Reset Control Register 2 SET (PRSTCTL2_SET)	32	RW	See section
70h	Peripheral Reset Control Register 0 CLR (PRSTCTL0_CLR)	32	W	See section
74h	Peripheral Reset Control Register 1 CLR (PRSTCTL1_CLR)	32	W	See section
78h	Peripheral Reset Control Register 2 CLR (PRSTCTL2_CLR)	32	RW	See section

5.5.2.2 System Reset Status Register (SYSRSTSTAT)

Offset

Register	Offset
SYSRSTSTAT	0h

Function

This register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external RESET pin - remains asserted after the POR signal is negated, then its bit is set to detected.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								WDT1 _RE...	WDT0 _RE...	ARM_ RES...	PAD_ RES...	Reserved			VDD_ POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Function
31-8 —	Reserved
7 WDT1_RESET	WDT1 reset was detected Write 1 to clear this bit. WDT1 is Watchdog Timer 1. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No WDT1 reset event is detected 1b - WDT1 reset was detected. Write '1' to clear this bit
6 WDT0_RESET	WDT0 reset was detected Write 1 to clear this bit. WDT0 is Watchdog Timer 0. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No WDT0 reset event is detected 1b - WDT0 reset was detected. Write '1' to clear this bit
5 ARM_RESET	ARM reset was detected Write 1 to clear this bit. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No ARM reset event is detected 1b - ARM reset was detected. Write '1' to clear this bit
4 PAD_RESET	RESETN pin reset was detected Write 1 to clear this bit. NOTE During power on, this bit is cleared by Power-On Reset (POR). 0b - No RESETN pin event is detected 1b - RESETN pin reset event was detected. Write '1' to clear this bit
3-1 —	Reserved
0 VDD_POR	VDD Power-On Reset (POR) was detected Write 1 to clear this bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE During power on, this bit will be set to '1'. 0b - No VDD POR event is detected 1b - VDD POR event was detected

5.5.2.3 Peripheral Reset Control Register 0 (PRSTCTL0)

Offset

Register	Offset
PRSTCTL0	10h

Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL0 registers be accomplished by using the related PRSTCTL0_SET and PRSTCTL0_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		FLEXI	Reserv	OSEV	Reserv	FLEX	DMICO	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX
W			O	ed	ENT...	ed	COM...		COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...
Reset	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEXC	FLEXC	FLEXC	FLEXC	FLEXC	FLEXC	FLEXC	FLEXC	Reserved							
W	OM...	OM...	OM...	OM...	OM...	OM...	OM...	OM...								
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 FLEXIO	FLEXIO reset control 0b - Clear Reset 1b - Set Reset
28 —	Reserved
27 OSEVENT_TIMER	OSEVENT Timer reset control 0b - Clear Reset 1b - Set Reset
26 —	Reserved
25 FLEXCOMM16	Flexcomm SPI reset control 0b - Clear Reset 1b - Set Reset
24 DMIC0	DMIC0 reset control 0b - Clear Reset 1b - Set Reset
23 FLEXCOMM15_I2C	Flexcomm15 I2C reset control 0b - Clear Reset 1b - Set Reset
22 FLEXCOMM14	Flexcomm14 SPI0 reset control 0b - Clear Reset 1b - Set Reset
21-8 FLEXCOMMn	Flexcommn reset control 0b - Clear Reset 1b - Set Reset
7-0 —	Reserved

5.5.2.4 Peripheral Reset Control Register 1 (PRSTCTL1)

Offset

Register	Offset
PRSTCTL1	14h

Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL1 register be accomplished by using the related PRSTCTL1_SET and PRSTCTL1_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FREQ	Reserv	SEMA	MU	Reserved				DMAC	DMAC	Reserved				CRC	
W	ME	ed							1	0						
Reset	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI
W									O7	O6	O5	O4	O3	O2	O1	O0
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fields

Field	Function
31 FREQME	FREQME reset control 0b - Clear Reset 1b - Set Reset
30 —	Reserved
29 SEMA	SEMA reset control 0b - Clear Reset 1b - Set Reset
28	MU reset control

Table continues on the next page...

Table continued from the previous page...

Field	Function
MU	0b - Clear Reset 1b - Set Reset
27-25 —	Reserved
24-23 DMACn	DMAC reset control 0b - Clear Reset 1b - Set Reset
22-17 —	Reserved
16 CRC	CRC reset control 0b - Clear Reset 1b - Set Reset
15-8 —	Reserved
7-0 HSGPIOn	HSGPIO[7:0] reset control 0b - Clear Reset 1b - Set Reset

5.5.2.5 Peripheral Reset Control Register 2 (PRSTCTL2)

Offset

Register	Offset
PRSTCTL2	18h

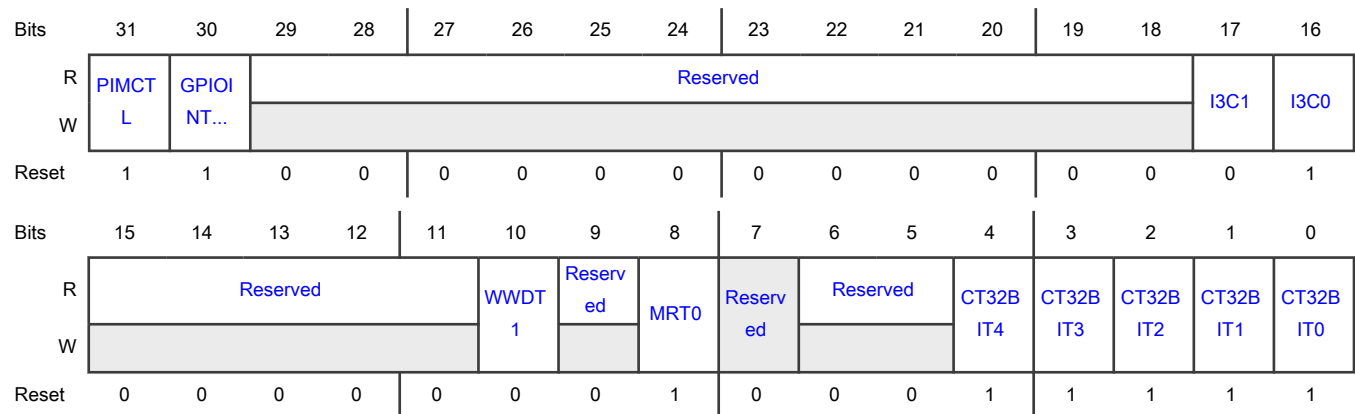
Function

This register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset. It is recommended that changes to the PRSTCTL2 registers be accomplished by using the related PRSTCTL2_SET and PRSTCTL2_CLR registers. This avoids any unintentional setting or clearing of other bits.

NOTE

The resets are synchronous so the clock to the peripheral needs to be enabled and ungated before the peripheral reset is cleared.

Diagram



Fields

Field	Function
31 PIMCTL	INPUTMUX reset control 0b - Clear Reset 1b - Set Reset
30 GPIOINTCTL	GPIOINTCTL reset control 0b - Clear Reset 1b - Set Reset
29-18 —	Reserved
17-16 I3Cn	I3Cn reset control 0b - Clear Reset 1b - Set Reset
15-11 —	Reserved
10 WWDT1	WWDT1 reset control 0b - Clear Reset 1b - Set Reset
9 —	Reserved
8 MRT0	MRT0 reset control 0b - Clear Reset 1b - Set Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 —	Reserved
6-5 —	Reserved
4-0 CT32BITn	CT32BIT[4:0] reset 0b - Clear Reset 1b - Set Reset

5.5.2.6 Peripheral Reset Control Register 0 SET (PRSTCTL0_SET)

Offset

Register	Offset
PRSTCTL0_SET	40h

Function

Writing a 1 to a bit position in PRSTCTL0_SET sets the corresponding position in PRSTCTL0. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			Reserv ed		Reserv ed										
W			FLEXI O		OSEV ENT...		FLEX COM...	DMIC0	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									Reserved							
W	FLEXC OM...	FLEXC OM...	FLEXC OM...	FLEXC OM...	FLEXC OM...	FLEXC OM...	FLEXC OM...	FLEXC OM...								
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-30 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 FLEXIO	FEXIO reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
28 —	Reserved
27 OSEVENT_TIMER	OSEVENT Timer reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
26 —	Reserved
25 FLEXCOMM16	Flexcomm16 SPI1 reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
24 DMIC0	DMIC0 reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
23 FLEXCOMM15_I2C	Flexcomm15 I2C reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
22 FLEXCOMM14	Flexcomm14 SPI0 reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
21-8 FLEXCOMMn	Flexcommn reset set 0b - No Effect 1b - Sets the PRSTCTL0 Bit
7-0 —	Reserved

5.5.2.7 Peripheral Reset Control Register 1 SET (PRSTCTL1_SET)

Offset

Register	Offset
PRSTCTL1_SET	44h

Function

Writing a 1 to a bit position in PRSTCTL1_SET sets the corresponding position in PRSTCTL1. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserv ed			Reserved					Reserved						
W	FREQ ME		SEMA	MU					DMAC 1	DMAC 0						CRC
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W									HSGPI O7	HSGPI O6	HSGPI O5	HSGPI O4	HSGPI O3	HSGPI O2	HSGPI O1	HSGPI O0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 FREQME	FREQME reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
30 —	Reserved
29 SEMA	SEMA reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
28 MU	MU reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
27-25 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
24-23 DMACn	DMACn reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
22-17 —	Reserved
16 CRC	CRC reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit
15-8 —	Reserved
7-0 HSGPIOn	HSGPIOn reset set 0b - No effect 1b - Sets the PRSTCTL1 Bit

5.5.2.8 Peripheral Reset Control Register 2 SET (PRSTCTL2_SET)

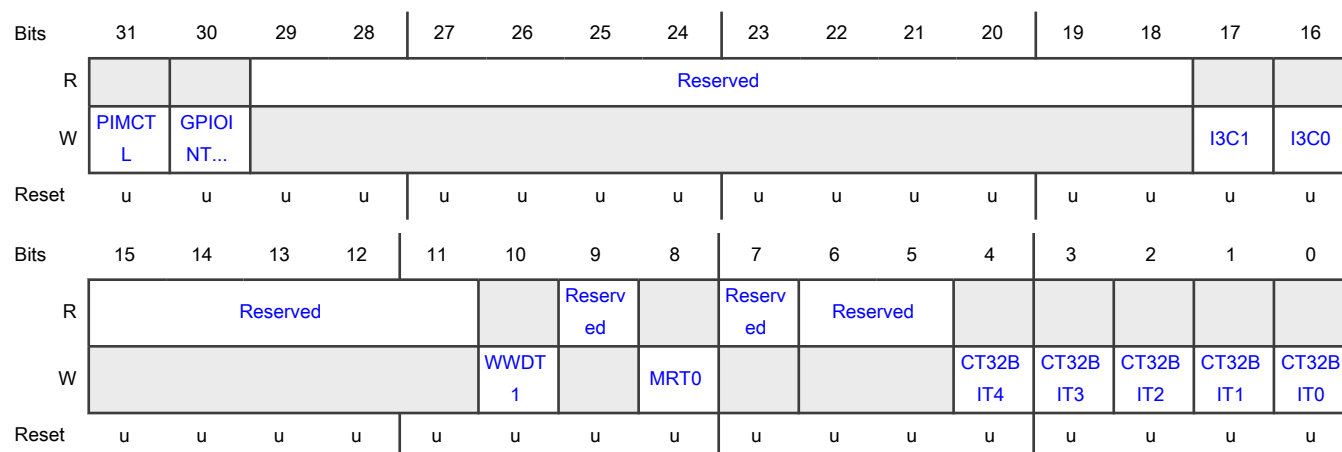
Offset

Register	Offset
PRSTCTL2_SET	48h

Function

Writing a 1 to a bit position in PRSTCTL2_SET sets the corresponding position in PRSTCTL2. This is a write-only register.

Diagram



Fields

Field	Function
31 PIMCTL	PIMCTL reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
30 GPIOINTCTL	GPIOINTCTL reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
29-18 —	Reserved
17-16 I3Cn	I3Cn reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
15-11 —	Reserved
10 WWDT1	WWDT1 reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
9 —	Reserved
8 MRT0	MRT0 reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit
7 —	Reserved
6-5 —	Reserved
4-0 CT32BITn	CT32BITn reset set 0b - No effect 1b - Sets the PRSTCTL2 Bit

5.5.2.9 Peripheral Reset Control Register 0 CLR (PRSTCTL0_CLR)

Offset

Register	Offset
PRSTCTL0_CLR	70h

Function

Writing a 1 to a bit position in PRSTCTL0_CLR clears the corresponding position in PRSTCTL0. This is a write-only register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved		FLEXIO	Reserved	OSEVENT...	Reserved	FLEXCOM...	DMIC0	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...	FLEXCOM...								
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-30 —	Reserved
29 FLEXIO	FLEXIO reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
28 —	Reserved
27 OSEVENT_TIMER	OSEVENT Timer reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
26 —	Reserved
25	Flexcomm SPI1 reset clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM16	0b - No effect 1b - Clears the PRSTCTL0 Bit
24 DMIC0	DMIC0 reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
23 FLEXCOMM15_I2C	Flexcomm I2C reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
22 FLEXCOMM14	FLexcomm SPI0 reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
21-8 FLEXCOMMn	Flexcommn reset clear 0b - No effect 1b - Clears the PRSTCTL0 Bit
7-0 —	Reserved

5.5.2.10 Peripheral Reset Control Register 1 CLR (PRSTCTL1_CLR)

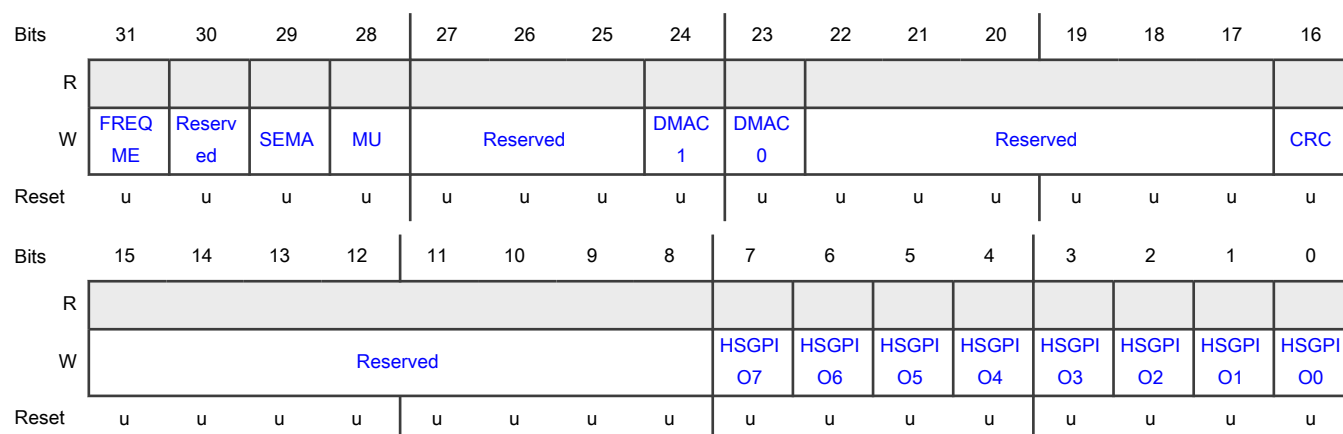
Offset

Register	Offset
PRSTCTL1_CLR	74h

Function

Writing a 1 to a bit position in PRSTCTL1_CLR clears the corresponding position in PRSTCTL1. This is a write-only register.

Diagram



Fields

Field	Function
31 FREQME	FREQME reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
30 —	Reserved
29 SEMA	SMEA reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
28 MU	MU reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
27-25 —	Reserved
24-23 DMACn	DMACn reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit
22-17 —	Reserved
16 CRC	CRC reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 —	Reserved
7-0 HSGPION	HSGPION reset clear 0b - No effect 1b - Clears the PRSTCTL1 Bit

5.5.2.11 Peripheral Reset Control Register 2 CLR (PRSTCTL2_CLR)

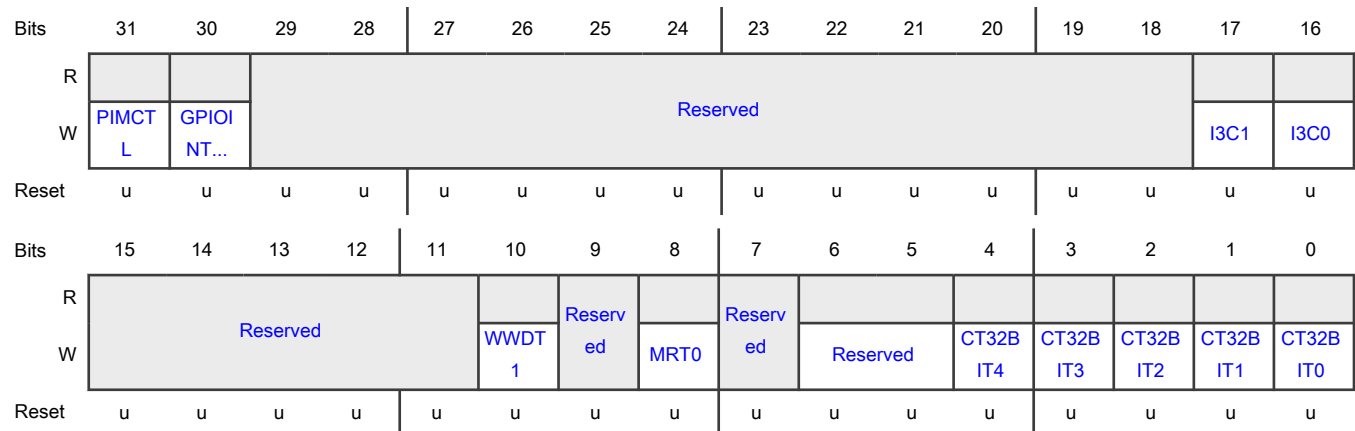
Offset

Register	Offset
PRSTCTL2_CLR	78h

Function

Writing a 1 to a bit position in PRSTCTL2_CLR clears the corresponding position in PRSTCTL2. This is a write-only register.

Diagram



Fields

Field	Function
31 PIMCTL	PIMCTL reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit
30	GPIointCTL reset clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
GPIINTCTL	0b - No effect 1b - Clears the PRSTCTL2 Bit
29-18 —	Reserved
17-16 I3Cn	I3C[1:0] reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit
15-11 —	Reserved
10 WWDT1	WWDT1 reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit
9 —	Reserved
8 MRT0	MRT0 reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit
7 —	Reserved
6-5 —	Reserved
4-0 CT32BITn	CT32BITn reset clear 0b - No effect 1b - Clears the PRSTCTL2 Bit

Chapter 6

Clock Control (CLKCTL)

6.1 Chip-specific CLKCTL information

Table 15. Reference links to related information

Topic	Related module	Reference
Full description	CLKCTL	CLKCTL
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

6.1.1 Module instances

This device has two instances of the CLKCTL module, CLKCTL0 and CLKCTL1.

6.1.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

The main clock as well as the divider values of the system clock can be controlled through the clock control registers (as shown in CLKCTL chapter).

In the Clocking diagram, MAINCLKSELA and MAINCLKSELB is used to select the main clock whereas SYSCPUAHBCLKDIV sets the divider values for the system clock.

6.1.3 Clock generation

Clocking sources included on this device are as follows:

- 192 MHz FRO ($\pm 1\%$, $\pm 0.1\%$ with tuning) -
 - Post-divider with 96, 48, 24, 12 MHz outputs.
 - Digital feedback control block for frequency tuning.
 - 192 MHz output and each of the divided outputs can be individually disabled.
- 96 MHz FRO ($\pm 1\%$, $\pm 0.1\%$ with tuning) -
 - Post-divider with 48, 24, 12, 6 MHz outputs.
 - Digital feedback control block for frequency tuning.
 - 96 MHz output and each of the divided outputs can be individually disabled.
- Main crystal oscillator (xtal-osc) with operating range of 4 MHz to 32 MHz.

- 32 kHz real-time clock (RTC) oscillator that can optionally be used as a system clock
- 1 MHz low-power oscillator (LPOSC) with $\pm 10\%$ accuracy: LPOSC serves as the watchdog oscillator and clock for the OS/Event Timer and the SysTick among others. It is also available as the system clock to both domains.
- Separate PLLs for Main Clock, DSP Clock, and Audio (Main clock and DSP clock PLLs are actually 2 PFD outputs from a single PLL module).

The following table describes signals shown in the clocking diagram.

Table 16. Clock signal descriptions

Signal name	Description
1m_lposc	Internal 1 MHz low power oscillator. Can be used as a low speed/low power system clock and/or to drive selected peripheral functions.
32k_clk	The 32 kHz output of the RTC oscillator. The 32 kHz clock must be enabled in the OSC32KHZCTL0 register (see CLKCTL). May be used for as a clock source for main_clk, DSP clock, and selected peripheral functions.
32k_wake_clk	Ultra-lower power background clock used by several peripherals. Selectable from 32k_clk or lp_32k.
audio_pll_clk	Output of the audio PLL, optionally divided by the audio PLL divider. Potentially the base clock of selected peripheral functions, including audio-specific features.
aux0_pll_clk	PFD2 output of the Main PLL. Potentially the base clock of aux0_pll_clk.
aux1_pll_clk	PFD3 output of the Main PLL. Potentially the base clock of aux1_pll_clk.
osc_clk	This is the internal clock that comes from either the CLKIN pin function or the main crystal oscillator. If used, the CLKIN pin function must be connected to a pin by selecting it in the IOPCTL block. May be used for as a clock source for the main and/or audio PLLs, main_clk, DSP clock, and selected peripheral functions.
clkln	The CLKIN pin function that can optionally be used to supply the internal clock osc_clk. If used, must be connected to a device pin by selecting it in the IOPCTL block.
CLKOUT	Many on-chip clocks can potentially be selected to be output on the CLKOUT function. This can be used for debugging purposes or to drive some external logic (see data sheet for limitations on pin output frequency. If used, the CLKOUT pin function must be connected to a pin by selecting it in the IOPCTL block.
dsp_main_clk	The clock used to derive the DSP CPU clock.
dsp_pll_clk	PFD1 output of the main PLL, optionally divided by the DSPPLLCLKDIV divider, potentially the base clock of dsp_main_clk.

Table continues on the next page...

Table 16. Clock signal descriptions (continued)

Signal name	Description
frg_clk n	The output of the Fractional Rate Generator for Flexcomm Interface n.
frgclk16	The output of the Fractional Rate Generator for Flexcomm 16, the high speed SPI interface.
frg_pll	The main_pll_clk optionally divided by a fractional rate generator. May be used to drive the function clock of any of the Flexcomm Interfaces.
FRO_DIV1	The output of 192/96 MHz fixed reference clock oscillator.
FRO_DIV2	FRO_DIV2 MHz output of FRO clock divider.
FRO_DIV4	FRO_DIV4 MHz output of FRO clock divider.
FRO_DIV8	FRO_DIV8 MHz output of FRO clock divider.
FRO_DIV16	FRO_DIV16 MHz output of FRO clock divider.
hclk	The AHB bus clock. Used by the Cortex-M33, AHB bus, APB bus, and others. Derived from main_clk.
lp_32k	1m_lposc divided by 32 to produce an ultra-low power 32 kHz clock. May be used as the source for 32k_wake_clk.
main_clk	The clock used to derive hclk (which is used by the Cortex-M33, AHB bus, APB bus, and others) and can be used as the source clock for many other peripherals functions.
main_pll_clk	PFD0 output of the Main PLL, optionally divided by the MAINPLLCLKDIV divider, potentially the base clock of main_clk, dsp_main_clk, and a number of peripheral functions.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOPCTL block. May be used as the function clock of any Flexcomm Interface, and/or to clock the DMIC peripheral.
ostimer_clk	Clock for the OS Event Timer peripheral.
rtcxin, rtcxout	Pins for the RTC 32 kHz crystal oscillator.
xtalin, xtalout	Pins for the main crystal oscillator. xtalín can be input to external square wave when oscillator is bypassed.

6.1.4 Clocking block diagram

The figure below shows the RT500 Clocking.

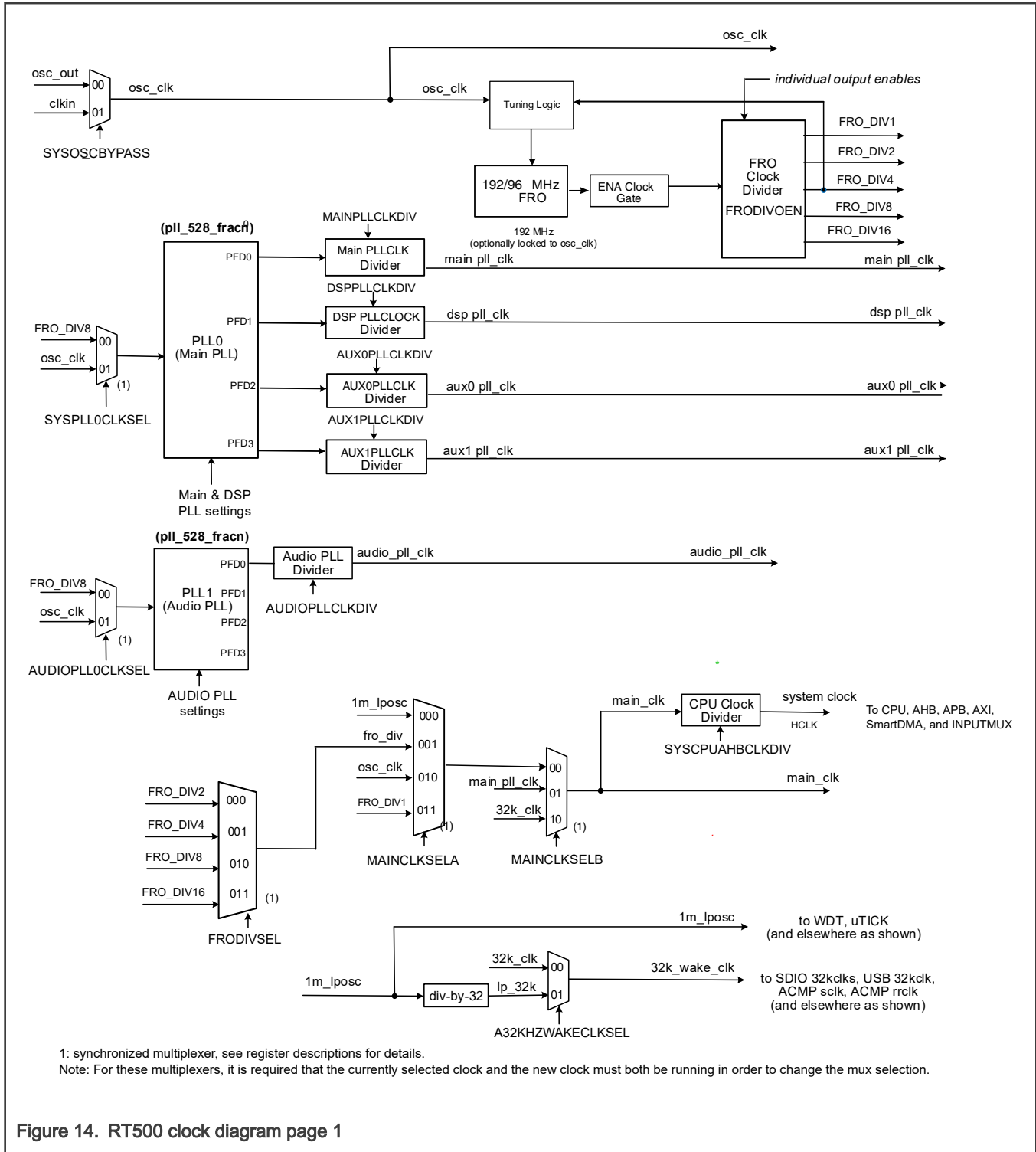
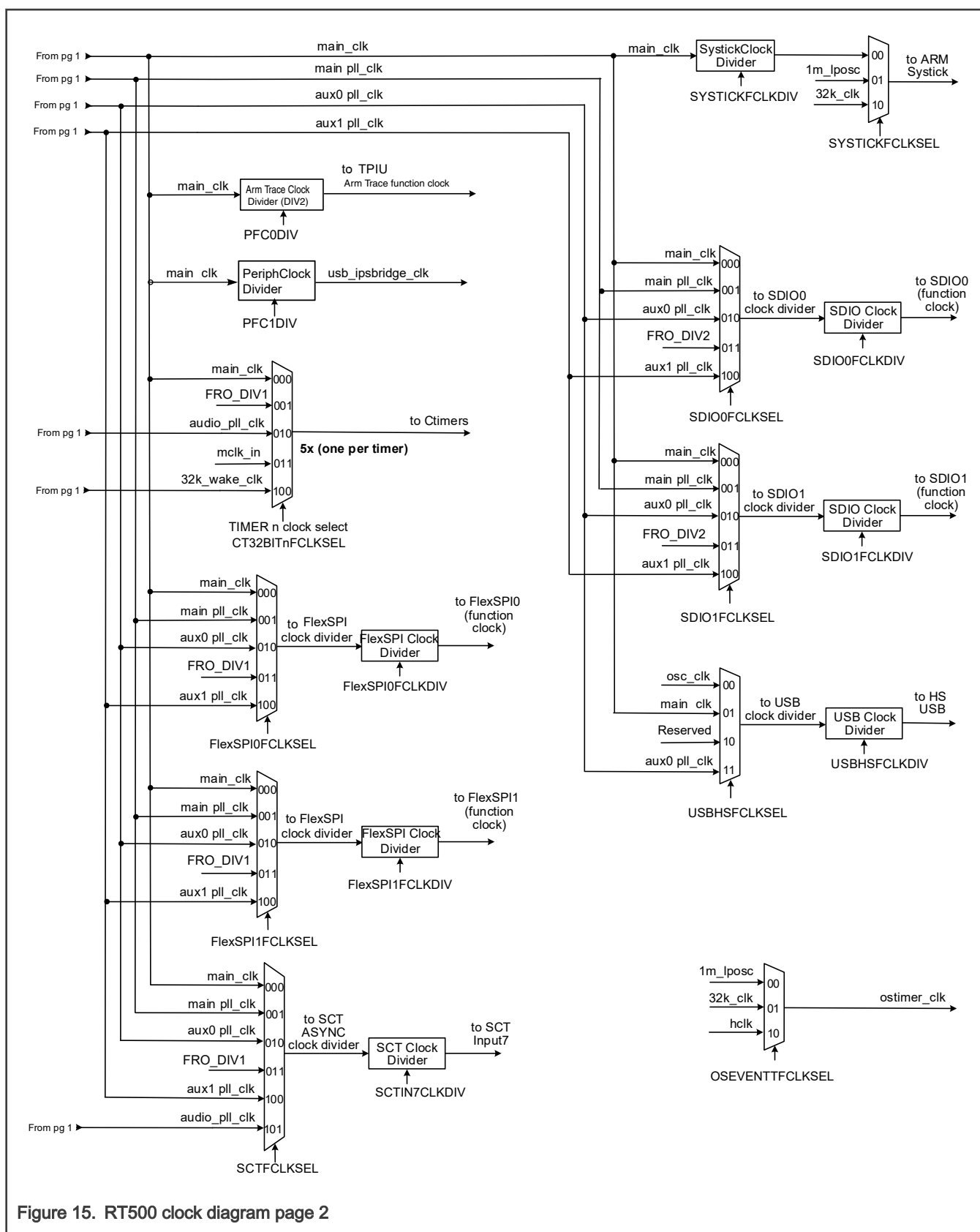
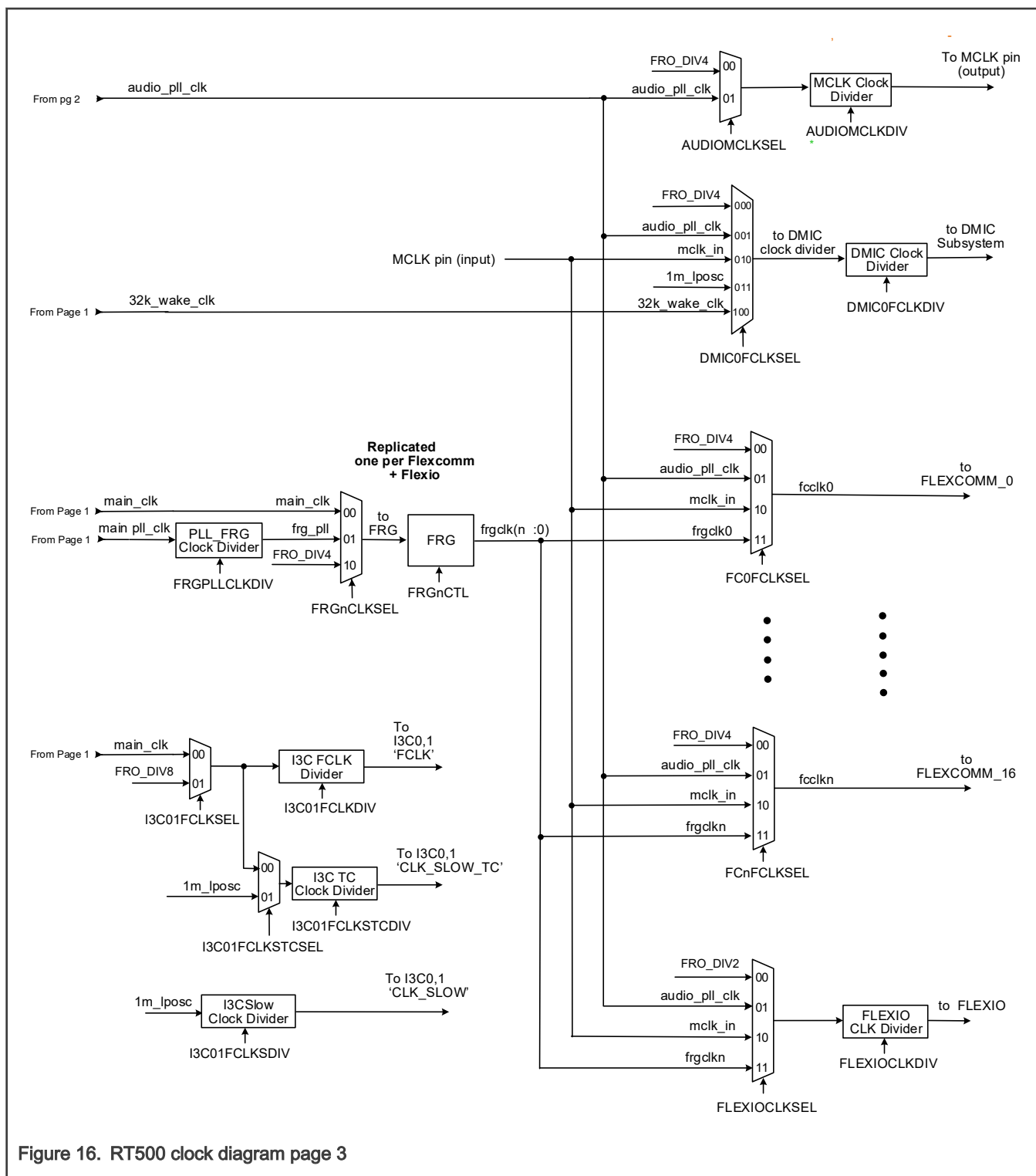


Figure 14. RT500 clock diagram page 1





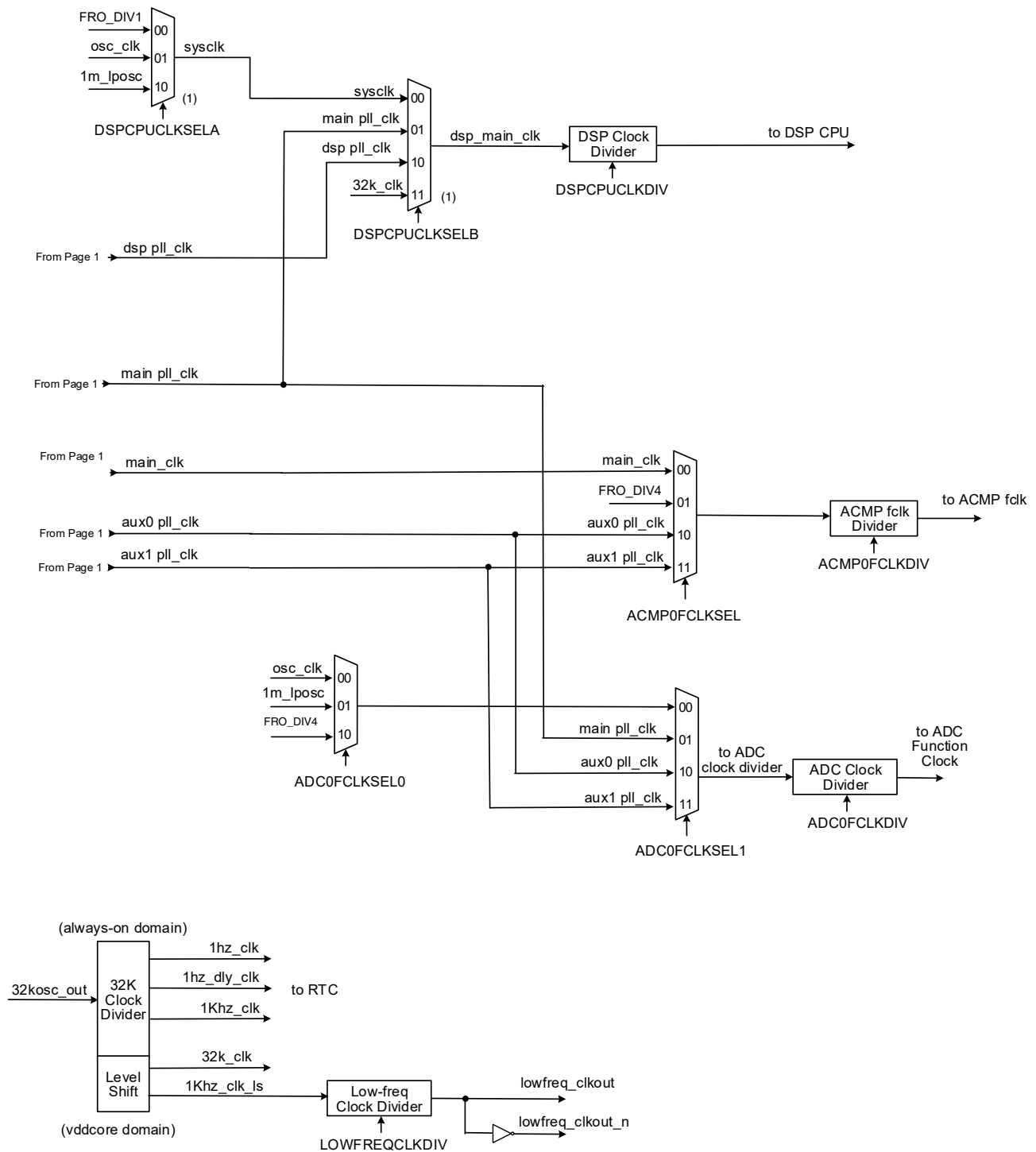


Figure 17. RT500 clock diagram page 4

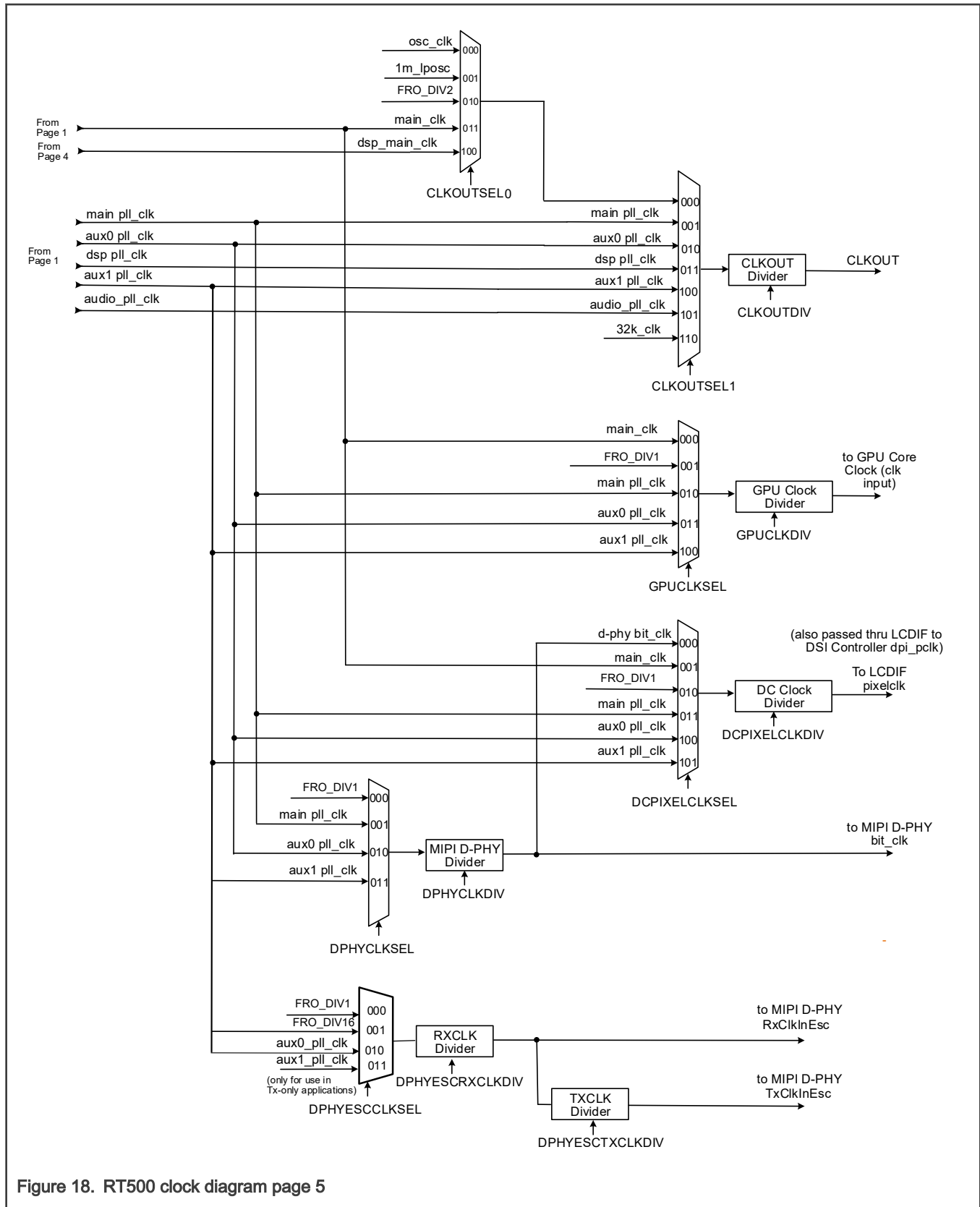


Figure 18. RT500 clock diagram page 5

6.2 Overview

Each clock controller module (CLKCTL0, CLKCTL1) takes clock sources as inputs, divides them if required, and sends the clock outputs to individual modules in the device.

The Main/System PLL can be configured to use a number of clock inputs and produce output clocks up to the maximum chip frequency, and can be used to run most on-chip functions. The outputs of the PLL can be monitored through the CLKOUT pin. The Audio PLL allows using a frequency unrelated to the Main/System PLL for audio processing.

NOTE
Main PLL is the same as the System PLL.

6.2.1 Block diagram

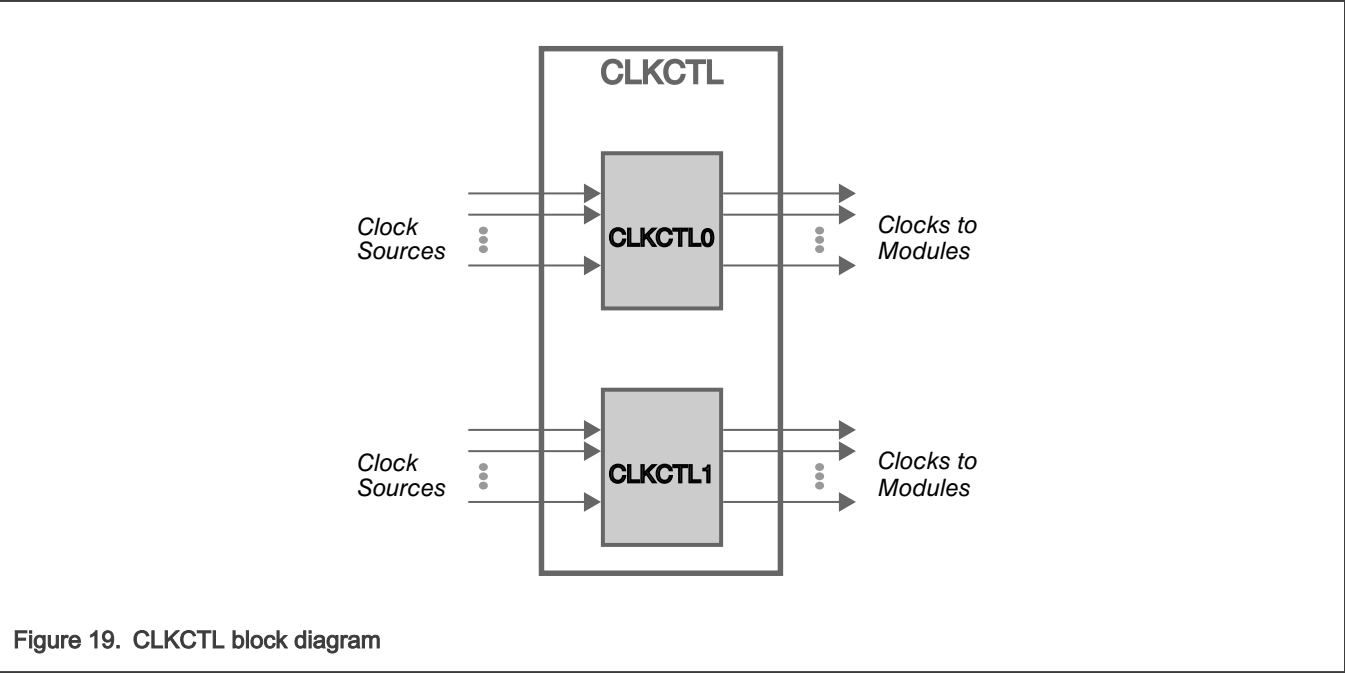


Figure 19. CLKCTL block diagram

6.2.2 Features

CLKCTL module features include:

- Can select from various clock sources:
 - 192/96 MHz Free Running Oscillator (FRO)
 - 1 MHz Low-Power Internal Oscillator
 - Crystal oscillator
 - 32 KHz crystal oscillator

6.3 Signals

This table describes the CLKCTL module signals.

Table 17. CLKCTL pin description

Name	I/O	Description
CLKOUT	O	Clock output

Table continues on the next page...

Table 17. CLKCTL pin description (continued)

Name	I/O	Description
CLKIN	I	External clock input

6.4 Functional description

Clock outputs can be programmed to use specific clock inputs. Clock outputs can be enabled or disabled, and clock sources can be divided to make slower clocks. Security schemes can be used for CLKCTL0 or CLKCTL1 separately, or for both CLKCTL0 and CLKCTL1.

6.4.1 FRO Tuner

This section provides an introduction to the FRO Tuner.

6.4.1.1 Features

- Allows a relatively inaccurate, on-chip, trimmable oscillator (FRO/IRC) to be tuned to within $\pm 0.1\%$ of its target frequency using a highly-accurate fixed reference clock (typically the crystal oscillator).
- Requires a software API to compute several required values required by the FRO_Tuner.
- Trim Register for the oscillator to be tuned resides inside this block.
- Tuning function can be enabled/disabled by software.
- For simplicity, tuning is accomplished in single steps of the oscillator's trim value. If desired, software can make the initial gross adjustment following the first measurement cycle to speed-up the tuning process.
- Includes a programmable Restart Counter to inject delay between successive measurements to give the oscillator time to settle. Boot code will load the timeout period for this counter based on the spec'd settling time of the oscillator being measured. It should typically not be necessary for the end-user to adjust this setting.

6.4.1.2 FRO Tuner Functional Description

The FRO Tuner uses an accurate "reference clock" of known frequency (typically a crystal oscillator) to tune a "target clock" provided by a relatively inaccurate, on-chip FRO or IRC to within $\pm 1\%$ of its ideal target frequency.

Tuning is accomplished by successively incrementing or decrementing the oscillator's Trim value until its frequency is within an acceptable range of the target frequency.

The sole purpose of the "reference clock" is to establish a measurement period of fixed, known duration. During this period a Target Counter counts the number of "target clock" edges that have occurred and compares this to a window around the ideal count value based on the relative frequencies of the reference and the target. Adjustments to the Trim value are made when the Target count falls outside this window.

NOTE

The "target clock" frequency must be greater than the reference clock frequency.

The design is based around two main counters. The first is a "Reference Counter" clocked by a known, accurate "reference clock" (typically the crystal oscillator). The purpose of this counter is simply to establish a fixed measurement duration. The second counter is a "Target Counter" clocked by the FRO that is being tuned ("target clock").

Each time a new measurement is triggered (either when the module is initially enabled or upon subsequent restarts) both counters are enabled. Counting proceeds until the Reference Counter reaches its maximum count. At that point the contents of the Target Counter are captured in the Capture Register. Both counters are then cleared and disabled until the next measurement cycle is launched.

The contents of the just-updated Capture Register are compared to a pair of values "range_top" and "range_bottom" which, together, establish a window of approximately 0.05% around the ideal/expected target count value (based on the ideal frequency of the FRO and the actual, known reference clock frequency).

The “Range_Top” and “Range_Bottom” values are computed from several fields programmed into the Control Register (“Expected_Count”, “Threshold_Range_Upper” and “Threshold_Range_Lower”). The values to be programmed into these fields will be provided by an API routine.

If the Target Counter value captured in the Capture Register at the end of a given measurement cycle is above “Threshold_Range_Upper” the Trim Register will be decremented. If the Capture Register value is below “Threshold_Range_Lower” the Trim Register will be incremented. Otherwise the current Trim value will be retained.

The captured Target Count value is available to be read by software via the Capture_Read Register. Bit 31 of the Capture_Read Register serves as a “data_valid” flag. This bit is set to ‘1’ each time the Capture_Read value is updated. This bit will be cleared if/when software writes to manually change the Trim Value. This register is useful if using software to perform a quick, initial “gross-tune” the FRO (see the “Software Tuning” section, below).

At the same time that the Capture Register is being loaded, an 8-bit “Restart Counter” will be launched. This counter inserts a programmable delay between successive starts to permit the FRO time to settle following any updating of the Trim value. When this counter times-out a “Restart” signal will be generated which will automatically trigger a new measurement cycle.

Measurement cycles will be repeated continuously unless the tuning function is disabled by software.

6.4.1.2.1 Software API Routine

An API routine will be provided to compute the values that must be programmed into the FRO_Tuner Control Register. These values are the expected (ideal) target count and the upper and lower ranges of acceptable count values around that ideal.

The values that the API routine will provide will be a function of the reference clock frequency.

6.4.1.2.2 Software Tuning

The FRO_Tuner makes adjustments to the oscillator frequency in single-step increments or decrements of the Trim value. These adjustments will usually be on the order of less than 1/10 percent of the oscillator frequency per cycle. If the oscillator is initially far off from its target, it could take many measurement/adjustment cycles to bring it into the target range. This could take an appreciable amount of time - especially if a slow reference clock is utilized.

If desired, software can intervene to perform an initial gross adjustment leaving the FRO_Tuner to perform fine-tuning and readjustment due to drift.

Bit 31 of the Capture_Read Register serves as a “data_valid” flag. This bit is set to ‘1’ each time the Capture_Read value is updated. This bit will be cleared if/when software writes to manually change the Trim Value.

After enabling the FRO_Tuner, software can poll this bit. When it goes high, software can compare the Capture_Read value to the “Expected_Count” value specified in the Control Register, compute how far off it is and adjust the frequency accordingly by writing directly to the Trim Register.

Writing to the Trim Register will also automatically clear the “data_valid” flag in the Capture_Read Register in case software chooses to repeat this procedure.

6.4.2 PLLs

The PLLs support a range of input frequencies and a range of VCO frequencies. In addition to a typical integer frequency multiplier, the PLLs also provide for a fractional frequency multiplier. The PLL VCO frequency is given by:

$$\text{PLL output frequency} = \text{Fref} * (\text{MULT} + \text{NUM}/\text{DENOM})$$

Fref = PLL input reference frequency

MULT = 8-bit PLL multiplier (See CLKCTL0[SYSPLL0CTL0] and CLKCTL1[SYSPLL0CTL0])

NUM = 30-bit numerator for fractional multiplier (See CLKCTL0[SYSPLL0NUM] and CLKCTL1[SYSPLL0NUM])

DENOM = 30-bit denominator for fractional multiplier (See CLKCTL0[SYSPLL0DENOM] and CLKCTL1[SYSPLL0DENOM])

While a PLL is operating, NUM value can be changed by a step of 1 at a time and a delay at each change for the PLL to settle to the new setting. DENOM value cannot be changed while a PLL is running.

The PLL VCO frequency will be output on a particular Phase Fractional Divider (PFD) output of the PLL if the PFD is set to 18. This does not alter the PLL output (the value 18 causes the PFD to divide by 1).

The PFD settings can be used to alter the PLL VCO frequency before it is output from the PLL. Each PFD output may have a different setting. The PFD output frequency is given by:

PFD Output = $18/N \times FVCO$ where $N = 12$ to 35 .

PFDn	Fraction	Decimal
12	18/12	1.5
13	18/13	1.3846
14	18/14	1.2857
15	18/15	1.2
16	18/16	1.125
17	18/17	1.0588
18	18/18	1
19	18/19	0.9474
20	18/20	0.9
21	18/21	0.8571
22	18/22	0.8182
23	18/23	0.7826
24	18/24	0.75
25	18/25	0.72
26	18/26	0.6923
27	18/27	0.6667
28	18/28	0.6429
29	18/29	0.6207
30	18/30	0.6
31	18/31	0.5806
32	18/32	0.5625
33	18/33	0.5455
34	18/34	0.5294
35	18/35	0.5143

6.4.2.1 Main/System PLL

The Main PLL can be configured to use a number of clock inputs and produce output clocks up to the maximum chip frequency. It can be used to run most on-chip functions. The outputs of the PLL can be monitored through the CLKOUT pin. The Main PLL creates a stable output clock at a higher frequency than the input clock.

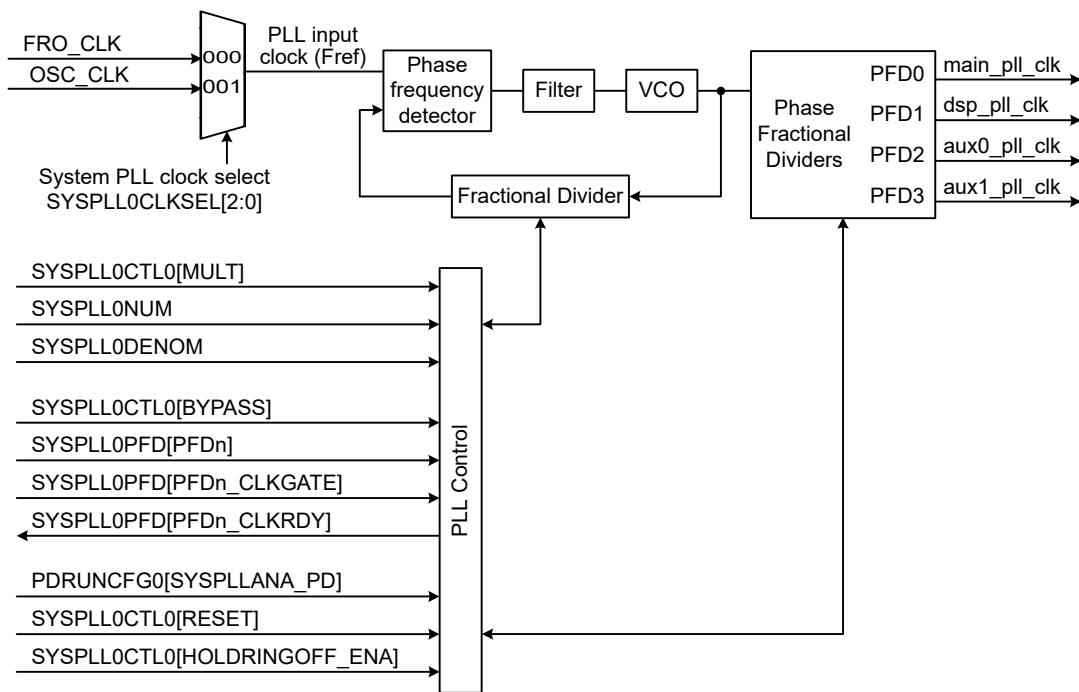


Figure 20. Main / System PLL

6.4.2.2 Audio PLL

The Audio PLL is intended to allow using an frequency unrelated to the Main PLL for audio processing. The PLLs have the ability to use a fractional multiplier that can be used to generate accurate audio frequencies.

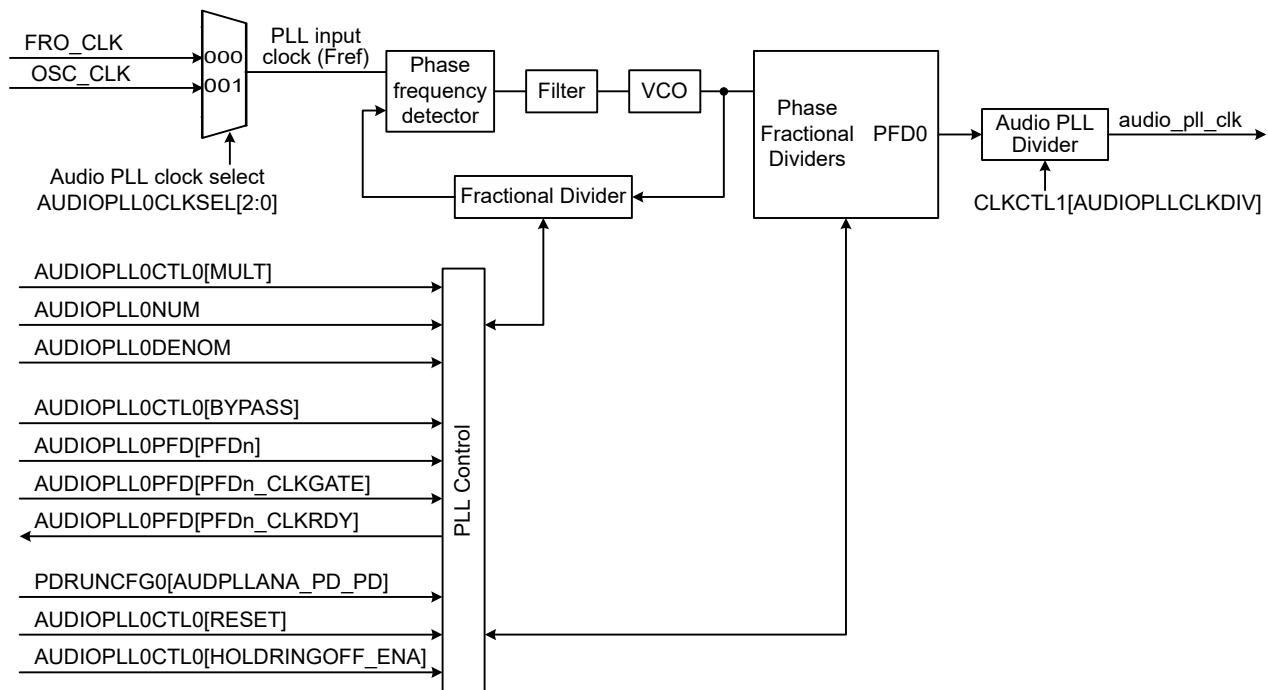


Figure 21. Audio PLL

6.4.2.2.1 Generating Audio Frequencies

PLLs have the ability to use a fractional multiplier that can be used to generate accurate audio frequencies.

Example PLL settings to generate desired audio frequency

Input frequency = 24 MHz

Desired frequency = 44.1 kHz

Over Sampled frequency = $256 \times 44.1 \text{ kHz} = 11.289 \text{ MHz}$

Multiplier = 22.578 [MULT = 22, NUM = 578, DENOM = 1000]

VCO Output = $24 \text{ MHz} \times 22.578 = 541.872 \text{ MHz}$

The PLL output can be divided by 48 to get the 11.289 MHz oversample frequency (see AUDIOPLLCLKDIV).

The above assumes that the related PFD is set to 18 so that it does not alter the PLL output. Other audio frequencies can be similarly generated by manipulating multiplier (MULT), numerator (NUM) and denominator (DENOM). Alternatively, the PFD might be used to create the desired frequency from a different VCO frequency.

6.4.2.3 PLL Limitations

- Input frequencies supported are 5 to 26 MHz.
- PLL multiplier values supported are 16 through 22.
- VCO frequency range supported is 80 MHz to 572 MHz. (This range can be calculated from the minimum input frequency times the minimum PLL multiplier and the maximum input frequency times the maximum PLL multiplier.)

6.5 Memory map and registers

This section includes the CLKCTL module memory map and detailed descriptions of all registers. The CLKCTL module is organized into 2 separate clock controllers: CLKCTL0 and CLKCTL1.

Having 2 separate clock controllers in this module allows the system to independently apply secure access control schemes to the clock controllers. A single clock controller or both clock controllers may be secured or not. See the device's system clocking information.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

6.5.1 Clock Controller 0 register descriptions

This section includes the memory map and register descriptions for Clock Controller 0.

6.5.1.1 CLKCTL0 memory map

CLKCTL0 base address: 4000_1000h

Offset	Register	Width (In bits)	Access	Reset value
10h	Clock Control 0 (PSCCTL0)	32	RW	0000_0005h
14h	Clock Control 1 (PSCCTL1)	32	RW	0000_0000h
18h	Clock Control 2 (PSCCTL2)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
40h	Clock Control 0 Set (PSCCTL0_SET)	32	RW	See section
44h	Clock Control 1 Set (PSCCTL1_SET)	32	RW	See section
48h	Clock Control 2 Set (PSCCTL2_SET)	32	RW	See section
70h	Clock Control 0 Clear (PSCCTL0_CLR)	32	RW	See section
74h	Clock Control 1 Clear (PSCCTL1_CLR)	32	RW	See section
78h	Clock Control 2 Clear (PSCCTL2_CLR)	32	RW	See section
80h	Free Running Oscillator Control (FRO_CONTROL)	32	RW	0000_0000h
84h	Free Running Oscillator Captured Value (FRO_CAPVAL)	32	R	0000_0000h
8Ch	Free Running Oscillator Trim (FRO_RDTRIM)	32	RW	0000_03BFh
90h	Free Running OscillatorSC Trim (FRO_SCTRM)	32	RW	0000_0020h
108h	FRO Clock Divider (FRODIVSEL)	32	RW	0000_0000h
10Ch	FRO Clock Status (FROCLKSTATUS)	32	R	0000_0000h
110h	FRO Enable Register (FRODIVOEN)	32	RW	0000_0000h
130h	Low Frequency Clock Divider (LOWFREQCLKDIV)	32	RW	0000_0000h
160h	System Oscillator Control 0 (SYSOSCCTL0)	32	RW	0000_0000h
168h	OSC Clock Source Select (SYSOSCBYPASS)	32	RW	0000_0000h
190h	Low Power Oscillator Control 0 (LPOSCCTL0)	32	RW	807B_C4D4h
1C0h	32 KHz Oscillator Control 0 (OSC32KHZCTL0)	32	RW	0000_0000h
200h	System PLL 0 Clock Select (SYSPLL0CLKSEL)	32	RW	0000_0007h
204h	System PLL0 Control 0 (SYSPLL0CTL0)	32	RW	0016_0002h
20Ch	System PLL0 Lock Time Div2 (SYSPLL0LOCKTIMEDIV2)	32	RW	0000_CAFeh
210h	System PLL0 Numerator (SYSPLL0NUM)	32	RW	04DD_2F15h
214h	System PLL0 Denominator (SYSPLL0DENOM)	32	RW	1FFF_FFDBh
218h	System PLL0 PFD (SYSPLL0PFD)	32	RW	8080_8080h
240h	Main PLL Clock Divider (MAINPLLCLKDIV)	32	RW	0000_0000h
244h	DSP PLL Clock Divider (DSPPLLCLKDIV)	32	RW	0000_0000h
248h	AUX0 PLL Clock Divider (AUX0PLLCLKDIV)	32	RW	0000_0000h
24Ch	AUX1 PLL Clock Divider (AUX1PLLCLKDIV)	32	RW	0000_0000h
400h	System CPU AHB Clock Divider (SYSCPUAHBCLKDIV)	32	RW	0000_0000h
430h	Main Clock Select A (MAINCLKSELA)	32	RW	0000_0000h
434h	Main Clock Select B (MAINCLKSELB)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
500h	PFC divider 0 (trace clock) (PFC0DIV)	32	RW	4000_0000h
504h	PFC divider 1 (USB HS PHY bus clock) (PFC1DIV)	32	RW	4000_0000h
620h	FlexSPI0 Functional Clock Select (FLEXSPI0CLKSEL)	32	RW	0000_0007h
624h	FlexSPI0 Functional Clock Divider (FLEXSPI0FCLKDIV)	32	RW	4000_0000h
630h	FlexSPI1 Functional Clock Select (FLEXSPI1CLKSEL)	32	RW	0000_0007h
634h	FlexSPI1 Functional Clock Divider (FLEXSPI1FCLKDIV)	32	RW	4000_0000h
640h	SCT Functional Clock Select (SCTFCLKSEL)	32	RW	0000_0007h
644h	SCT Functional Clock Divider (SCTIN7CLKDIV)	32	RW	4000_0000h
660h	High Speed USB Functional Clock Select (USBHSFCLKSEL)	32	RW	0000_0007h
664h	High Speed USB Functional Clock Divider (USBHSFCLKDIV)	32	RW	4000_0000h
680h	SDIO0 Functional Clock Select (SDIO0FCLKSEL)	32	RW	0000_0007h
684h	SDIO0 Functional Clock Divider (SDIO0FCLKDIV)	32	RW	4000_0000h
690h	SDIO1 Functional Clock Select (SDIO1FCLKSEL)	32	RW	0000_0007h
694h	SDIO1 Functional Clock Divider (SDIO1FCLKDIV)	32	RW	4000_0000h
6D0h	ADC0 Functional Clock Select 0 (ADC0FCLKSEL0)	32	RW	0000_0007h
6D4h	ADC0 Functional Clock Select 1 (ADC0FCLKSEL1)	32	RW	0000_0007h
6D8h	ADC0 Functional Clock Divider (ADC0FCLKDIV)	32	RW	4000_0000h
700h	UTICK Functional Clock Select (UTICKFCLKSEL)	32	RW	0000_0007h
720h	WDT0 Functional Clock Select (WDT0FCLKSEL)	32	RW	0000_0000h
730h	32 KHz Wake Clock Source Select (A32KHZWAKECLKSEL)	32	RW	0000_0001h
734h	32 KHz Wake Clock Divider (A32KHZWAKECLKDIV)	32	RW	0000_001Fh
760h	SYSTICK Functional Clock Select (SYSTICKFCLKSEL)	32	RW	0000_0007h
764h	SYSTICK Functional Clock Divider (SYSTICKFCLKDIV)	32	RW	4000_0000h
770h	MIPI-DSI PHY Clock Select (DPHYCLKSEL)	32	RW	0000_0007h
774h	MIPI-DSI PHY Clock Divider (DPHYCLKDIV)	32	RW	4000_0000h
778h	MIPI-DSI DPHY Escape Mode Clock Select (DPHYESCCLKSEL)	32	RW	0000_0007h
77Ch	MIPI-DSI DPHY Escape Mode Receive Clock Divider (DPHYESCRXCLKDIV)	32	RW	4000_0010h
780h	MIPI-DSI DPHY Escape Mode Transmit Clock Divider (DPHYESCTXCLKDIV)	32	RW	4000_0011h
790h	GPU Clock Select (GPUCLKSEL)	32	RW	0000_0007h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
794h	GPU Clock Divider (GPUCLKDIV)	32	RW	4000_0000h
7A0h	LCDIF Pixel Clock Select (DCPIXELCLKSEL)	32	RW	0000_0007h
7A4h	LCDIF Pixel Clock Divider (DCPIXELCLKDIV)	32	RW	4000_0000h

6.5.1.2 Clock Control 0 (PSCCTL0)

Offset

Register	Offset
PSCCTL0	10h

Function

This register enables the clocks to individual system and peripheral blocks. It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTLn_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	SMAR	Reserv	MIPI_	DISPL	GPU_	Reserv	SCT_	USBH	USBH	USBH	USBH	Reserv	FLEXS	OTP_	FLEXS
W	ed	TDM...	ed	DS...	AY...	CLK	ed	CLK	S_S...	S_H...	S_D...	S_P...	ed	PI...	CTL...	PI...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			RNG_	PUF_	HASH	CASP	POWE	Reserved			AXI_C	AXI_S	ROM_	DSP_	Reserv
W	Reserved			CLK	CLK	CRY...	ER...	RQU...	Reserved			TL...	WI...	CTR...	CLK	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Function
31	Reserved
—	
30	Smart DMA clock control 0b - Disable clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
SMARTDMA_C LK	1b - Enable clock
29 —	Reserved
28 MIPI_DSI_CTL R_CLK	MIPI-DSI Controller clock control 0b - Disable clock 1b - Enable clock
27 DISPLAY_CTL R_CLK	Display Controller clock control 0b - Disable clock 1b - Enable clock
26 GPU_CLK	GPU clock control 0b - Disable clock 1b - Enable clock
25 —	Reserved
24 SCT_CLK	SCT clock control 0b - Disable clock 1b - Enable clock
23 USBHS_SRAM _CLK	USB HS SRAM clock control 0b - Disable clock 1b - Enable clock
22 USBHS_HOST_ CLK	USB HS Host clock control 0b - Disable clock 1b - Enable clock
21 USBHS_DEVIC E_CLK	USB HS Device clock control 0b - Disable clock 1b - Enable clock
20 USBHS_PHY_C LK	USB HS PHY clock control 0b - Disable clock 1b - Enable clock
19	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
18 FLEXSPI1_CLK	FLEXSPI1 clock control 0b - Disable clock 1b - Enable clock
17 OTP_CTLR_CLK	OTP Controller clock control 0b - Disable clock 1b - Enable clock
16 FLEXSPI0_OTFAD_CLK	FLEXSPI0 / OTFAD clock control 0b - Disable clock 1b - Enable clock
15-13 —	Reserved
12 RNG_CLK	Random Number Generator (RNG) clock control 0b - Disable clock 1b - Enable clock
11 PUF_CLK	PUF clock control 0b - Disable clock 1b - Enable clock
10 HASHCRYPT_CLK	HASHCRYPT clock control 0b - Disable clock 1b - Enable clock
9 CASPER_CLK	CASPER clock control 0b - Disable clock 1b - Enable clock
8 POWERQUAD_CLK	POWERQUAD clock control 0b - Disable clock 1b - Enable clock
7-5 —	Reserved
4 AXI_CTLR_CLK	AXI Controller clock control 0b - Disable clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable clock
3 AXI_SWITCH_CLK	AXI Switch clock control 0b - Disable clock 1b - Enable clock
2 ROM_CTRLR_CLK	128KB ROM Controller clock control 0b - Disable clock 1b - Enable clock
1 DSP_CLK	DSP clock control 0b - Disable clock 1b - Enable clock
0 —	Reserved

6.5.1.3 Clock Control 1 (PSCCTL1)

Offset

Register	Offset
PSCCTL1	14h

Function

This register enables the clocks to individual system and peripheral blocks. It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserved				Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ACMP				Reserved								SDIO1		SDIO0	
W	0_C...												_C...		_C...	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29-25 —	Reserved
24 SHSGPIO0_CLK	SHSGPIO0 clock control 0b - Disable clock 1b - Enable clock
23-17 —	Reserved
16 ADC0_CLK	ADC0 clock control 0b - Disable clock 1b - Enable clock
15 ACMP0_CLK	ACMP0 clock control 0b - Disable clock 1b - Enable clock
14-4 —	Reserved
3 SDIO1_CLK	SDIO1 clock control 0b - Disable clock 1b - Enable clock
2 SDIO0_CLK	SDIO0 clock control 0b - Disable clock 1b - Enable clock
1-0 —	Reserved

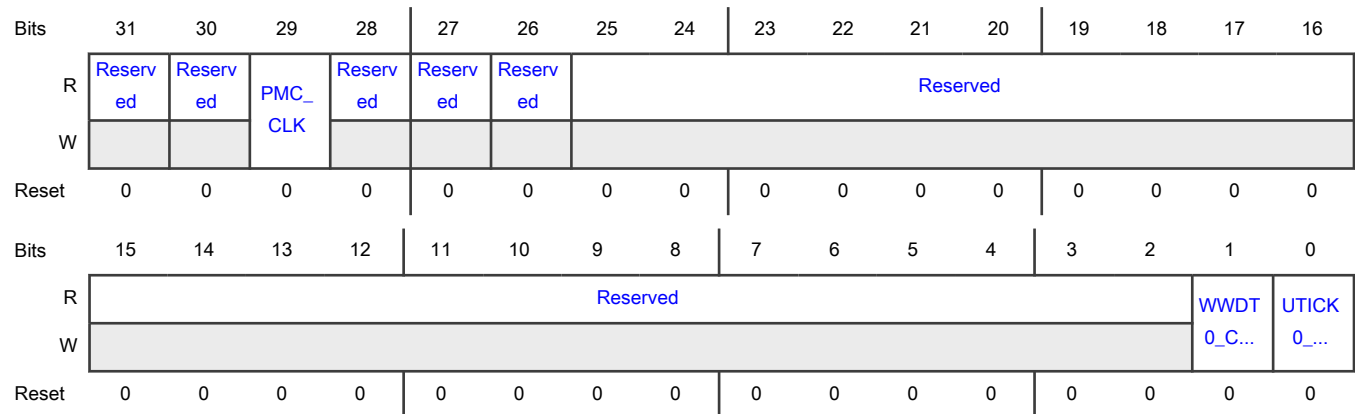
6.5.1.4 Clock Control 2 (PSCCTL2)**Offset**

Register	Offset
PSCCTL2	18h

Function

This register enables the clocks to individual system and peripheral blocks. It is recommended that changes to the PSCCTLn registers be accomplished by using the related PSCCTLn_SET and PSCCTL0n_CLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Diagram



Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 PMC_CLK	Power Management Controller clock control 0b - Disable clock 1b - Enable clock
28 —	Reserved
27 —	Reserved
26 —	Reserved
25-2 —	Reserved
1 WWDT0_CLK	Watchdog Timer 0 clock control 0b - Disable clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable clock
0 UTICK0_CLK	Micro-Tick Timer 0 clock control 0b - Disable clock 1b - Enable clock

6.5.1.5 Clock Control 0 Set (PSCCTL0_SET)

Offset

Register	Offset
PSCCTL0_SET	40h

Function

Writing a 1 to a bit position in sets the corresponding position in PSCCTL0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed		Reserv ed				Reserv ed						Reserv ed			
W		SMAR TDM...		MIPI_ DS...	DISPL AY...	GPU_ CLK		SCT_ CLK	USBH S_S...	USBH S_H...	USBH S_D...	USBH S_P...		FLEXS PI...	OTP_ CTL...	FLEXS PI...
Reset	0	u	0	u	u	u	0	u	u	u	u	u	0	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserved							Reserv ed
W				RNG_ CLK	PUF_ CLK	HASH CRY...	CASP ER...	POWE RQU...					AXI_C TL...	AXI_S WI...	ROM_ CTR...	DSP_ CLK
Reset	0	0	0	u	u	u	u	u	0	0	0	u	u	u	u	0

Fields

Field	Function
31 —	Reserved
30 SMARTDMA_C LK	Smart DMA clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
29 —	Reserved
28 MIPI_DSI_CTL R_CLK	MIPI-DSI Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
27 DISPLAY_CTL R_CLK	Display Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
26 GPU_CLK	GPU clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
25 —	Reserved
24 SCT_CLK	SCT clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
23 USBHS_SRAM _CLK	USB HS SRAM clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
22 USBHS_HOST_ CLK	USB HS Host clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
21 USBHS_DEVIC E_CLK	USB HS Device clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
20 USBHS_PHY_C LK	USB HS PHY clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
19 —	Reserved
18	FLEXSPI1 clock set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXSPI1_CLK	0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
17 OTP_CTLR_CLK	OTP Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
16 FLEXSPI0_OTFAD_CLK	FLEXSPI0 / OTFAD clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
15-13 —	Reserved
12 RNG_CLK	Random Number Generator (RNG) clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
11 PUF_CLK	PUF clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
10 HASHCRYPT_CLK	HASHCRYPT clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
9 CASPER_CLK	CASPER clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
8 POWERQUAD_CLK	POWERQUAD clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
7-5 —	Reserved
4 AXI_CTLR_CLK	AXI Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
3	AXI Switch clock set

Table continues on the next page...

Table continued from the previous page...

Field	Function
AXI_SWITCH_CLK	0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
2 ROM_CTRLR_CLK	128KB ROM Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
1 DSP_CLK	DSP clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL0 register
0 —	Reserved

6.5.1.6 Clock Control 1 Set (PSCCTL1_SET)

Offset

Register	Offset
PSCCTL1_SET	44h

Function

Writing a 1 to a bit position in sets the corresponding position in PSCCTL1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		Reserved						Reserved							
W							SHSG PIO...									ADC0_ CLK
Reset	0	0	0	0	0	0	0	u	0	0	0	0	0	0	0	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		Reserved													Reserved	
W	ACMP 0_C...												SDIO1 _C...	SDIO0 _C...		
Reset	u	0	0	0	0	0	0	0	0	0	0	0	u	u	0	0

Fields

Field	Function
31-30 —	Reserved
29-25 —	Reserved
24 SHSGPIO0_CLK	SHSGPIO0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL1 register
23-17 —	Reserved
16 ADC0_CLK	ADC0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL1 register
15 ACMP0_CLK	ACMP0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL1 register
14-4 —	Reserved
3 SDIO1_CLK	SDIO1 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL1 register
2 SDIO0_CLK	SDIO0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL1 register
1-0 —	Reserved

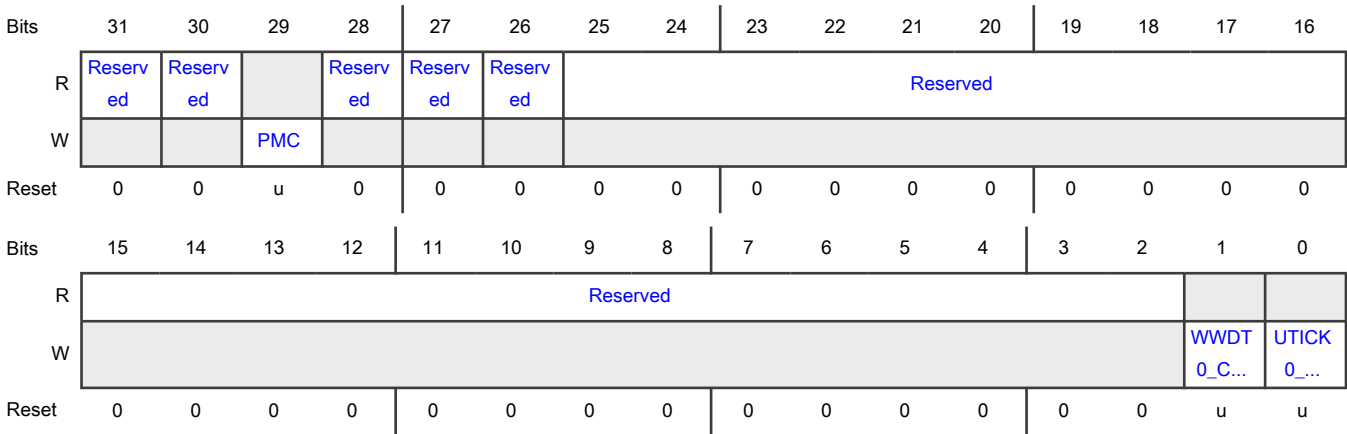
6.5.1.7 Clock Control 2 Set (PSCCTL2_SET)**Offset**

Register	Offset
PSCCTL2_SET	48h

Function

Writing a 1 to a bit position in sets the corresponding position in PSCCTL2.

Diagram



Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 PMC	Power Management Controller clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL2 register
28 —	Reserved
27 —	Reserved
26 —	Reserved
25-2 —	Reserved
1 WWDT0_CLK	Watchdog Timer 0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL2 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 UTICK0_CLK	Micro-Tick Timer 0 clock set 0b - No effect 1b - Sets the corresponding bit in PSCCTL2 register

6.5.1.8 Clock Control 0 Clear (PSCCTL0_CLR)

Offset

Register	Offset
PSCCTL0_CLR	70h

Function

Writing a 1 to a bit position in clears the corresponding position in PSCCTL0 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed		Reserv ed				Reserv ed						Reserv ed			
W		SMAR TDM...		MIPI_ DS...	DISPL AY...	GPU_ CLK		SCT_ CLK	USBH S_S...	USBH S_H...	USBH S_D...	USBH S_P...		FLEXS PI...	OTP_ CTL...	FLEXS PI...
Reset	0	u	0	u	u	u	0	u	u	u	u	u	0	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserved							Reserv ed
W				RNG_ CLK	PUF_ CLK	HASH CRY...	CASP ER...	POWE RQU...					AXI_C TL...	AXI_S WI...	ROM_ CTR...	DSP_ CLK
Reset	0	0	0	u	u	u	u	u	0	0	0	u	u	u	u	0

Fields

Field	Function
31 —	Reserved
30 SMARTDMA_C LK	Smart DMA clock set 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
28 MIPI_DSI_CTL R_CLK	MIPI-DSI Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
27 DISPLAY_CTL R_CLK	Display Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
26 GPU_CLK	GPU clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
25 —	Reserved
24 SCT_CLK	SCT clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
23 USBHS_SRAM _CLK	USB HS SRAM clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
22 USBHS_HOST_ CLK	USB HS Host clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
21 USBHS_DEVIC E_CLK	USB HS Device clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
20 USBHS_PHY_C LK	USB HS PHY clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
19 —	Reserved
18 FLEXSPI1_CLK	FLEXSPI1 clock clear 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Clears the corresponding bit in PSCCTL0 register
17 OTP_CTLR_CLK	OTP Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
16 FLEXSPI0_OTFAD_CLK	FLEXSPI0 / OTFAD clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
15-13 —	Reserved
12 RNG_CLK	RNG clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
11 PUF_CLK	PUF clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
10 HASHCRYPT_CLK	HASHCRYPT clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
9 CASPER_CLK	CASPER clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
8 POWERQUAD_CLK	POWERQUAD clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
7-5 —	Reserved
4 AXI_CTLR_CLK	AXI Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
3	AXI Switch clock clear 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
AXI_SWITCH_CLK	1b - Clears the corresponding bit in PSCCTL0 register
2 ROM_CTRLR_CLK	128KB ROM Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
1 DSP_CLK	DSP clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL0 register
0 —	Reserved

6.5.1.9 Clock Control 1 Clear (PSCCTL1_CLR)

Offset

Register	Offset
PSCCTL1_CLR	74h

Function

Writing a 1 to a bit position in clears the corresponding position in PSCCTL1 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		Reserved						Reserved							
W							SHSG PIO...								ADC0_ CLK	
Reset	0	0	0	0	0	0	0	u	0	0	0	0	0	0	0	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		Reserved													Reserved	
W	ACMP 0_C...												SDIO1 _C...	SDIO0 _C...		
Reset	u	0	0	0	0	0	0	0	0	0	0	0	u	u	0	0

Fields

Field	Function
31-30 —	Reserved
29-25 —	Reserved
24 SHSGPIO0_CLK	SHSGPIO0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL1 register
23-17 —	Reserved
16 ADC0_CLK	ADC0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL1 register
15 ACMP0_CLK	ACMP0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL1 register
14-4 —	Reserved
3 SDIO1_CLK	SDIO1 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL1 register
2 SDIO0_CLK	SDIO0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL1 register
1-0 —	Reserved

6.5.1.10 Clock Control 2 Clear (PSCCTL2_CLR)**Offset**

Register	Offset
PSCCTL2_CLR	78h

Function

Writing a 1 to a bit position in clears the corresponding position in PSCCTL2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	Reserved		Reserved	Reserved	Reserved	Reserved									
W			PMC_CLK													
Reset	0	0	u	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W													WWDT0_C...	UTICK0_...		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	u	u

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 PMC_CLK	Power Management Controller clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL2 register
28 —	Reserved
27 —	Reserved
26 —	Reserved
25-2 —	Reserved
1 WWDT0_CLK	Watchdog Timer 0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL2 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 UTICK0_CLK	Micro-Tick Timer 0 clock clear 0b - No effect 1b - Clears the corresponding bit in PSCCTL2 register

6.5.1.11 Free Running Oscillator Control (FRO_CONTROL)

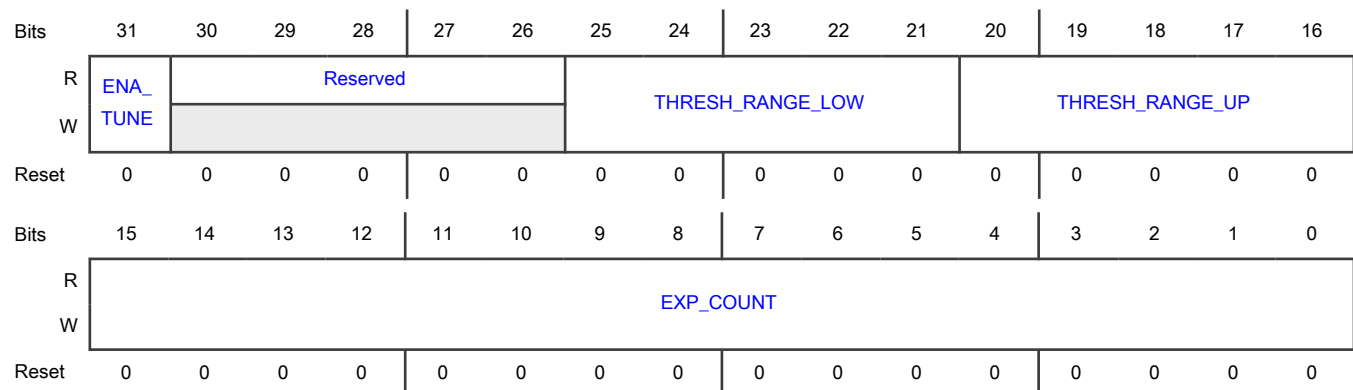
Offset

Register	Offset
FRO_CONTROL	80h

Function

This register enables tuning the FRO Tuner.

Diagram



Fields

Field	Function
31 ENA_TUNE	Enable Tuning Setting ENA_TUNE bit will start the tuning process. Tuning will continue indefinitely until ENA_TUNE bit is cleared. 0b - Stop tuning 1b - Start tuning
30-26 —	Reserved
25-21	Threshold Range Lower Limit

Table continues on the next page...

Table continued from the previous page...

Field	Function
THRESH_RANGE_LOW	The upper and lower threshold values define a range of counts above EXP_COUNT to keep the oscillator within the target range. NOTE An API routine must provide the Threshold Range Lower Limit value.
20-16 THRESH_RANGE_UP	Threshold Range Upper Limit The upper and lower threshold values define a range of counts above EXP_COUNT to keep the oscillator within the target range. NOTE An API routine must provide the Threshold Range Upper Limit value.
15-0 EXP_COUNT	Expected Count The “ideal” count value for the Target Counter. The EXP_COUNT field is a function of the ratio of the reference clock frequency and the ideal target clock frequency.

6.5.1.12 Free Running Oscillator Captured Value (FRO_CAPVAL)

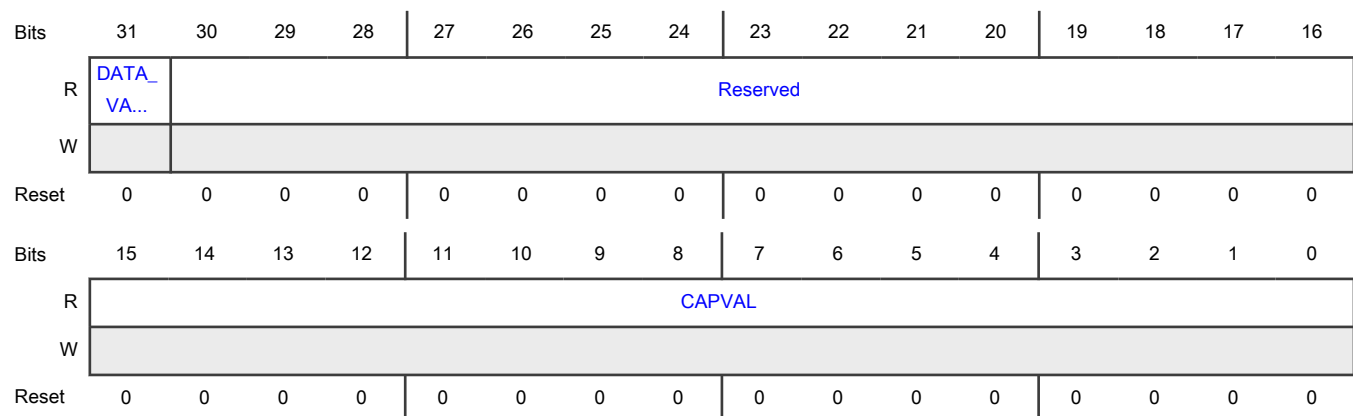
Offset

Register	Offset
FRO_CAPVAL	84h

Function

Capture register for the FRO Tuner.

Diagram



Fields

Field	Function
31 DATA_VALID	Data Valid DATA_VALID bit is set by hardware when CAPVAL is loaded at the end of each measurement cycle. DATA_VALID is cleared automatically when the Trim Register is written to. The DATA_VALID flag may be used for software gross tuning, which is likely on initial start-up, especially when using a low-frequency reference clock. 0b - CAPVAL data is not valid 1b - CAPVAL data is valid
30-16 —	Reserved
15-0 CAPVAL	Captured Value The Target Count value captured at the end of the measurement cycle

6.5.1.13 Free Running Oscillator Trim (FRO_RDTRIM)

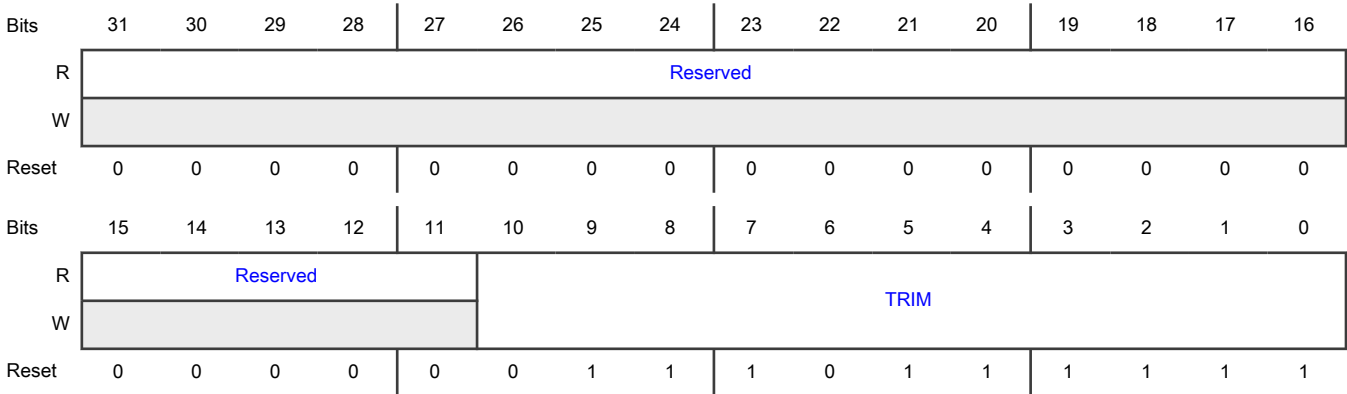
Offset

Register	Offset
FRO_RDTRIM	8Ch

Function

Contains the trim value supplied to the oscillator.

Diagram



Fields

Field	Function
31-11 —	Reserved
10-0 TRIM	It is the trim value supplied to the oscillator The value is initialized by the bootcode. During operation, the FRO_Tuner will increment or decrement the TRIM value as required, to get the oscillator into its target frequency range and keep it there. Software may also write to the TRIM field to perform “gross tuning”, which is likely on initial start-up, especially when using a low-frequency reference clock. Writing to the TRIM field also clears the DATA_VALID flag in the Captured Value Register (FRO_CAPVAL). NOTE The 0x03BF reset value shown applies only to the hardware initialization on reset. For production parts, the reset value will be modified by the bootcode before the execution of user software.

6.5.1.14 Free Running OscillatorSC Trim (FRO_SCTRIM)

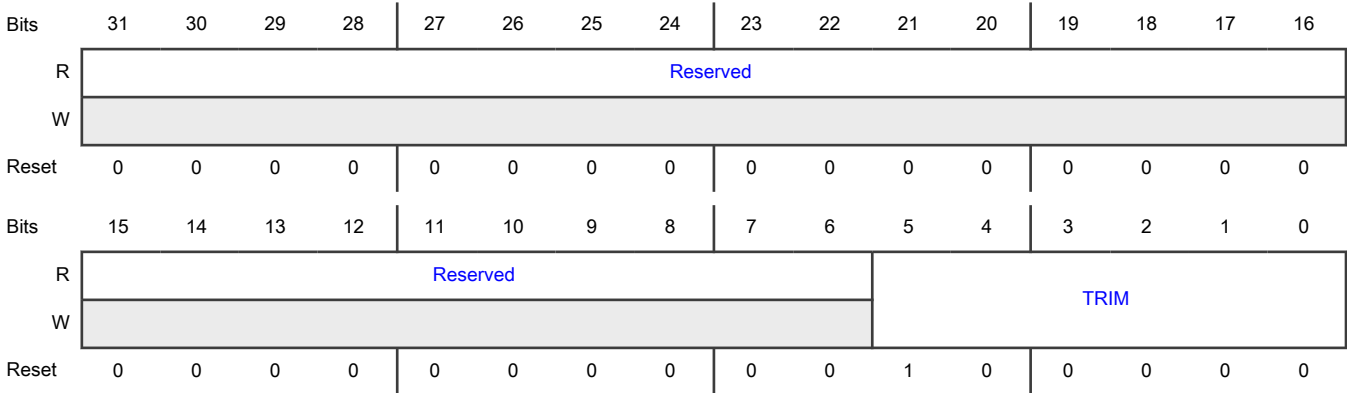
Offset

Register	Offset
FRO_SCTRIM	90h

Function

This is the sc_trim value supplied to the oscillator.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TRIM	sc_trim value for the oscillator. TRIM is the SC trim value supplied to the oscillator, and will be initialized by the bootcode. NOTE The 0x0020 reset value shown applies only to the hardware initialization on reset. The reset value will be modified by the bootcode before the execution of user software.

6.5.1.15 FRO Clock Divider (FRODIVSEL)

Offset

Register	Offset
FRODIVSEL	108h

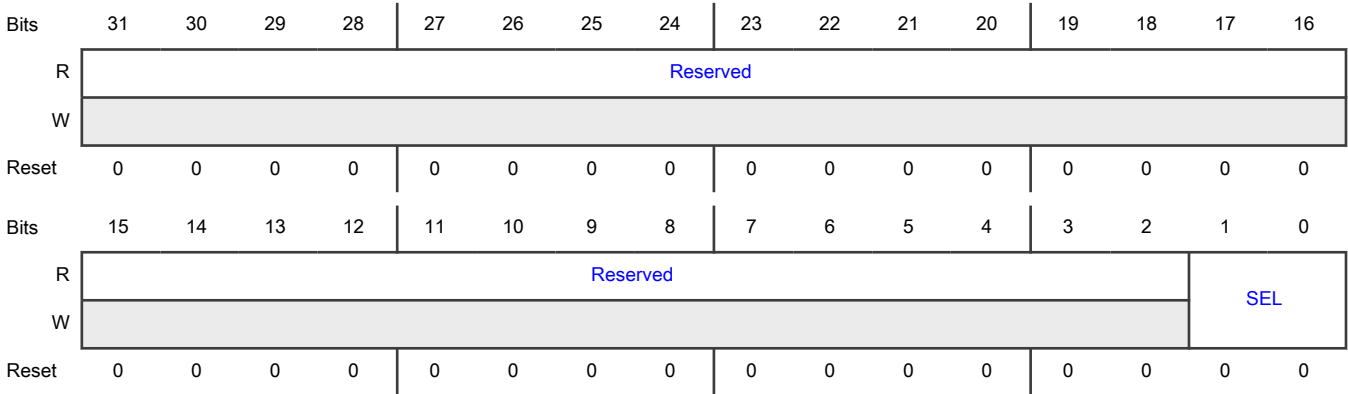
Function

Selects the divider value for the FRO clock.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change. The default clock for this mux, “FRO Clock Divided by 2”, is OFF coming out of reset. Prior to changing this mux selection it is necessary to first enable the FRO in the PDRUNCFG0 register and set both the FRO_DIV2_O_EN bit and the O_EN bit for the new clock to be selected in the FRODIVOEN register.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 SEL	Select clock Selects the FRO divided clock to be used as the main 1st Stage Clock Source. 00b - FRO_DIV2 01b - FRO_DIV4 10b - FRO_DIV8 11b - FRO_DIV16

6.5.1.16 FRO Clock Status (FROCLKSTATUS)**Offset**

Register	Offset
FROCLKSTATUS	10Ch

Function

Indicates whether the FRO clock has reached its final frequency.

Diagram

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

Fields

Field	Function
31-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 CLK_OK	FRO Clock OK Indicates when the Free-Running Oscillator (FRO) reaches 10% frequency accuracy. 0b - FRO clock has not yet reached 10% frequency accuracy 1b - FRO clock has reached 10% frequency accuracy

6.5.1.17 FRO Enable Register (FRODIVOEN)

Offset

Register	Offset
FRODIVOEN	110h

Function

The output enable bits will provide control to the 192MHz and 96MHz FRO trim output and its divided clock options.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												FRO_ DIV...	FRO_ DIV...	FRO_ DIV...	FRO_ DIV...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4 FRO_DIV16_O_ EN	FRO Divided-by-16 Clock Enable Output Enable for the FRO Divided-by-16; generates the FRO_196M and FRO_96 trim clock 0b - Disable clock 1b - Enable clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 FRO_DIV8_O_ EN	FRO Divided-by-8 Clock Enable Output Enable for the FRO Divided-by-8; generates the FRO_196M and FRO_96 trim clock 0b - Disable clock 1b - Enable clock
2 FRO_DIV4_O_ EN	FRO Divided-by-4 Clock Enable Output Enable for the FRO Divided-by-4; generates the FRO_196M and FRO_96 trim clock 0b - Disable clock 1b - Enable clock
1 FRO_DIV2_O_ EN	FRO Divided-by-2 Clock Enable Output Enable for the FRO Divided-by-2; generates the FRO_196M and FRO_96 trim clock 0b - Disable clock 1b - Enable clock
0 FRO_DIV1_O_ EN	FRO Divided-by-1 Clock Enable Output Enable for the FRO Divided-by-1; generates the FRO_196M and FRO_96 trim clock 0b - Disable clock 1b - Enable clock

6.5.1.18 Low Frequency Clock Divider (LOWFREQCLKDIV)

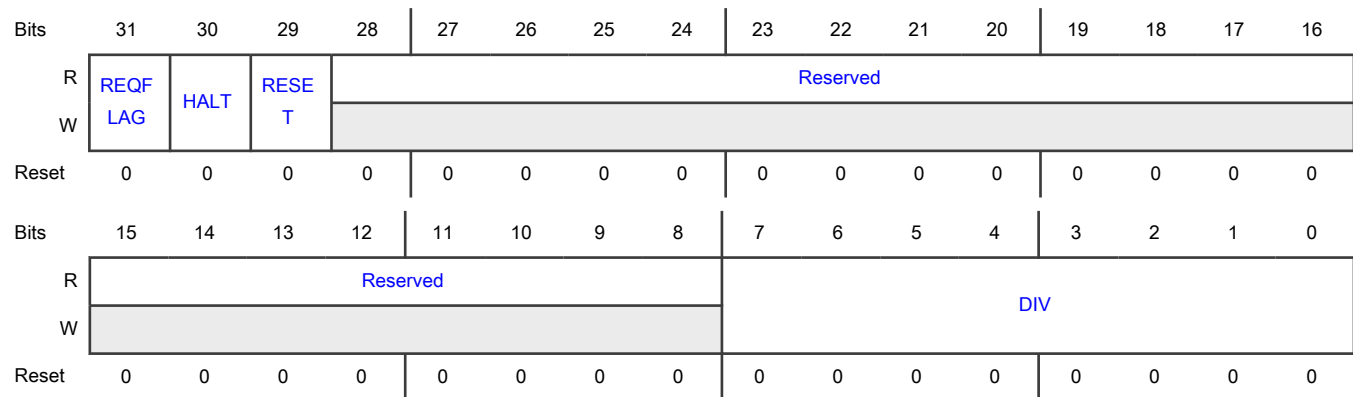
Offset

Register	Offset
LOWFREQCLKDIV	130h

Function

This register controls the Low Frequency Clock Divider.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. 0b - The Divider change has finished 1b - The Divider value has changed
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Low Frequency Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.1.19 System Oscillator Control 0 (SYSOSCCTL0)

Offset

Register	Offset
SYSOSCCTL0	160h

Function

This register controls whether the crystal oscillator is enabled or bypassed, and the oscillator mode.

One aspect of the oscillator high gain mode is that a larger voltage swing is used at the crystal pin. This gives a higher noise immunity within the oscillator and less edge to edge jitter of the internal clock. When high gain mode is not required, power used by the crystal oscillator can be reduced by using low power mode.

NOTE

High gain mode requires a 1 megohm resistor to be inserted in parallel with the external crystal. For this reason, high gain mode and low power mode cannot both be used in the same application. The board design must reflect the mode that will be used.

Diagram

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

Fields

Field	Function
31-2 —	Reserved
1 BYPASS_ENABLE	Bypass Enable Enable signal for external bypass clock. 0b - Enable Normal mode. Oscillation with crystal connected. 1b - Enable Bypass mode. In this mode a clock can be directly input into the XTALIN pin.
0 LP_ENABLE	Low Power Mode Enable 0b - Enable High Gain Mode (HP) 1b - Enable Low Power mode (LP)

6.5.1.20 OSC Clock Source Select (SYSOSCBYPASS)

Offset

Register	Offset
SYSOSCBYPASS	168h

Function

This register allows selecting either the main crystal oscillator or the CLKIN function as the source for the internal osc_clk signal.

NOTE

If SEL > 0, the oscillator is not being used and can be turned off to save power.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select SYSOSC Bypass 000b - Select OSC Clock 001b - Select Clock IN clock 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed

6.5.1.21 Low Power Oscillator Control 0 (LPOSCCTL0)

Offset

Register	Offset
LPOSCCTL0	190h

Function

This register provides the status of the 1 MHz low power oscillator (1m_lposc).

Diagram

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

Fields

Field	Function
31 CLKRDY	LPOSC Clock Ready LPOSC clock ready takes 64 microseconds before the clock is ready 0b - LPOSC clock is not ready 1b - LPOSC clock is ready
30-0 —	Reserved

6.5.1.22 32 KHz Oscillator Control 0 (OSC32KHZCTL0)

Offset

Register	Offset
OSC32KHZCTL0	1C0h

Function

This register enables the 32 kHz output of the RTC oscillator (32k_clk). This output is only available if the RTC oscillator is running. See RTC_OSC_PD bit in the RTC control register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															ENA32
W																KHZ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 ENA32KHZ	32 KHz Oscillator Enable 0b - Disable oscillator 1b - Enable oscillator

6.5.1.23 System PLL 0 Clock Select (SYSPLL0CLKSEL)**Offset**

Register	Offset
SYSPLL0CLKSEL	200h

Function

The input reference frequency can be selected using SYSPLL0CLKSEL[SEL].

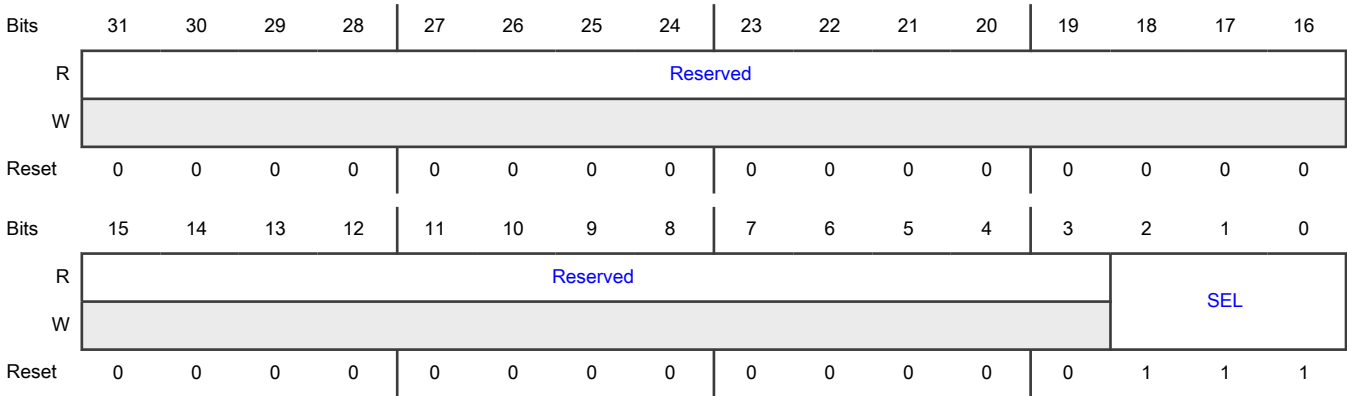
NOTE

The PLL Reference input frequency is 5 MHz to 26 MHz.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change. The default clock for this mux, "FRO Clock (Divided-by-8 selection)", is OFF coming out of reset. Prior to changing this mux selection it is necessary to first enable the FRO in the PDRUNCFG0 register and set the FRO_DIV8_O_EN bit in the FRODIVOEN register.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	System PLL0 Reference Input Clock Source 000b - FRO_DIV8 Clock 001b - OSC_CLK clock 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.1.24 System PLL0 Control 0 (SYSPLL0CTL0)

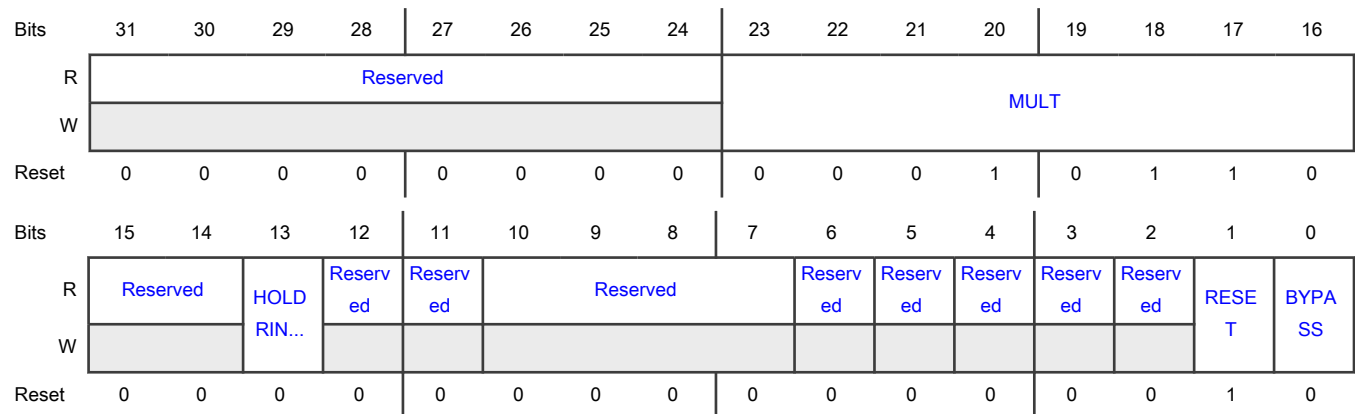
Offset

Register	Offset
SYSPLL0CTL0	204h

Function

The SYSPLL0CTL0 register has several control options for the system PLL as described below.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 MULT	Multiplication Factor Multiplication Factor for F_{SYSPLL0_OUTPUT}. The value of MULT is directly decoded. NOTE SYSPLL0CTL0[MULT] should be greater than or equal to 20 when SYSPLL0NUM[NUM] is not 0. 0001_0000b - Multiply by 16 0001_0001b - Multiply by 17 0001_0010b - Multiply by 18 0001_0011b - Multiply by 19 0001_0100b - Multiply by 20 0001_0101b - Multiply by 21 0001_0110b - Multiply by 22
15-14 —	Reserved
13 HOLDRINGOFF_F_ENA	Hold Ring Off Control Use HOLDRINGOFF_F_ENA to avoid multi-wave problems within the VCO. NOTE Assert HOLDRINGOFF_F_ENA bit in the 1st half of lock time; de-assert HOLDRINGOFF_F_ENA bit in the 2nd half of lock time. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
12 —	Reserved
11 —	Reserved
10-7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 —	Reserved
1 RESET	SYSPLL0 Reset 0b - SYSPLL0 reset is removed 1b - SYSPLL0 is placed into reset
0 BYPASS	SYSPLL0 BYPASS Mode 0b - PFD outputs are PFD-programmed clocks 1b - Bypass Mode: PFD outputs are sourced directly from the reference input clock

6.5.1.25 System PLL0 Lock Time Div2 (SYSPLL0LOCKTIMEDIV2)

Offset

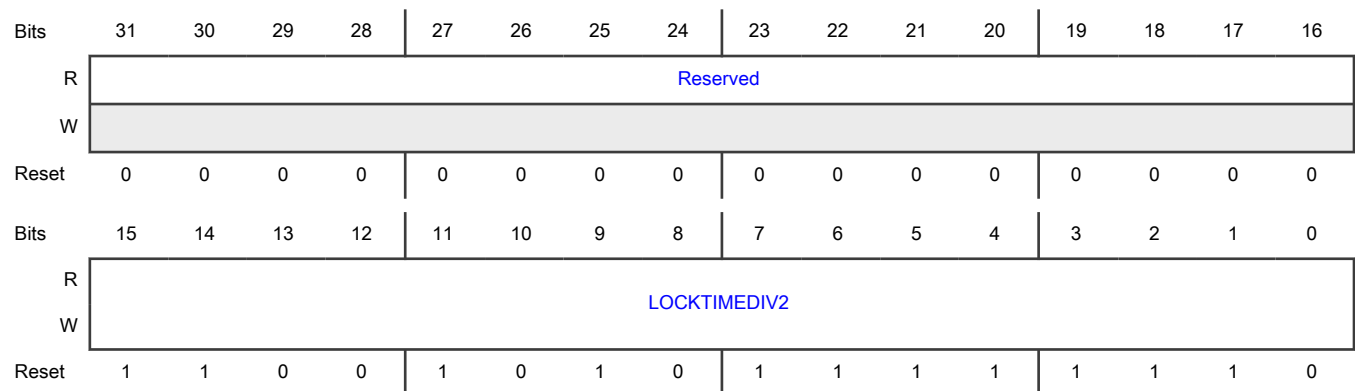
Register	Offset
SYSPLL0LOCKTIMEDIV2	20Ch

Function

On any reset, the BOOTROM will copy the SYSPLL0 worst case characterized lock time value found in the OTP Fuses into the respective fields (if the OTP valid signature is present).

To know when half of the lock time has expired and when the full lock has passed, application software needs this worst case lock time value, to control the Hold Ring Off Control bit (SYSPLL0CTL0[HOLDRINGOFF]). Any PLL lock sequence requires the HOLDRINGOFF bit:

- to be enabled ('1') during the 1st half of the lock time
- to be disabled ('0') during the 2nd half of the lock time

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 LOCKTIMEDIV2	SYSPLL0 Lock Time Divide-by-2 The programmed lock time is in uS (micro-seconds) and is programmed as 1/2 the actual lock time value.

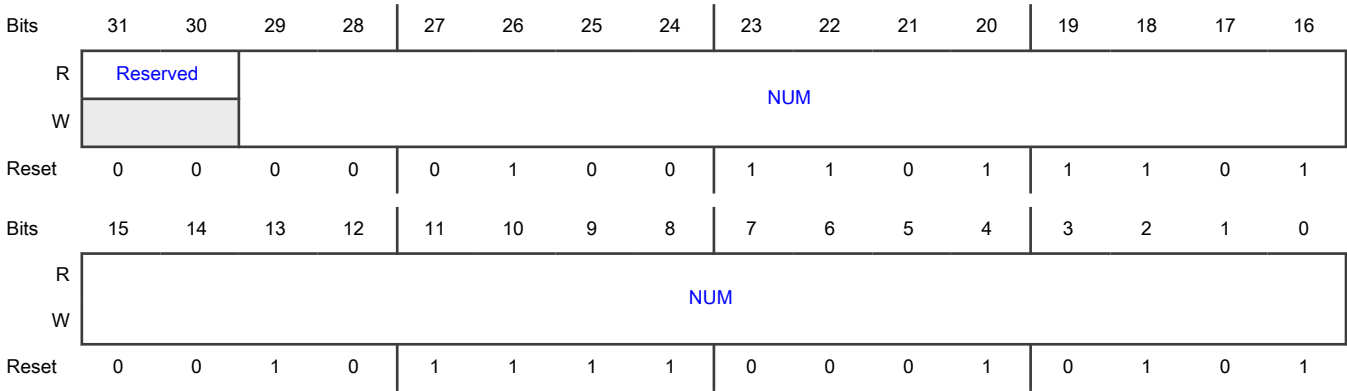
6.5.1.26 System PLL0 Numerator (SYSPLL0NUM)**Offset**

Register	Offset
SYSPLL0NUM	210h

Function

This register provides the numerator for the system PLL fractional divider.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-0 NUM	Numerator of the SYSPLL0 fractional loop divider The value of numerator must always be configured to be less than the value of the denominator. The SYSPLL0NUM register can only be changed when the SYSPLL0 is disabled. NOTE SYSPLL0CTL0[MULT] should be greater than or equal to 20 when SYSPLL0NUM[NUM] is not 0.

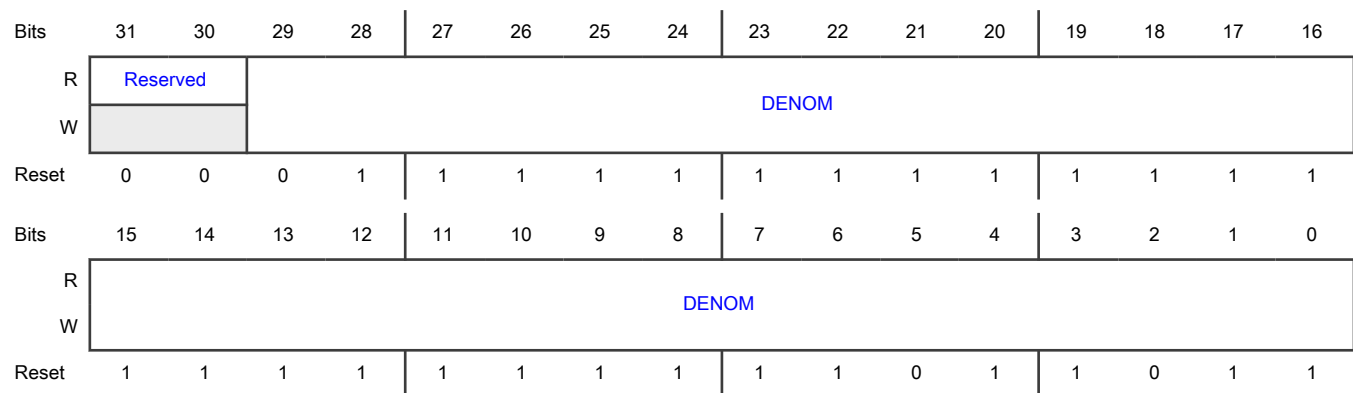
6.5.1.27 System PLL0 Denominator (SYSPLL0DENOM)

Offset

Register	Offset
SYSPLL0DENOM	214h

Function

This register provides the denominator for the system PLL fractional divider. For more information on PLL operation.

Diagram**Fields**

Field	Function
31-30 —	Reserved
29-0 DENOM	Denominator of the SYSPLL0 fractional loop divider The value of numerator must always be configured to be less than the value of the denominator. The SYSPLL0DENOM register can only be changed when the SYSPLL0 is disabled.

6.5.1.28 System PLL0 PFD (SYSPLL0PFD)**Offset**

Register	Offset
SYSPLL0PFD	218h

Function

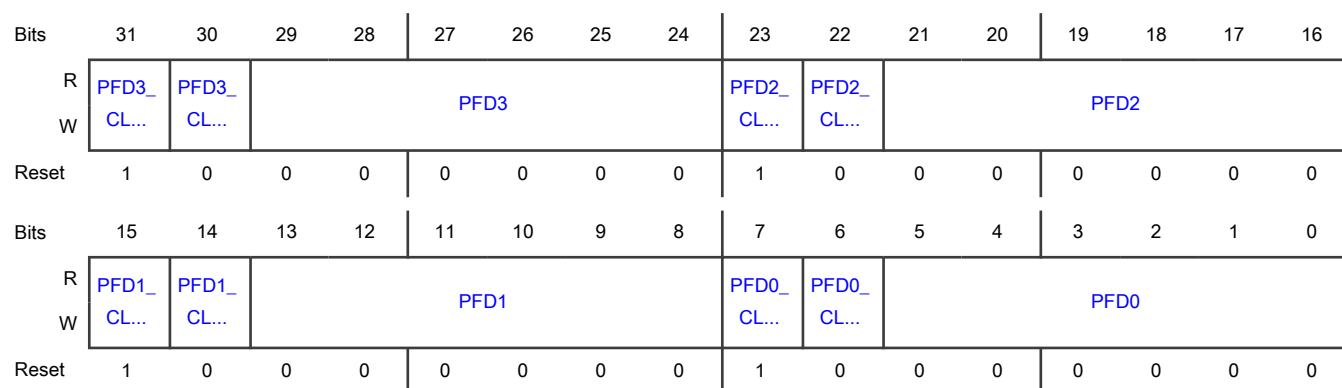
The SYSPLL0PFD register defines the control bits for the PFD0 - PFD3 clocks derived from the SYSPLL0. The PFD output frequencies are governed by:

$$F_{PFDx} = F_{SYSPLL0\_OUT} * (18 / PFDx)$$

When changing PFD values, follow this recommended sequence:

- Ensure that the PFDx clock gets gated, by writing '1' to the PFDx_CLKGATE bit,
- Program the new PFD value (which are limited to decimal values 12-35),
- Write '0' to the PFDx_CLKGATE register, to un-gate the PFDx clock,
- Poll the PFDx_CLKRDY status flag until it (the status flag) is set to '1'.
- Clear the PFDx_CLKRDY status flag, by writing '1' to that specific PFDx_CLKRDY bit
- The PFD outputs can be no higher than 600MHz.

Diagram



Fields

Field	Function
31 PFD3_CLKGATE	PFD3 Clock Gate 0b - PFD3 clock is not gated 1b - PFD3 clock is gated
30 PFD3_CLKRDY	PFD3 Clock Ready Status Flag Cleared by writing '1', which can only be done once. 0b - PFD3 clock is not ready 1b - PFD3 clock is ready
29-24 PFD3	PLL Fractional Divider 3 Controls the fractional divider value. Valid PFD values are 12 - 35 (decimal).
23 PFD2_CLKGATE	PFD2 Clock Gate 0b - PFD2 clock is not gated 1b - PFD2 clock is gated
22 PFD2_CLKRDY	PFD2 Clock Ready Status Flag Cleared by writing '1', which can only be done once. 0b - PFD2 clock is not ready 1b - PFD2 clock is ready
21-16 PFD2	PLL Fractional Divider 2 Controls the fractional divider value. Valid PFD values are 12 - 35 (decimal).
15 PFD1_CLKGATE	PFD1 Clock Gate 0b - PFD1 clock is not gated 1b - PFD1 clock is gated
14	PFD1 Clock Ready Status Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
PFD1_CLKRDY	Cleared by writing '1', which can only be done once. 0b - PFD1 clock is not ready 1b - PFD1 clock is ready
13-8 PFD1	PLL Fractional Divider 1 Controls the fractional divider value. Valid PFD values are 12 - 35 (decimal).
7 PFD0_CLKGATE	PFD0 Clock Gate 0b - PFD0 clock is not gated 1b - PFD0 clock is gated
6 PFD0_CLKRDY	PFD0 Clock Ready Status Flag Cleared by writing '1', which can only be done once. 0b - PFD0 clock is not ready 1b - PFD0 clock is ready
5-0 PFD0	PLL Fractional Divider 0 Controls the fractional divider value. Valid PFD values are 12 - 35 (decimal).

6.5.1.29 Main PLL Clock Divider (MAINPLLCLKDIV)

Offset

Register	Offset
MAINPLLCLKDIV	240h

Function

It is a clock divider register, dividing the Main PLL PFD0/PFD1/PFD2/PFD3 into main_pll_clk.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag It is set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts the divider counter The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets the divider counter It can be used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	Reserved
7-0 DIV	Low Frequency Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • . . . • 255: Divide by 256

6.5.1.30 DSP PLL Clock Divider (DSPPLLCLKDIV)

Offset

Register	Offset
DSPPLLCLKDIV	244h

Function

It is a clock divider register, dividing the Main PLL PFD0/PFD1/PFD2/PFD3 into dsp_pll_clk.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQFLAG	HALT	RESET	Reserved												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag It is set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts the divider counter The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets the divider counter It can be used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	Reserved
7-0 DIV	Low Frequency Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • . . . • 255: Divide by 256

6.5.1.31 AUX0 PLL Clock Divider (AUX0PLLCLKDIV)

Offset

Register	Offset
AUX0PLLCLKDIV	248h

Function

It is a clock divider register, dividing the Main PLL PFD0/PFD1/PFD2/PFD3 into aux0_pll_clk.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQFLAG	HALT	RESET	Reserved												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag It is set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts the divider counter The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets the divider counter It can be used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	Reserved
7-0 DIV	Low Frequency Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • . . . • 255: Divide by 256

6.5.1.32 AUX1 PLL Clock Divider (AUX1PLLCLKDIV)

Offset

Register	Offset
AUX1PLLCLKDIV	24Ch

Function

It is a clock divider register, dividing the Main PLL PFD0/PFD1/PFD2/PFD3 into aux1_pll_clk.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG		T													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag It is set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts the divider counter The intent is to allow the divider clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets the divider counter It can be used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	Reserved
7-0 DIV	Low Frequency Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • . . . • 255: Divide by 256

6.5.1.33 System CPU AHB Clock Divider (SYSCPUAHBCLKDIV)

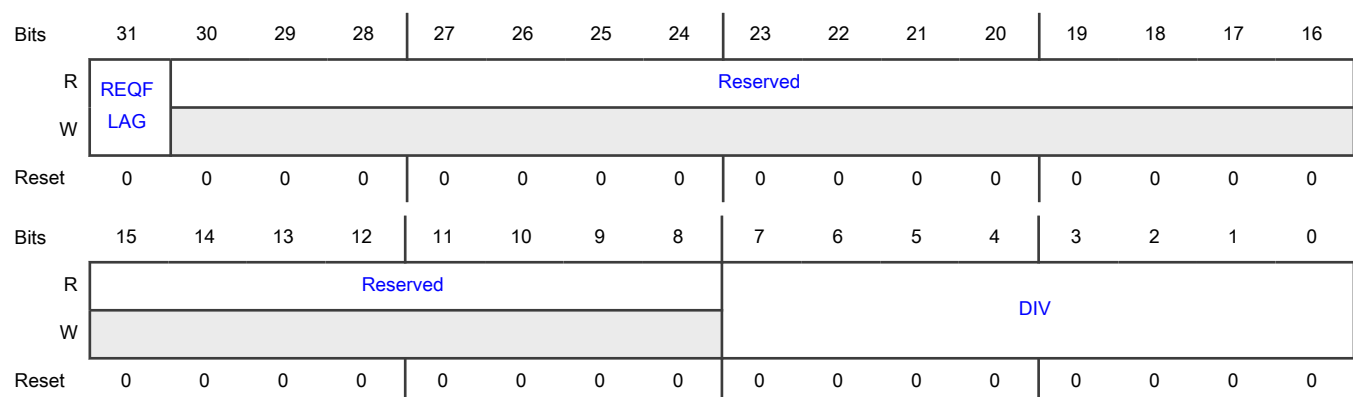
Offset

Register	Offset
SYSCPUAHBCLKDIV	400h

Function

This register controls how the main clock is divided to provide the system clock to the AHB bus, CPU, and memories.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.34 Main Clock Select A (MAINCLKSELA)**Offset**

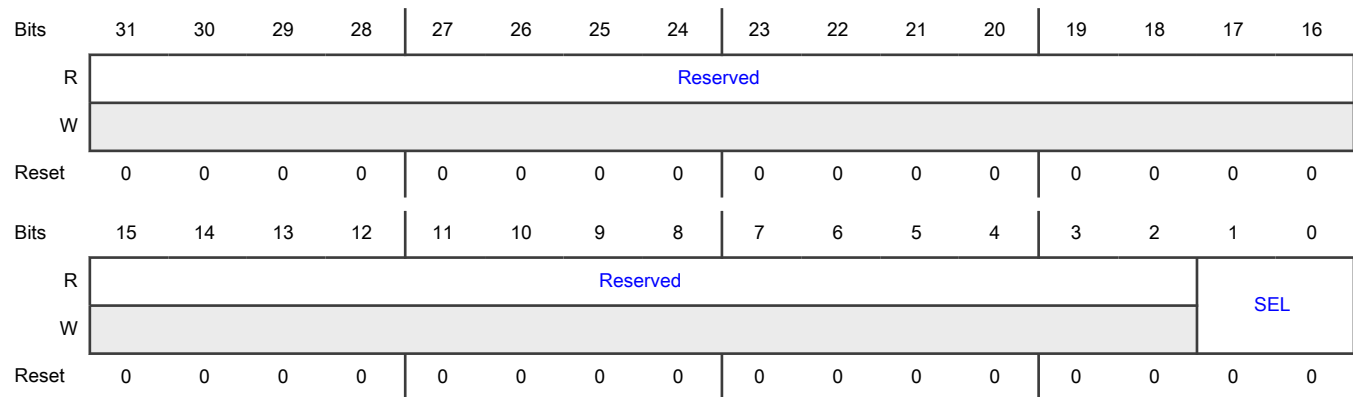
Register	Offset
MAINCLKSELA	430h

Function

This register selects one of several sources that become one of the inputs to the main clock source select register B, which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change.

Diagram**Fields**

Field	Function
31-2 —	Reserved
1-0 SEL	Control Main 1st Stage Control Clock Source 00b - Low Power Oscillator Clock (LPOSC) 01b - FRODIV which is the output of the FRODIVSEL mux 10b - OSC_CLK clock 11b - FRO_DIV1 clock

6.5.1.35 Main Clock Select B (MAINCLKSELB)**Offset**

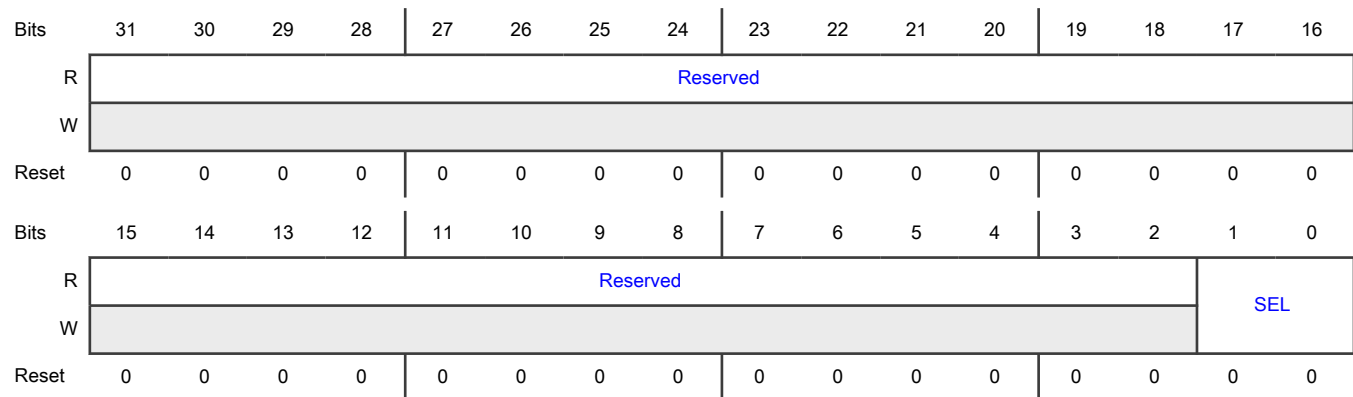
Register	Offset
MAINCLKSELB	434h

Function

Selects the clock source for the Main Clock.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change.

Diagram**Fields**

Field	Function
31-2 —	Reserved
1-0 SEL	Main Clock Source Selection 00b - MAINCLKSELA 1st Stage Clock 01b - Main System PLL Clock 10b - RTC 32 KHz Clock 11b - Reserved

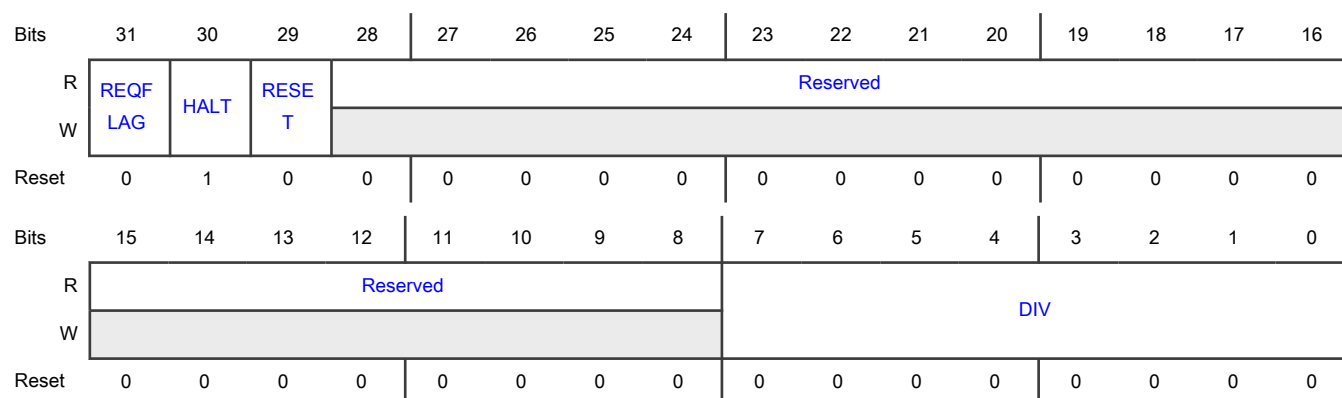
6.5.1.36 PFC divider 0 (trace clock) (PFC0DIV)**Offset**

Register	Offset
PFC0DIV	500h

Function

This register controls the clock divider for the trace function. The trace function can be slowed down to be operable on device pins when the CPU is running faster than the pins can accommodate. This is I/O voltage dependent, see device datasheet for details. The trace function clock is divided by two by the trace function in order to produce the output TRACECLK.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.37 PFC divider 1 (USB HS PHY bus clock) (PFC1DIV)

Offset

Register	Offset
PFC1DIV	504h

Function

This register controls the divider for the bus interface of the High Speed USB PHY. This clock must not be greater than 120 MHz.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF LAG	HALT	RESET	Reserved												
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter

Table continues on the next page...

Table continued from the previous page...

Field	Function
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.38 FlexSPI0 Functional Clock Select (FLEXSPI0CLKSEL)

Offset

Register	Offset
FLEXSPI0CLKSEL	620h

Function

This register selects the clock source for FlexSPI0.

Diagram

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

Fields

Field	Function
31-3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 SEL	Select Clock Source 000b - Main Clock 001b - Main System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - FRO_DIV1 Clock 100b - SYSPLL0 AUX1_PLL_Clock 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.39 FlexSPI0 Functional Clock Divider (FLEXSPI0FCLKDIV)

Offset

Register	Offset
FLEXSPI0FCLKDIV	624h

Function

This register controls the divider for FlexSPI0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • . . . • 255: Divide by 256

6.5.1.40 FlexSPI1 Functional Clock Select (FLEXSPI1FCLKSEL)

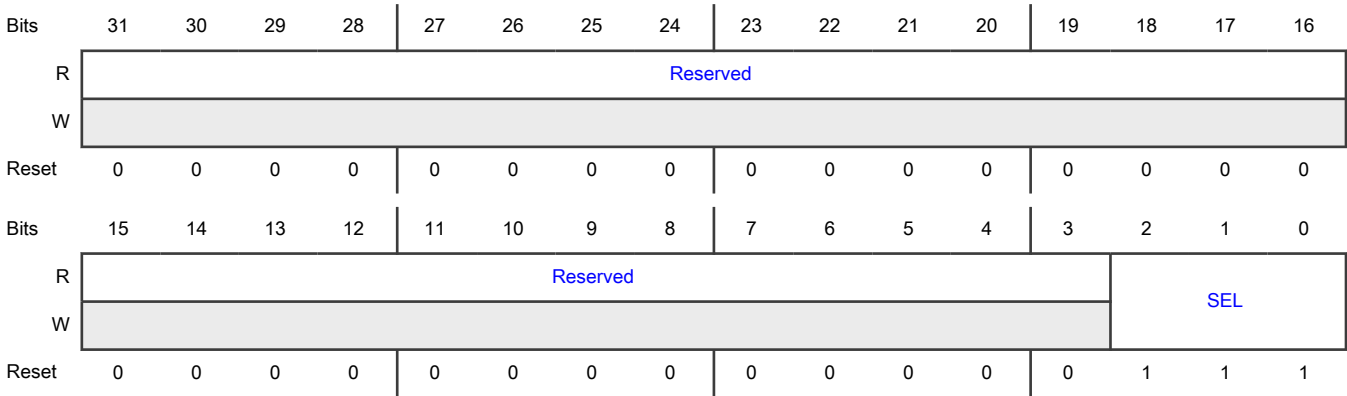
Offset

Register	Offset
FLEXSPI1FCLKSEL	630h

Function

This register selects the clock source for FlexSPI1.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Main Clock 001b - Main System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - FRO_DIV1 Clock 100b - SYSPLL0 AUX1_PLL_Clock 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.41 FlexSPI1 Functional Clock Divider (FLEXSPI1FCLKDIV)

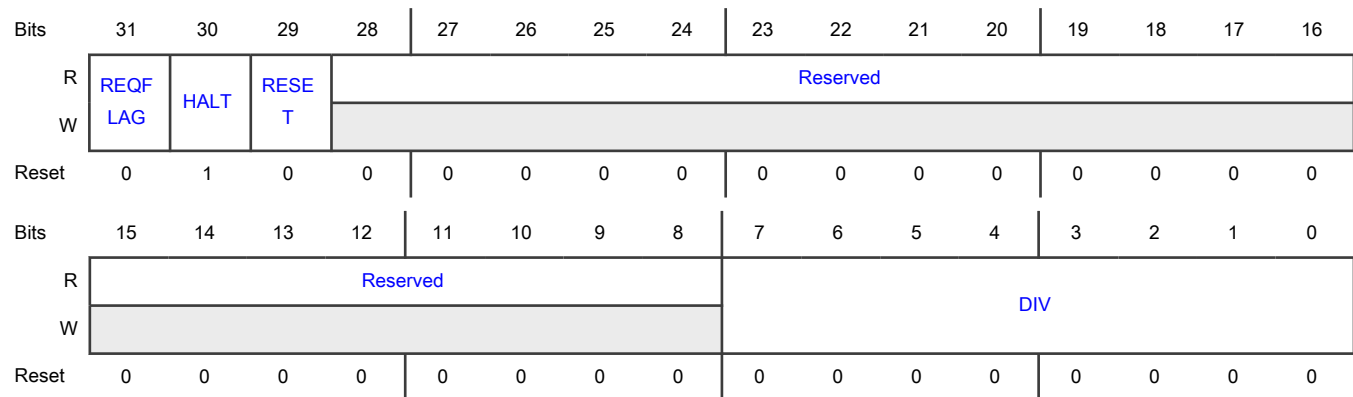
Offset

Register	Offset
FLEXSPI1FCLKDIV	634h

Function

This register controls the divider for FlexSPI1.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.42 SCT Functional Clock Select (SCTFCLKSEL)

Offset

Register	Offset
SCTFCLKSEL	640h

Function

Selects the SCT Functional clock, for SCT Input 7 clock only.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Main Clock 001b - Main System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - FRO_DIV1 Clock 100b - SYSPLL0 AUX1_PLL_Clock 101b - AUDIO PLL Clock 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.43 SCT Functional Clock Divider (SCTIN7CLKDIV)

Offset

Register	Offset
SCTIN7CLKDIV	644h

Function

This register controls the divider for the SCTimer/PWM input 7.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.44 High Speed USB Functional Clock Select (USBHSFCLKSEL)

Offset

Register	Offset
USBHSFCLKSEL	660h

Function

This register selects the clock source for the High Speed USB.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0	Select Clock Source

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEL	NOTE Main Clock can only be selected if it is sourced by either the osc_clk or the PLL sourced by the OSC_CLK. 000b - OSC_CLK Clock 001b - Main Clock 010b - Reserved 011b - AUX0_PLL_CLOCK 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.45 High Speed USB Functional Clock Divider (USBHSFCLKDIV)

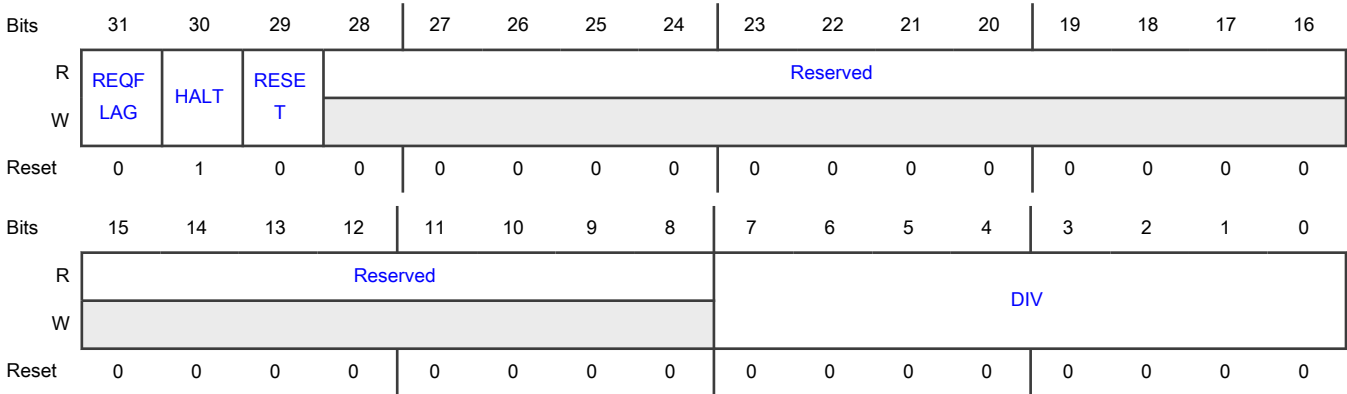
Offset

Register	Offset
USBHSFCLKDIV	664h

Function

This register controls the divider for the High Speed USB function clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

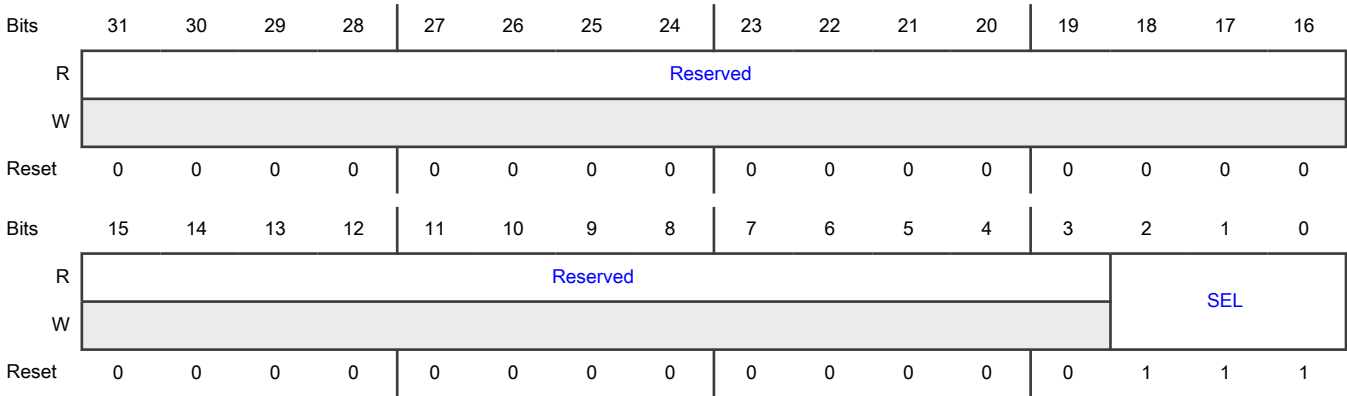
6.5.1.46 SDIO0 Functional Clock Select (SDIO0FCLKSEL)**Offset**

Register	Offset
SDIO0FCLKSEL	680h

Function

This register selects the clock source for the SD/MMC/SDIO interface 0 (uSDHC0).

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Main Clock 001b - System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - FRO_DIV2 100b - SYSPLL0 AUX1_PLL_Clock 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.47 SDIO0 Functional Clock Divider (SDIO0FCLKDIV)

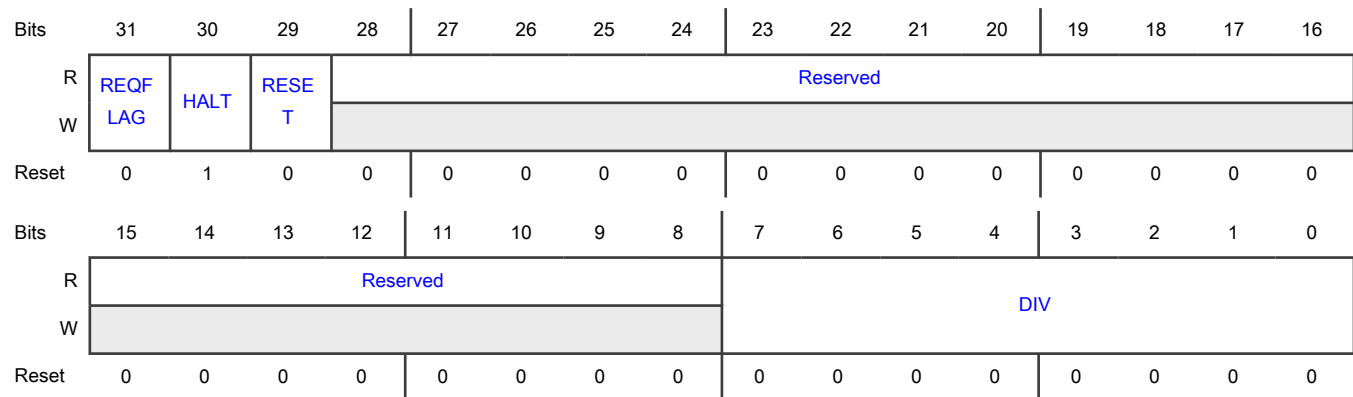
Offset

Register	Offset
SDIO0FCLKDIV	684h

Function

This register controls the divider for the SD/MMC/SDIO interface 0 (uSDHC0) function clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.48 SDIO1 Functional Clock Select (SDIO1FCLKSEL)

Offset

Register	Offset
SDIO1FCLKSEL	690h

Function

This register selects the clock source for the SD/MMC/SDIO interface 1 (uSDHC1).

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Main Clock 001b - Main System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - FRO_DIV2 100b - SYSPLL0 AUX1_PLL_Clock 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.49 SDIO1 Functional Clock Divider (SDIO1FCLKDIV)

Offset

Register	Offset
SDIO1FCLKDIV	694h

Function

This register controls the divider for the SD/MMC/SDIO interface 0 (uSDHC0) function clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.50 ADC0 Functional Clock Select 0 (ADC0FCLKSEL0)

Offset

Register	Offset
ADC0FCLKSEL0	6D0h

Function

This register selects one of several clock sources to be made available to the final ADC0 multiplexer.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0	Select Clock Source

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEL	000b - OSC_CLK Clock 001b - Low Power Oscillator Clock (LPOSC) 010b - FRO_DIV4 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.51 ADC0 Functional Clock Select 1 (ADC0FCLKSEL1)

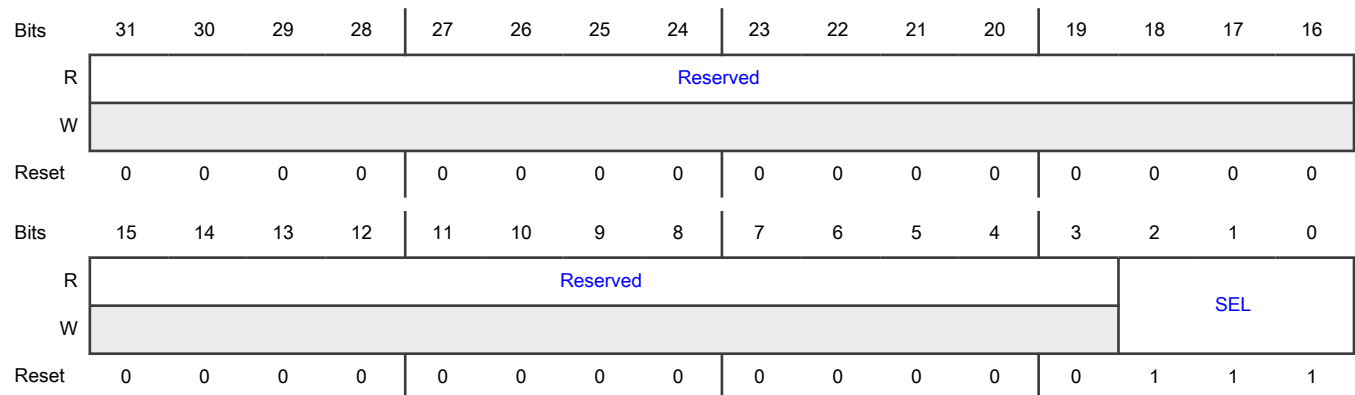
Offset

Register	Offset
ADC0FCLKSEL1	6D4h

Function

This register selects the clock source for ADC0.

Diagram



Fields

Field	Function
31-3	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 SEL	Select Clock Source 000b - ADC0CLKSEL0 Multiplexed Output 001b - SYSPLL0 MAIN_CLK (PFD0 Output) 010b - SYSPLL0 AUX0_PLL_Clock 011b - SYSPLL0 AUX1_PLL_Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.52 ADC0 Functional Clock Divider (ADC0CLKDIV)

Offset

Register	Offset
ADC0CLKDIV	6D8h

Function

This register controls the divider for the ADC0 function clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • . . . • 255: Divide by 256

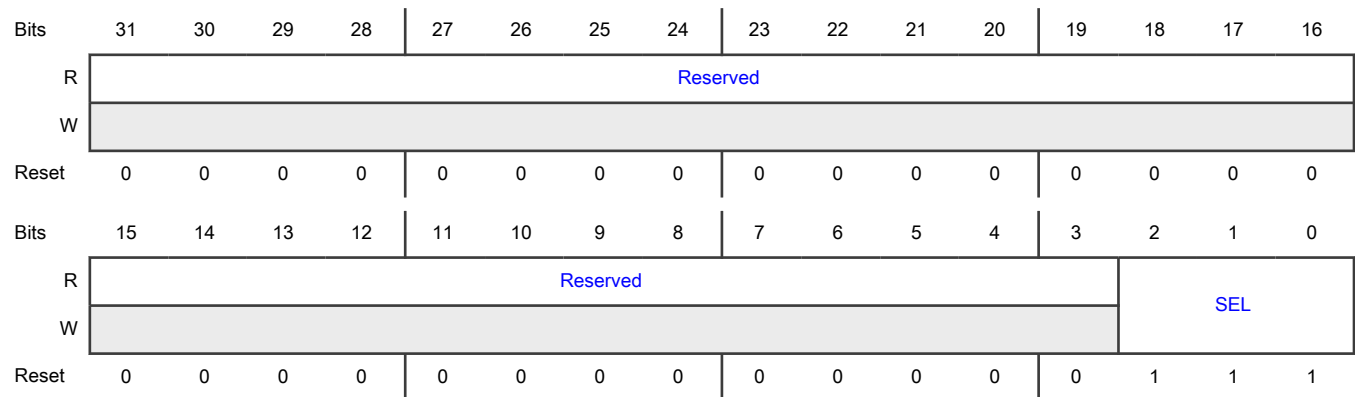
6.5.1.53 UTICK Functional Clock Select (UTICKFCLKSEL)

Offset

Register	Offset
UTICKFCLKSEL	700h

Function

This register selects the clock source for the UTICK timer.

Diagram**Fields**

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Low Power Oscillator Clock (LPOSC) 001b - Reserved 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

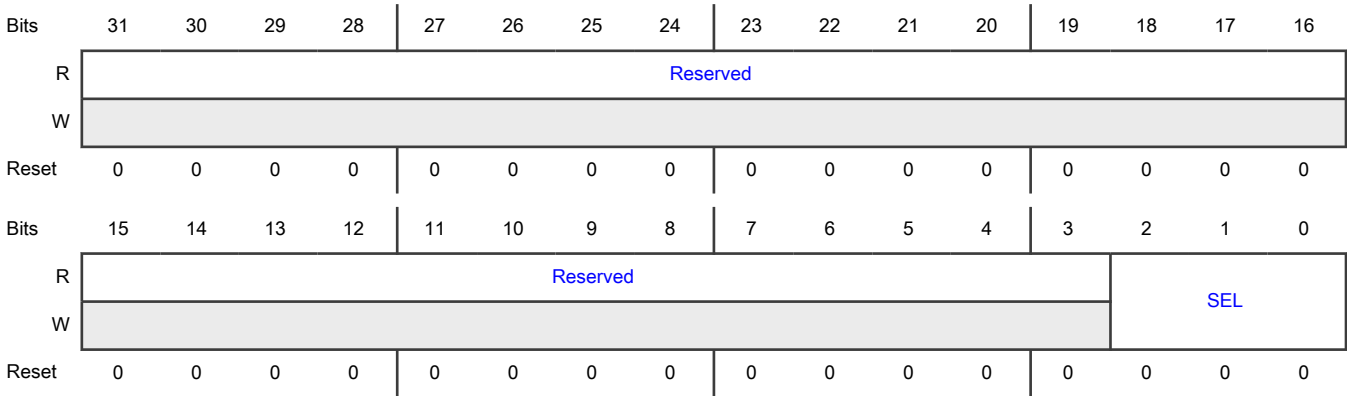
6.5.1.54 WDT0 Functional Clock Select (WDT0FCLKSEL)**Offset**

Register	Offset
WDT0FCLKSEL	720h

Function

This register selects the clock source for watchdog timer 0.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - Low Power Oscillator Clock (LPOSC) 001b - Reserved 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.55 32 KHz Wake Clock Source Select (A32KHZWAKECLKSEL)

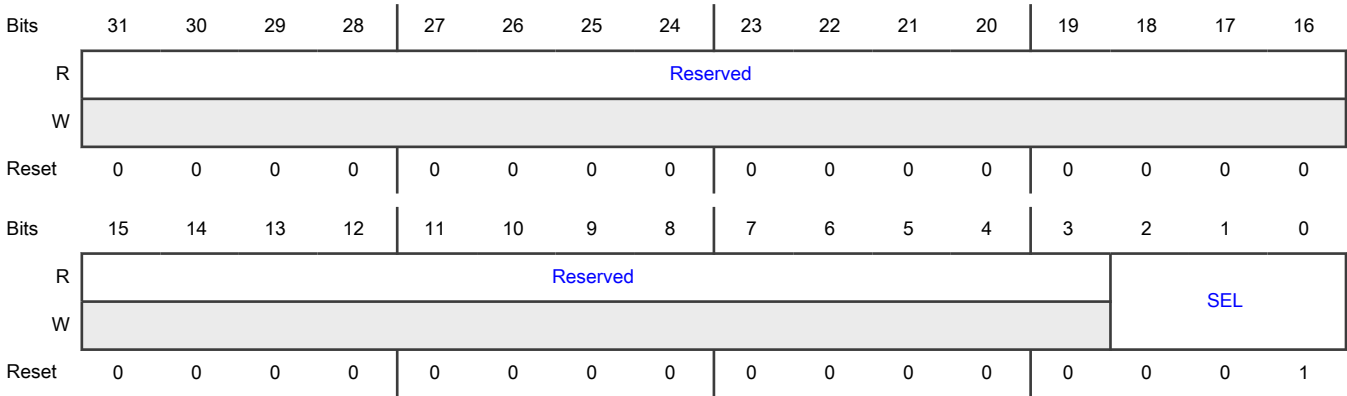
Offset

Register	Offset
A32KHZWAKECLKSEL	730h

Function

This register selects the clock source for the 32 kHz wake-up function.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - 32 KHz 001b - Low Power Oscillator Clock (LPOSC); divided by 32 by default 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.56 32 KHz Wake Clock Divider (A32KHZWAKECLKDIV)

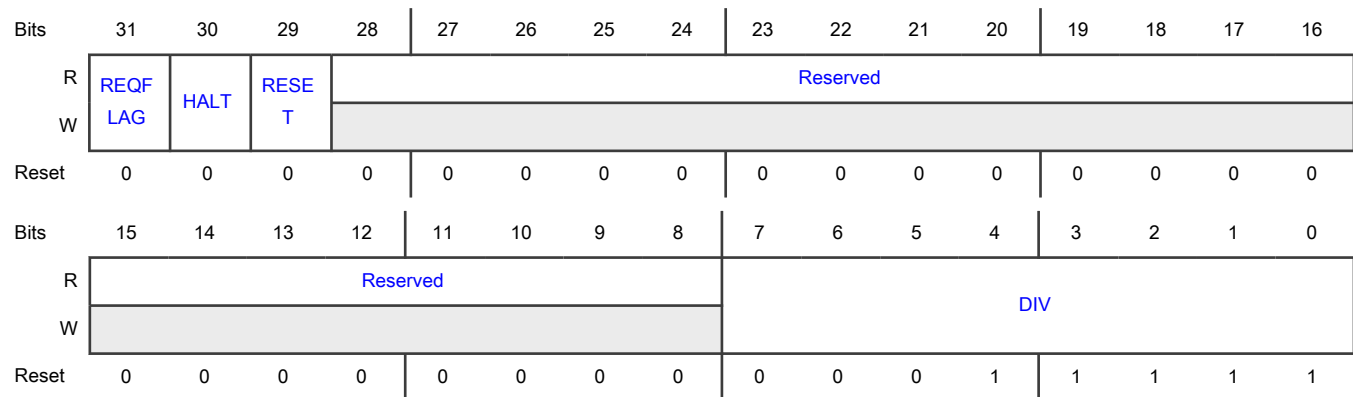
Offset

Register	Offset
A32KHZWAKECLKDIV	734h

Function

This register allows halting the divider when 1m_lposc is not used as the basis for the 32 kHz wake-up function.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.57 SYSTICK Functional Clock Select (SYSTICKFCLKSEL)

Offset

Register	Offset
SYSTICKFCLKSEL	760h

Function

This register selects the clock source for the SysTick timer.

This clock source is used, if the CLKSOURCE bit in System Timer Control and Status register is set to 0. Please refer Arm Cortex M33 Devices Generic User Guide, section 4.3 for more information on SysTick. The SYSTICK CLKSOURCE bit can be set to 1 to select the CPU clock as the SYSTICK clock source.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - SysTick Divider Output Clock 001b - Low Power Oscillator Clock (LPOSC) 010b - 32 KHz RTC Clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.58 SYSTICK Functional Clock Divider (SYSTICKFCLKDIV)

Offset

Register	Offset
SYSTICKFCLKDIV	764h

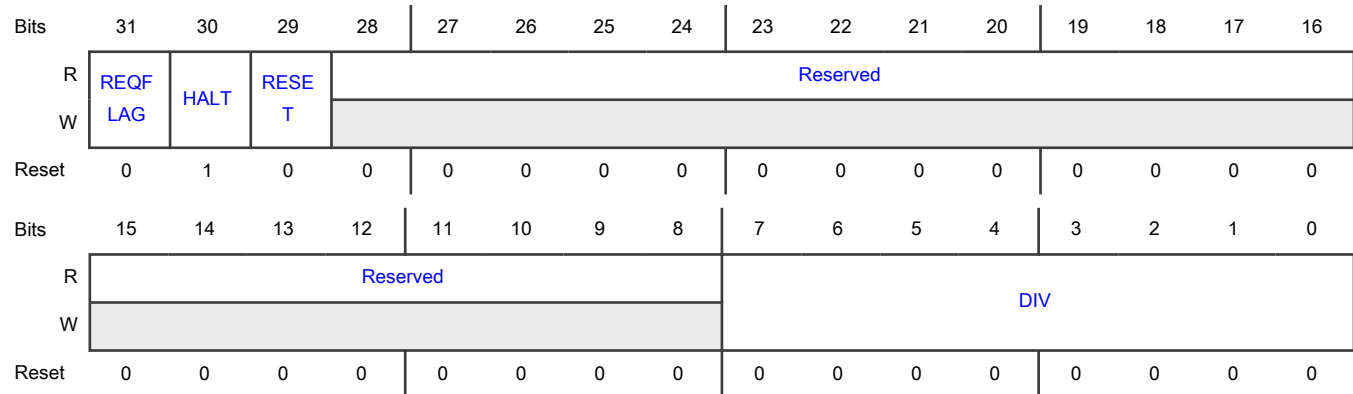
Function

This register controls the divider for the Systick function clock. This divides main_clk, for input to SYSTICKFCLKSEL.

NOTE

The clock frequency resulting from the Systick Clock Divider cannot be equal or higher than the CPU clock frequency resulting from the CPU Clock Divider for divide by 1 to work.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.59 MIPI-DSI PHY Clock Select (DPHYCLKSEL)

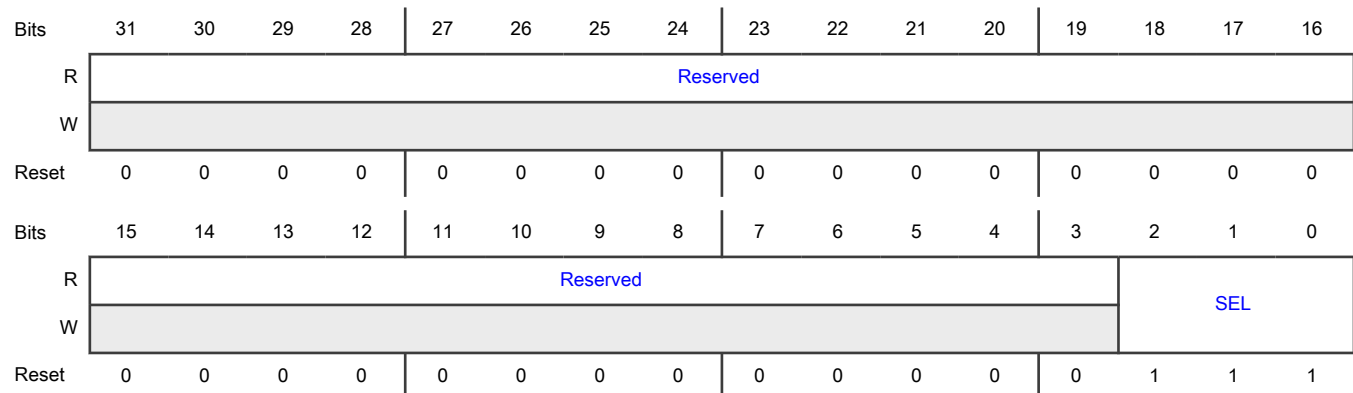
Offset

Register	Offset
DPHYCLKSEL	770h

Function

This register selects the clock source for the MIPI-DSI PHY Clock.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - FRO_DIV1 Clock 001b - SYSPLL0 MAIN_CLK (PFD0 Output) 010b - SYSPLL0 AUX0_PLL_Clock 011b - SYSPLL0 AUX1_PLL_Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.60 MIPI-DSI PHY Clock Divider (DPHYCLKDIV)

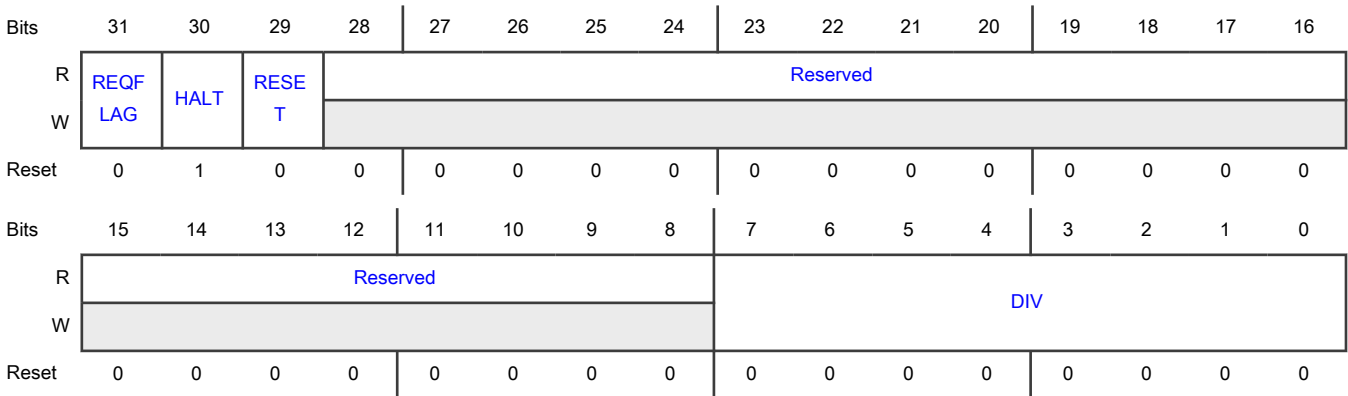
Offset

Register	Offset
DPHYCLKDIV	774h

Function

This register controls the divider for the MIPI-DSI PHY Clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

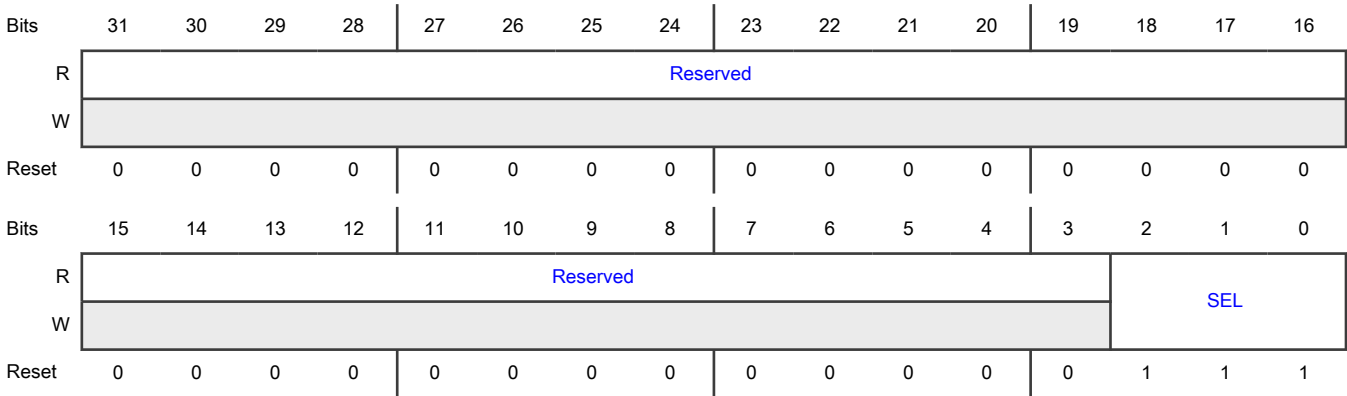
6.5.1.61 MIPI-DSI DPHY Escape Mode Clock Select (DPHYESCCLKSEL)**Offset**

Register	Offset
DPHYESCCLKSEL	778h

Function

This register selects the clock source for the MIPI-DSI DPHY Escape Mode Clock.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - FRO_DIV1 clock 001b - FRO_DIV16 Clock 010b - AUX0_PLL_CLK 011b - AUX1_PLL_CLK 100b - Reserved 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.62 MIPI-DSI DPHY Escape Mode Receive Clock Divider (DPHYESCRXCLKDIV)

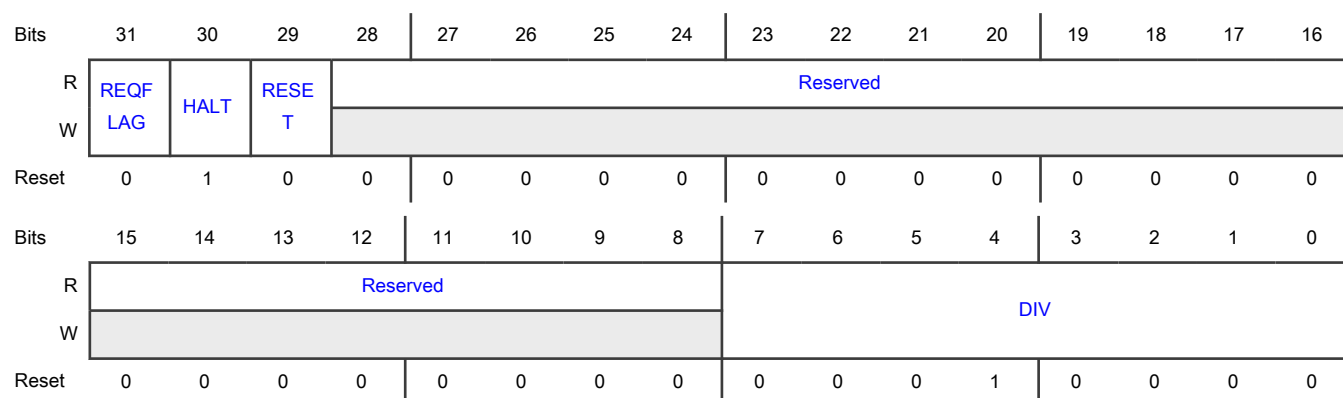
Offset

Register	Offset
DPHYESCRXCLKDIV	77Ch

Function

This register controls the divider for the MIPI-DSI DPHY Escape Mode Receive Clock divider.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.63 MIPI-DSI DPHY Escape Mode Transmit Clock Divider (DPHYESCTXCLKDIV)

Offset

Register	Offset
DPHYESCTXCLKDIV	780h

Function

This register controls the divider for the MIPI-DSI DPHY Escape Mode Transmit Clock divider.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.64 GPU Clock Select (GPUCLKSEL)

Offset

Register	Offset
GPUCLKSEL	790h

Function

This register selects the clock source for the GPU Clock.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0	Select Clock Source

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEL	000b - Main Clock 001b - FRO_DIV1 clock 010b - SYSPLL0 MAIN_CLK (PFD0 Output) 011b - SYSPLL0 AUX0_PLL_Clock 100b - SYSPLL0 AUX1_PLL_Clock 101b - Reserved 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

6.5.1.65 GPU Clock Divider (GPUCLKDIV)

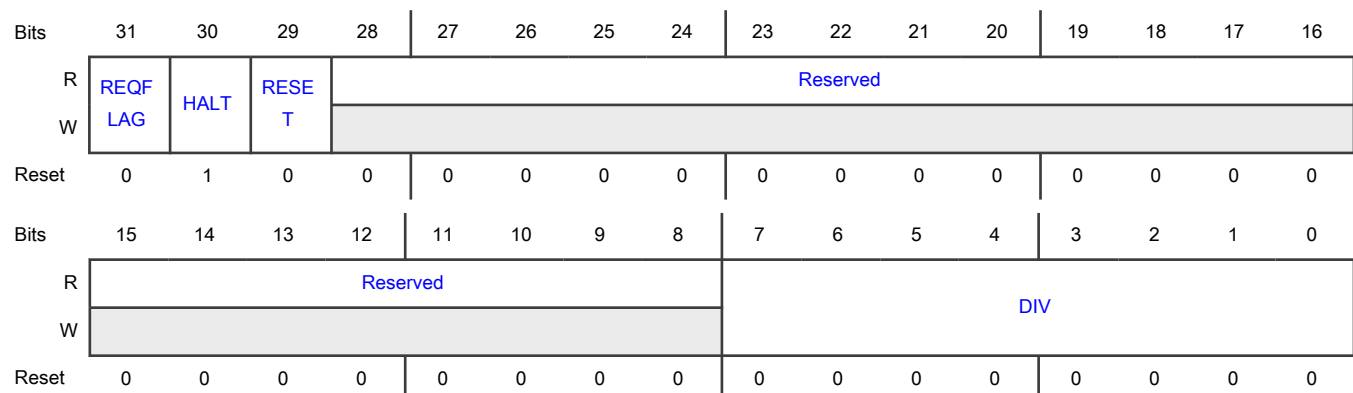
Offset

Register	Offset
GPUCLKDIV	794h

Function

This register controls the divider for the GPU Clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.1.66 LCDIF Pixel Clock Select (DCPIXELCLKSEL)

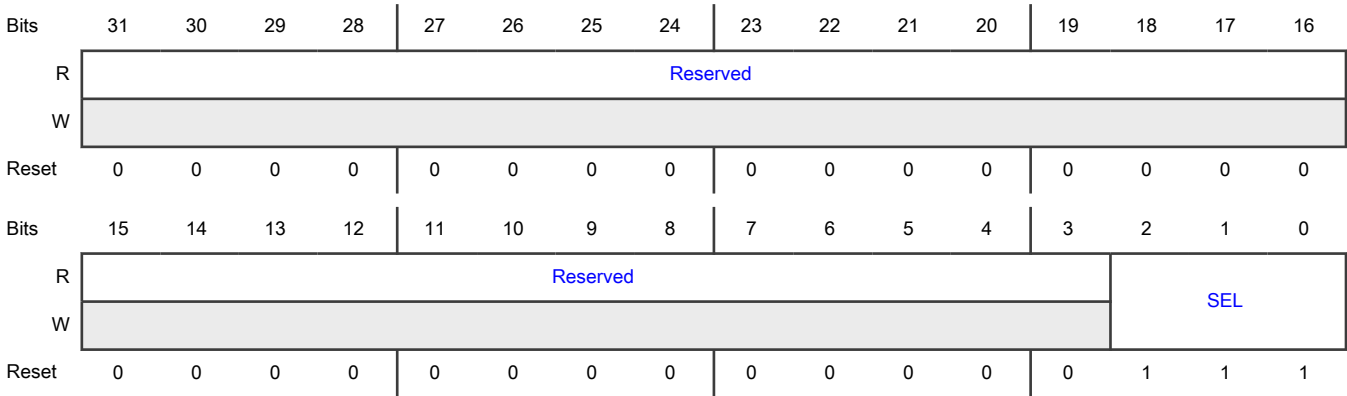
Offset

Register	Offset
DCPIXELCLKSEL	7A0h

Function

This register selects the clock source for the LCDIF Pixel Clock.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Select Clock Source 000b - MIPI-DSI PHY Clock 001b - Main Clock 010b - FRO_DIV1 Clock 011b - SYSPLL0 MAIN_CLK (PFD0 Output) 100b - SYSPLL0 AUX0_PLL_Clock 101b - SYSPLL0 AUX1_PLL_Clock 110b - Reserved 111b - None; this may be selected to reduce power when no output is needed.

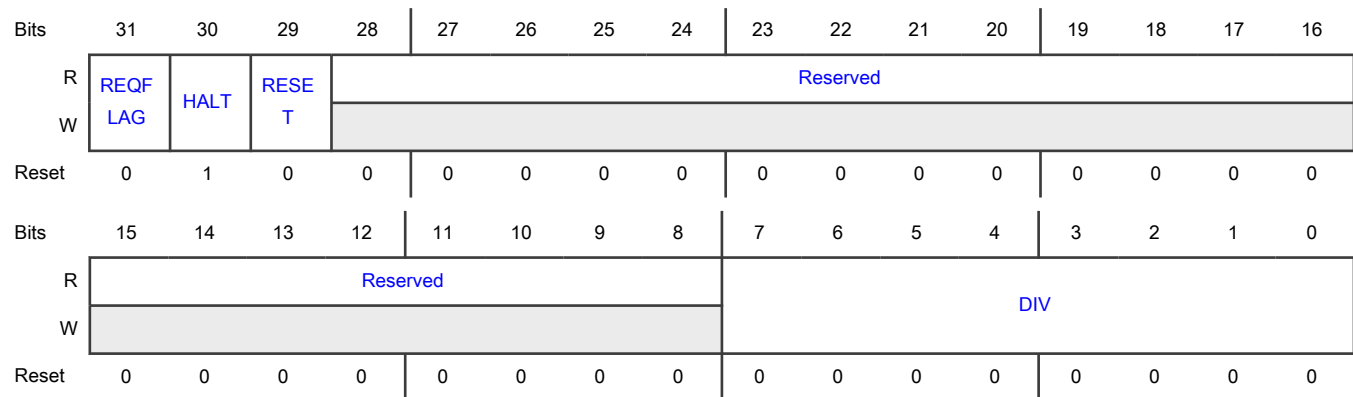
6.5.1.67 LCDIF Pixel Clock Divider (DCPIXELCLKDIV)

Offset

Register	Offset
DCPIXELCLKDIV	7A4h

Function

This register controls the divider for the LCDIF Pixel Clock.

Diagram**Fields**

Field	Function
31 REQFLAG	Divider status flag Clock being divided must be running for this status to change. 0b - The change to the divider value has finished 1b - A change is being made to the divider value
30 HALT	Halt the Divider Counter Stops the Divider Counter 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Divider Value Selection • 0: Divide by 1 • 1: Divide by 2 • 2: Divide by 3 • ... • 255: Divide by 256

6.5.2 Clock Controller 1 register descriptions

This section includes the memory map and register descriptions for Clock Controller 1.

6.5.2.1 CLKCTL1 memory map

CLKCTL1 base address: 4002_1000h

Offset	Register	Width (In bits)	Access	Reset value
10h	Clock Control 0 (PSCCTL0)	32	RW	0000_0000h
14h	Clock Control 1 (PSCCTL1)	32	RW	0000_0000h
18h	Clock Control 2 (PSCCTL2)	32	RW	0000_0000h
40h	Clock Set 0 (PSCCTL0_SET)	32	RW	See section
44h	Clock Set 1 (PSCCTL1_SET)	32	RW	See section
48h	Clock Set 2 (PSCCTL2_SET)	32	RW	See section
70h	Clock Clear 0 (PSCCTL0_CLR)	32	RW	See section
74h	Clock Clear 1 (PSCCTL1_CLR)	32	RW	See section
78h	Clock Clear 2 (PSCCTL2_CLR)	32	RW	See section
200h	Audio PLL0 Clock Select (AUDIOPLL0CLKSEL)	32	RW	0000_0007h
204h	Audio PLL0 Control 0 (AUDIOPLL0CTL0)	32	RW	0016_0002h
20Ch	Audio PLL0 Lock Time Divide-by-2 (AUDIOPLL0LOCKTIMEDIV2)	32	RW	0000_CAFeh
210h	Audio PLL0 Numerator (AUDIOPLL0NUM)	32	RW	04DD_2F15h
214h	Audio PLL0 Denominator (AUDIOPLL0DENOM)	32	RW	1FFF_FFDBh
218h	Audio PLL0 PFD (AUDIOPLL0PFD)	32	RW	8080_8080h
240h	Audio PLL Clock Divider (AUDIOPLLCLKDIV)	32	RW	4000_0000h
400h	DSP CPU Clock Divider (DSPCPUCLKDIV)	32	RW	4000_0000h
430h	DSP CPU Clock Select A (DSPCPUCLKSELA)	32	RW	0000_0000h
434h	DSP CPU Clock Select B (DSPCPUCLKSELB)	32	RW	0000_0000h
480h	OS Event Timer Functional Clock Select (OSEVENTTFCLKSEL)	32	RW	0000_0000h
500h	Fractional Rate Generator 0 Clock Select (FRG0CLKSEL)	32	RW	0000_0007h
504h	Fractional Rate Generator 0 Control (FRG0CTL)	32	RW	0000_00FFh
508h	Flexcomm0 Clock Select (FC0FCLKSEL)	32	RW	0000_0007h
520h	Fractional Rate Generator 1 Clock Select (FRG1CLKSEL)	32	RW	0000_0007h
524h	Fractional Rate Generator 1 Control (FRG1CTL)	32	RW	0000_00FFh
528h	Flexcomm1 Clock Select (FC1FCLKSEL)	32	RW	0000_0007h
540h	Fractional Rate Generator 2 Clock Select (FRG2CLKSEL)	32	RW	0000_0007h
544h	Fractional Rate Generator 2 Control (FRG2CTL)	32	RW	0000_00FFh
548h	Flexcomm2 Clock Select (FC2FCLKSEL)	32	RW	0000_0007h
560h	Fractional Rate Generator 3 Clock Select (FRG3CLKSEL)	32	RW	0000_0007h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
564h	Fractional Rate Generator 3 Control (FRG3CTL)	32	RW	0000_00FFh
568h	Flexcomm3 Clock Select (FC3FCLKSEL)	32	RW	0000_0007h
580h	Fractional Rate Generator 4 Clock Select (FRG4CLKSEL)	32	RW	0000_0007h
584h	Fractional Rate Generator 4 Control (FRG4CTL)	32	RW	0000_00FFh
588h	Flexcomm4 Clock Select (FC4FCLKSEL)	32	RW	0000_0007h
5A0h	Fractional Rate Generator 5 Clock Select (FRG5CLKSEL)	32	RW	0000_0007h
5A4h	Fractional Rate Generator 5 Control (FRG5CTL)	32	RW	0000_00FFh
5A8h	Flexcomm5 Clock Select (FC5FCLKSEL)	32	RW	0000_0007h
5C0h	Fractional Rate Generator 6 Clock Select (FRG6CLKSEL)	32	RW	0000_0007h
5C4h	Fractional Rate Generator 6 Control (FRG6CTL)	32	RW	0000_00FFh
5C8h	Flexcomm6 Clock Select (FC6FCLKSEL)	32	RW	0000_0007h
5E0h	Fractional Rate Generator 7 Clock Select (FRG7CLKSEL)	32	RW	0000_0007h
5E4h	Fractional Rate Generator 7 Control (FRG7CTL)	32	RW	0000_00FFh
5E8h	Flexcomm7 Clock Select (FC7FCLKSEL)	32	RW	0000_0007h
600h	Fractional Rate Generator 8 Clock Select (FRG8CLKSEL)	32	RW	0000_0007h
604h	Fractional Rate Generator 8 Control (FRG8CTL)	32	RW	0000_00FFh
608h	Flexcomm8 Clock Select (FC8FCLKSEL)	32	RW	0000_0007h
620h	Fractional Rate Generator 9 Clock Select (FRG9CLKSEL)	32	RW	0000_0007h
624h	Fractional Rate Generator 9 Control (FRG9CTL)	32	RW	0000_00FFh
628h	Flexcomm9 Clock Select (FC9FCLKSEL)	32	RW	0000_0007h
640h	Fractional Rate Generator 10 Clock Select (FRG10CLKSEL)	32	RW	0000_0007h
644h	Fractional Rate Generator 10 Control (FRG10CTL)	32	RW	0000_00FFh
648h	Flexcomm10 Clock Select (FC10FCLKSEL)	32	RW	0000_0007h
660h	Fractional Rate Generator 11 Clock Select (FRG11CLKSEL)	32	RW	0000_0007h
664h	Fractional Rate Generator 11 Control (FRG11CTL)	32	RW	0000_00FFh
668h	Flexcomm11 Clock Select (FC11FCLKSEL)	32	RW	0000_0007h
680h	Fractional Rate Generator 12 Clock Select (FRG12CLKSEL)	32	RW	0000_0007h
684h	Fractional Rate Generator 12 Control (FRG12CTL)	32	RW	0000_00FFh
688h	Flexcomm12 Clock Select (FC12FCLKSEL)	32	RW	0000_0007h
6A0h	Fractional Rate Generator 13 Clock Select (FRG13CLKSEL)	32	RW	0000_0007h
6A4h	Fractional Rate Generator 13 Control (FRG13CTL)	32	RW	0000_00FFh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
6A8h	Flexcomm13 Clock Select (FC13CLKSEL)	32	RW	0000_0007h
6C0h	Fractional Rate Generator 14 Clock Select (FRG14CLKSEL)	32	RW	0000_0007h
6C4h	Fractional Rate Generator 14 Control (FRG14CTL)	32	RW	0000_00FFh
6C8h	Flexcomm14 Clock Select (FC14CLKSEL)	32	RW	0000_0007h
6E0h	Fractional Rate Generator 15 Clock Select (FRG15CLKSEL)	32	RW	0000_0007h
6E4h	Fractional Rate Generator 15 Control (FRG15CTL)	32	RW	0000_00FFh
6E8h	Flexcomm15 Clock Select (FC15CLKSEL)	32	RW	0000_0007h
700h	Fractional Rate Generator 16 Clock Select (FRG16CLKSEL)	32	RW	0000_0007h
704h	Fractional Rate Generator 16 Control (FRG16CTL)	32	RW	0000_00FFh
708h	Flexcomm16 Clock Select (FC16CLKSEL)	32	RW	0000_0007h
720h	Fractional Rate Generator 17 Clock Select (FRG17CLKSEL)	32	RW	0000_0007h
724h	Fractional Rate Generator 17 Control (FRG17CTL)	32	RW	0000_00FFh
728h	FlexIO Clock Select (FLEXIOCLKSEL)	32	RW	0000_0007h
740h	FlexIO Clock Divider (FLEXIOCLKDIV)	32	RW	4000_0000h
760h	Fractional Rate Generator PLL Clock Divider (FRGPLLCLKDIV)	32	RW	4000_0000h
780h	DMIC0 Functional Clock Select (DMIC0FCLKSEL)	32	RW	0000_0007h
784h	DMIC0 Functional Clock Divider (DMIC0FCLKDIV)	32	RW	4000_0000h
7A0h	CT32BIT bit timer 0 Functional Clock Select (CT32BIT0FCLKSEL)	32	RW	0000_0007h
7A4h	CT32BIT bit timer 1 Functional Clock Select (CT32BIT1FCLKSEL)	32	RW	0000_0007h
7A8h	CT32BIT bit timer 2 Functional Clock Select (CT32BIT2FCLKSEL)	32	RW	0000_0007h
7ACh	CT32BIT bit timer 3 Functional Clock Select (CT32BIT3FCLKSEL)	32	RW	0000_0007h
7B0h	CT32BIT bit timer 4 Functional Clock Select (CT32BIT4FCLKSEL)	32	RW	0000_0007h
7C0h	Audio MCLK Clock Select (AUDIOMCLKSEL)	32	RW	0000_0007h
7C4h	Audio MCLK Clock Divider (AUDIOMCLKDIV)	32	RW	4000_0000h
7E0h	CLKOUT Clock Select 0 (CLKOUTSEL0)	32	RW	0000_0007h
7E4h	CLKOUT Clock Select 1 (CLKOUTSEL1)	32	RW	0000_0007h
7E8h	CLKOUT Functional Clock Divider (CLKOUTFCLKDIV)	32	RW	4000_0000h
800h	I3C0, I3C1 Functional Clock Select (I3C01FCLKSEL)	32	RW	0000_0007h
804h	I3C0, I3C1 Functional Slow Time Control Clock Select (I3C01FCLKSTCSEL)	32	RW	0000_0007h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
808h	I3C0, I3C1 Functional Slow Time Control Clock Divider (I3C01FCLKSTCDIV)	32	RW	4000_0000h
80Ch	I3C0, I3C1 Functional Slow Clock Divider (I3C01FCLKSDIV)	32	RW	4000_0000h
810h	I3C0, I3C1 Functional Clock Divider (I3C01FCLKDIV)	32	RW	4000_0000h
814h	I3C01 Functional Clock Select (I3C01FCLKSTSTCLKSEL)	32	RW	0000_0000h
820h	Watchdog Timer 1 Functional Clock Select (WDT1FCLKSEL)	32	RW	0000_0000h
830h	Analog Comparator 0 Clock Select (ACMP0FCLKSEL)	32	RW	0000_0007h
834h	Analog comparator 0 FCLK divider (ACMP0FCLKDIV)	32	RW	4000_0000h

6.5.2.2 Clock Control 0 (PSCCTL0)

Offset

Register	Offset
PSCCTL0	10h

Function

This register enables the clocks to individual system and peripheral blocks.

When making changes to a PSCCTLn register, use the related PSCCTLn_SET and PSCCTLn_CLR registers, to avoid the unintended setting or clearing of other bits.

NOTE

All bits are reset by all reset sources.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		FlexIO	Reserv ed	OSEV ENT...	Reserv ed	FC16_ SP...	DMIC0	FC15_ I2...	FC14_ SP...	FC13_ CLK	FC12_ CLK	FC11_ CLK	FC10_ CLK	FC9_ CLK	FC8_ CLK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FC7_ CLK	FC6_ CLK	FC5_ CLK	FC4_ CLK	FC3_ CLK	FC2_ CLK	FC1_ CLK	FC0_ CLK	Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29 FlexIO	FlexIO clock control 0b - Disable Clock 1b - Enable Clock
28 —	Reserved
27 OSEVENT_TIMER	OS event timer bus clock control 0b - Disable Clock 1b - Enable Clock
26 —	Reserved
25 FC16_SPI_CLK	Flexcomm Interface 16 SPI clock control 0b - Disable Clock 1b - Enable Clock
24 DMIC0	DMIC0 clock control 0b - Disable Clock 1b - Enable Clock
23 FC15_I2C_CLK	Flexcomm Interface 15 I2C clock control 0b - Disable Clock 1b - Enable Clock
22 FC14_SPI_CLK	Flexcomm Interface 14 SPI clock control 0b - Disable Clock 1b - Enable Clock
21-8 FCn_CLK	Flexcomm Interface n clock control 0b - Disable Clock 1b - Enable Clock
7-0 —	Reserved

6.5.2.3 Clock Control 1 (PSCCTL1)

Offset

Register	Offset
PSCCTL1	14h

Function

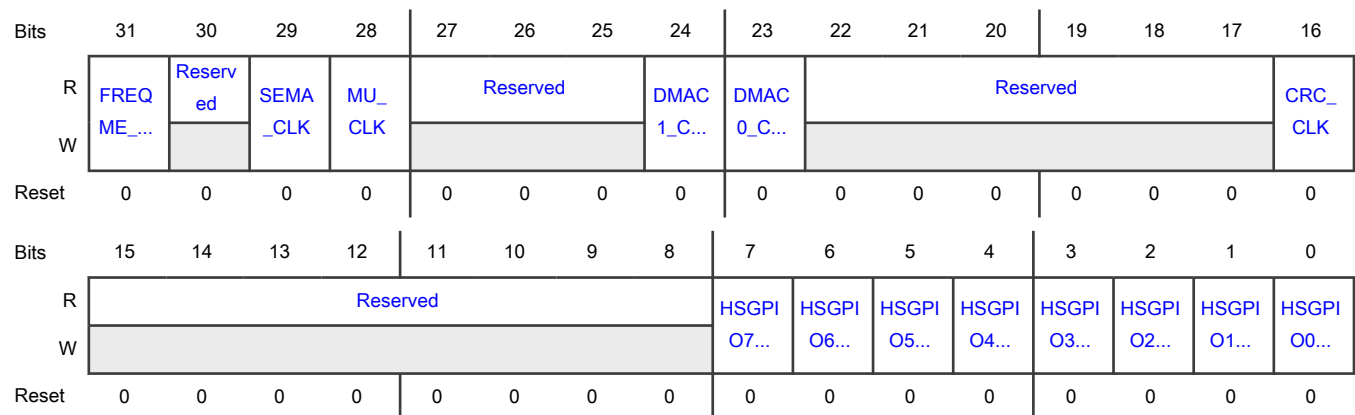
This register enables the clocks to individual system and peripheral blocks.

When making changes to a PSCCTLn register, use the related PSCCTLn_SET and PSCCTLn_CLR registers, to avoid the unintended setting or clearing of other bits.

NOTE

All bits are reset by all reset sources.

Diagram



Fields

Field	Function
31 FREQME_CLK	Frequency Measurement clock control 0b - Disable Clock 1b - Enable Clock
30 —	Reserved
29 SEMA_CLK	Semaphore clock control 0b - Disable Clock 1b - Enable Clock
28 MU_CLK	Messaging Unit clock control

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable Clock 1b - Enable Clock
27-25 —	Reserved
24 DMAC1_CLK	DMAC1 clock control 0b - Disable Clock 1b - Enable Clock
23 DMAC0_CLK	DMAC0 clock control 0b - Disable Clock 1b - Enable Clock
22-17 —	Reserved
16 CRC_CLK	CRC clock control 0b - Disable Clock 1b - Enable Clock
15-8 —	Reserved
7-0 HSGPION_CLK	Non-secure GPION clock control 0b - Disable Clock 1b - Enable Clock

6.5.2.4 Clock Control 2 (PSCCTL2)

Offset

Register	Offset
PSCCTL2	18h

Function

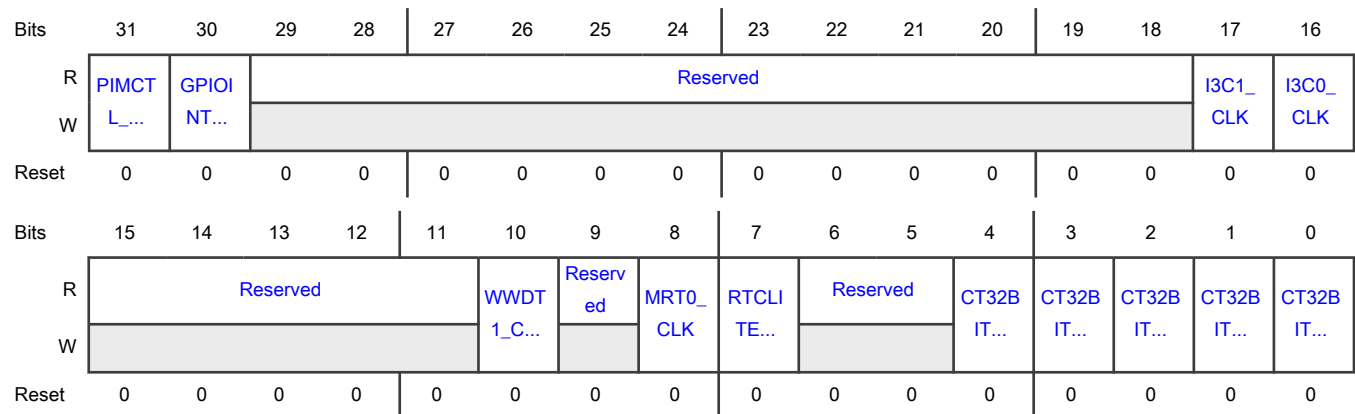
This register enables the clocks to individual system and peripheral blocks.

When making changes to a PSCCTLn register, use the related PSCCTLn_SET and PSCCTLn_CLR registers, to avoid the unintended setting or clearing of other bits.

NOTE

All bits are reset by all reset sources.

Diagram



Fields

Field	Function
31 PIMCTL_CLK	INPUTMUX clock control 0b - Disable Clock 1b - Enable Clock
30 GPIOINTCTL_CLK	PINT clock control 0b - Disable Clock 1b - Enable Clock
29-18 —	Reserved
17 I3C1_CLK	I3C1 clock control 0b - Disable Clock 1b - Enable Clock
16 I3C0_CLK	I3C0 clock control 0b - Disable Clock 1b - Enable Clock
15-11 —	Reserved
10 WWDT1_CLK	Watchdog Timer 1 clock control 0b - Disable Clock 1b - Enable Clock
9 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 MRT0_CLK	Multi-Rate Timer 0 clock control 0b - Disable Clock 1b - Enable Clock
7 RTCLITE_CLK	RTC clock control 0b - Disable Clock 1b - Enable Clock
6-5 —	Reserved
4 CT32BIT4_CLK	CT32BIT bit timer 4 clock control 0b - Disable Clock 1b - Enable Clock
3 CT32BIT3_CLK	CT32BIT bit timer 3 clock control 0b - Disable Clock 1b - Enable Clock
2 CT32BIT2_CLK	CT32BIT bit timer 2 clock control 0b - Disable Clock 1b - Enable Clock
1 CT32BIT1_CLK	CT32BIT bit timer 1 clock control 0b - Disable Clock 1b - Enable Clock
0 CT32BIT0_CLK	CT32BIT bit timer 0 clock control 0b - Disable Clock 1b - Enable Clock

6.5.2.5 Clock Set 0 (PSCCTL0_SET)

Offset

Register	Offset
PSCCTL0_SET	40h

Function

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved			FlexIO	Reserv ed	OSEV ENT...	Reserv ed	FC16_ SP...	DMIC0	FC15_ I2...	FC14_ SP...	FC13_ CLK	FC12_ CLK	FC11_ CLK	FC10_ CLK	FC9_ CLK	FC8_ CLK
W																	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FC7_ CLK	FC6_ CLK	FC5_ CLK	FC4_ CLK	FC3_ CLK	FC2_ CLK	FC1_ CLK	FC0_ CLK	Reserved							
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-30 —	Reserved
29 FlexIO	FlexIO clock control 0b - No effect 1b - Sets the PSCCTL0 bit
28 —	Reserved
27 OSEVENT_TIMER	OS event timer bus clock control 0b - No effect 1b - Sets the PSCCTL0 bit
26 —	Reserved
25 FC16_SPI_CLK	Flexcomm Interface 16 SPI clock control 0b - No effect 1b - Sets the PSCCTL0 bit
24 DMIC0	DMIC0 clock control 0b - No effect 1b - Sets the PSCCTL0 bit
23 FC15_I2C_CLK	Flexcomm Interface 15 I2C clock control 0b - No effect 1b - Sets the PSCCTL0 bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 FC14_SPI_CLK	Flexcomm Interface 14 SPI clock control 0b - No effect 1b - Sets the PSCCTL0 bit
21-8 FCn_CLK	Flexcomm Interface n clock set 0b - No effect 1b - Sets the PSCCTL0 bit
7-0 —	Reserved

6.5.2.6 Clock Set 1 (PSCCTL1_SET)

Offset

Register	Offset
PSCCTL1_SET	44h

Function

Use to set the clock enable bits in PSCCTL1 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FREQ	Reserv	SEMA	MU_	Reserved				DMAC	DMAC	Reserved				CRC_	
W	ME_...		_CLK	CLK					1_C...	0_C...					CLK	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI
W									O7...	O6...	O5...	O4...	O3...	O2...	O1...	O0...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 FREQME_CLK	Frequency Measurement clock control set 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the PSCCTL1 bit
30 —	Reserved
29 SEMA_CLK	Semaphore clock control set 0b - No effect 1b - Sets the PSCCTL1 bit
28 MU_CLK	Messaging Unit clock control set 0b - No effect 1b - Sets the PSCCTL1 bit
27-25 —	Reserved
24 DMAC1_CLK	DMAC1 clock control set 0b - No effect 1b - Sets the PSCCTL1 bit
23 DMAC0_CLK	DMAC0 clock control set 0b - No effect 1b - Sets the PSCCTL1 bit
22-17 —	Reserved
16 CRC_CLK	CRC clock control set 0b - No effect 1b - Sets the PSCCTL1 bit
15-8 —	Reserved
7-0 HSGPION_CLK	Non-secure GPION clock control set 0b - No effect 1b - Sets the PSCCTL1 bit

6.5.2.7 Clock Set 2 (PSCCTL2_SET)

Offset

Register	Offset
PSCCTL2_SET	48h

Function

Writing a 1 to a bit position in this register sets the corresponding position in PSCCTL2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIMCT	GPIOI	Reserved												I3C1_	I3C0_
W	L...	NT...													CLK	CLK
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				WWD1		Reserv	MRT0_	RTCLI	Reserved						
W					1_C...			CLK	TE...			CT32B	CT32B	CT32B	CT32B	CT32B
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 PIMCTL_CLK	INPUTMUX clock control set 0b - Disable Clock 1b - Enable Clock
30 GPIOINTCTL_CLK	PINT clock control set 0b - Disable Clock 1b - Enable Clock
29-18 —	Reserved
17 I3C1_CLK	I3C1 clock control set 0b - Disable Clock 1b - Enable Clock
16 I3C0_CLK	I3C0 clock control set 0b - Disable Clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable Clock
15-11 —	Reserved
10 WWDT1_CLK	Watchdog Timer 1 clock control set 0b - Disable Clock 1b - Enable Clock
9 —	Reserved
8 MRT0_CLK	Multi-Rate Timer 0 clock control set 0b - Disable Clock 1b - Enable Clock
7 RTCLITE_CLK	RTC clock control set 0b - Disable Clock 1b - Enable Clock
6-5 —	Reserved
4 CT32BIT4_CLK	CT32BIT bit timer 4 clock set Sets the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
3 CT32BIT3_CLK	CT32BIT bit timer 3 clock set Sets the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
2 CT32BIT2_CLK	CT32BIT bit timer 2 clock set Sets the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
1 CT32BIT1_CLK	CT32BIT bit timer 1 clock set Sets the PSCCTL2 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Effect 1b - Set Bit
0 CT32BIT0_CLK	CT32BIT bit timer 0 clock set Sets the PSCCTL2 Bit 0b - No Effect 1b - Set Bit

6.5.2.8 Clock Clear 0 (PSCCTL0_CLR)

Offset

Register	Offset
PSCCTL0_CLR	70h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		FlexIO	Reserved	OSEV ENT...	Reserved	FC16_ SP...	DMIC0	FC15_ I2...	FC14_ SP...	FC13_ CLK	FC12_ CLK	FC11_ CLK	FC10_ CLK	FC9_ CLK	FC8_ CLK
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FC7_ CLK	FC6_ CLK	FC5_ CLK	FC4_ CLK	FC3_ CLK	FC2_ CLK	FC1_ CLK	FC0_ CLK	Reserved							
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-30 —	Reserved
29 FlexIO	FlexIO clock control clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Clears the PSCCTL0 bit
28 —	Reserved
27 OSEVENT_TIMER	OS event timer bus clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
26 —	Reserved
25 FC16_SPI_CLK	Flexcomm Interface 16 SPI clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
24 DMIC0	DMIC0 clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
23 FC15_I2C_CLK	Flexcomm Interface 15 I2C clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
22 FC14_SPI_CLK	Flexcomm Interface 14 SPI clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
21-8 FCn_CLK	Flexcomm Interface n clock control clear 0b - No effect 1b - Clears the PSCCTL0 bit
7-0 —	Reserved

6.5.2.9 Clock Clear 1 (PSCCTL1_CLR)

Offset

Register	Offset
PSCCTL1_CLR	74h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FREQ	Reserv	SEMA	MU_	Reserved				DMAC	DMAC	Reserved				CRC_	
W	ME_...		_CLK	CLK					1_C...	0_C...					CLK	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI	HSGPI
W									O7...	O6...	O5...	O4...	O3...	O2...	O1...	O0...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 FREQME_CLK	Frequency Measurement clock control 0b - No effect 1b - Clears the PSCCTL1 bit
30 —	Reserved
29 SEMA_CLK	Semaphore clock control 0b - No effect 1b - Clears the PSCCTL1 bit
28 MU_CLK	Messaging Unit clock control 0b - No effect 1b - Clears the PSCCTL1 bit
27-25 —	Reserved
24 DMAC1_CLK	DMAC1 clock control 0b - No effect 1b - Clears the PSCCTL1 bit
23 DMAC0_CLK	DMAC0 clock control 0b - No effect 1b - Clears the PSCCTL1 bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
22-17 —	Reserved
16 CRC_CLK	CRC clock control 0b - No effect 1b - Clears the PSCCTL1 bit
15-8 —	Reserved
7-0 HSGPIOn_CLK	Non-secure GPIOn clock control 0b - No effect 1b - Clears the PSCCTL1 bit

6.5.2.10 Clock Clear 2 (PSCCTL2_CLR)

Offset

Register	Offset
PSCCTL2_CLR	78h

Function

Writing a 1 to a bit position in this register clears the corresponding position in PSCCTL2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIMCT	GPIOI	Reserved												I3C1_	I3C0_
W	L_...	NT...													CLK	CLK
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				WWDT		Reserv	MRT0_	RTCLI	Reserved						
W					1_C...			CLK	TE...			CT32B	CT32B	CT32B	CT32B	CT32B
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 PIMCTL_CLK	INPUTMUX clock control clear 0b - Disable Clock 1b - Enable Clock
30 GPIOINTCTL_CLK	PINT clock control clear 0b - Disable Clock 1b - Enable Clock
29-18 —	Reserved
17 I3C1_CLK	I3C1 clock control clear 0b - Disable Clock 1b - Enable Clock
16 I3C0_CLK	I3C0 clock control clear 0b - Disable Clock 1b - Enable Clock
15-11 —	Reserved
10 WWDT1_CLK	Watchdog Timer 1 clock control clear 0b - Disable Clock 1b - Enable Clock
9 —	Reserved
8 MRT0_CLK	Multi-Rate Timer 0 clock control clear 0b - Disable Clock 1b - Enable Clock
7 RTCLITE_CLK	RTC clock control clear 0b - Disable Clock 1b - Enable Clock
6-5 —	Reserved
4 CT32BIT4_CLK	CT32BIT bit timer 4 clock clear Clears the PSCCTL2 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Effect 1b - Set Bit
3 CT32BIT3_CLK	CT32BIT bit timer 3 clock clear Clears the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
2 CT32BIT2_CLK	CT32BIT bit timer 2 clock clear Clears the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
1 CT32BIT1_CLK	CT32BIT bit timer 1 clock clear Clears the PSCCTL2 Bit 0b - No Effect 1b - Set Bit
0 CT32BIT0_CLK	CT32BIT bit timer 0 clock clear Clears the PSCCTL2 Bit 0b - No Effect 1b - Set Bit

6.5.2.11 Audio PLL0 Clock Select (AUDIOPLL0CLKSEL)

Offset

Register	Offset
AUDIOPLL0CLKSEL	200h

Function

This register selects the clock source for the audio PLL.

The AUDIOPLL0 output frequency is controlled by the programmed values of MULT, NUM and DENOM and can be expressed as a function of the AUDIOPLL0 reference input frequency. Use this AUDIOPLL0CLKSEL register to select the AUDIOPLL0 reference input frequency.

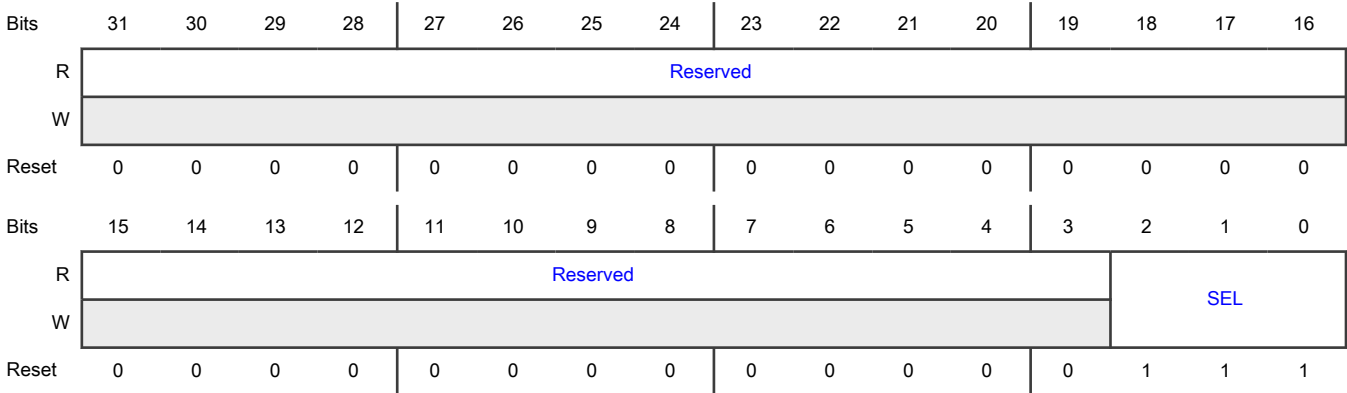
$$F_{\text{SYSPLL0}} = F_{\text{PLLREFIN}} * (\text{MULT} + \text{NUM}/\text{DENOM})$$

After an input frequency has been chosen, MULT must be chosen so that the AUDIOPLL0 output frequency is 528 MHz. Therefore, if choosing a AUDIOPLL0 reference input frequency of 24 MHz, then program MULT as (0x16) or decimal 22, because 24 MHz * 22 = 528 MHz. Fine-tuning of the frequency can be done using NUM and DENOM.

NOTE

The clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change. The selected clock coming out of reset is OFF and to turn it ON enable FRO_DIV8.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Audio PLL0 Clock Select Audio PLL0 Reference Input Clock Source NOTE The PLL Reference Input Frequency is limited to the following discrete frequencies: 32, 30, 26, 24, 19.2, 16 MHz. 000b - FRO_DIV8 001b - OSC_CLK clock (User-Selectable) 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.12 Audio PLL0 Control 0 (AUDIOPLL0CTL0)

Offset

Register	Offset
AUDIOPLL0CTL0	204h

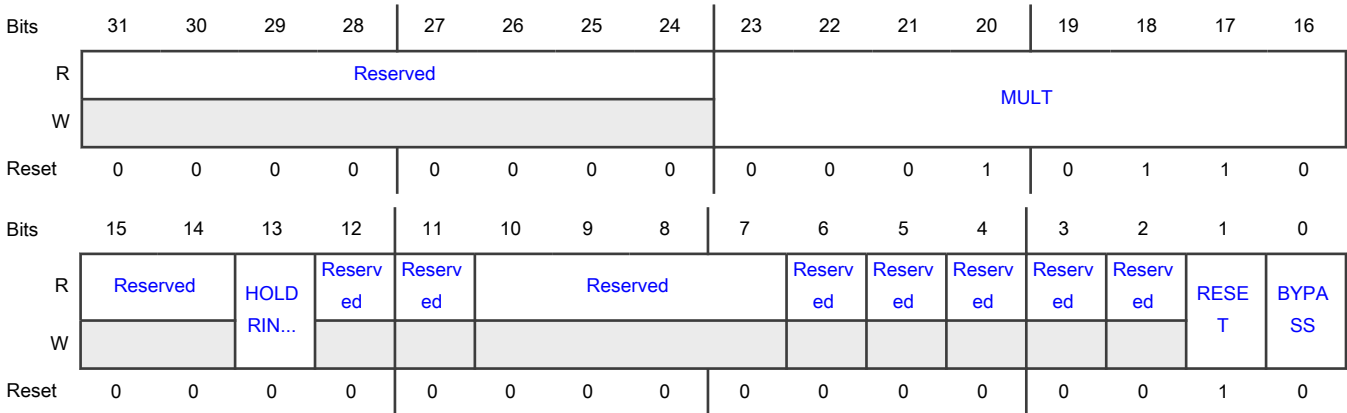
Function

The AUDIOPLL0CTL0 register contains control options for the AUDIO PLL0.

NOTE

AUDIOPLL0CTL0[MULT] should be greater than or equal to 20 when AUDIOPLL0NUM[NUM] is not 0.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 MULT	Multiplication Factor Multiplication Factor for F_{SYSPLL0_OUTPUT}. The value of MULT must always be one of the 6 options here. The value of MULT is directly decoded. NOTE AUDIOPLL0CTL0[MULT] should be greater than or equal to 20 when AUDIOPLL0NUM[NUM] is not 0. 0001_0000b - Multiply by 16 0001_0001b - Multiply by 17 0001_0010b - Multiply by 18

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0001_0011b - Multiply by 19 0001_0100b - Multiply by 20 0001_0101b - Multiply by 21 0001_0110b - Multiply by 22
15-14 —	Reserved
13 HOLDRINGOFF F_ENA	Hold Ring Off Control Use HOLDRINGOFF_ENA to avoid multi-wave problems within the VCO. NOTE Assert HOLDRINGOFF_ENA bit in the 1st half of lock time; de-assert HOLDRINGOFF_ENA bit in the 2nd half of lock time. 0b - Disable 1b - Enable
12 —	Reserved
11 —	Reserved
10-7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 —	Reserved
1	AUDIOPLL0 Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
RESET	0b - AUDIOPLL0 reset is removed 1b - AUDIOPLL0 is placed into reset
0 BYPASS	AUDIOPLL0 BYPASS Mode 0b - PFD outputs are PFD-programmed clocks 1b - Bypass Mode. PFD outputs are sourced directly from the reference input clock

6.5.2.13 Audio PLL0 Lock Time Divide-by-2 (AUDIOPLL0LOCKTIMEDIV2)

Offset

Register	Offset
AUDIOPLL0LOCKTIME DIV2	20Ch

Function

On any reset, the BOOTROM will copy the AUDIOPLL0 worst case characterized lock time value (found in the OTP fuses) into the respective fields (if the OTP valid signature is present). Software needs this value to control the HOLDRINGOFF bit in the AUDIOPLL0CTL0 register, to know when half the lock time has expired and when the full lock has passed. Any PLL lock sequence requires the HOLDRINGOFF to be enabled ('1') during the first half of the lock time, and the HOLDRINGOFF to be disabled ('0') during the second half of the lock time.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCKTIMEDIV2															
W																
Reset	1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	0

Fields

Field	Function
31-16	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-0 LOCKTIMEDIV 2	AUDIOPLL0 Lock Time Divide-by-2 The programmed lock time is in uS (microseconds) and is programmed as 1/2 the actual lock time value.

6.5.2.14 Audio PLL0 Numerator (AUDIOPLL0NUM)

Offset

Register	Offset
AUDIOPLL0NUM	210h

Function

This register provides the numerator for the audio PLL fractional divider. If the PLL numerator and denominator are set up and subsequently altered while the PLL is running, the amount of change on a single step of the NUM should be limited to 100 ppm of the current setting.

Diagram

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

Fields

Field	Function
31-30 —	Reserved
29-0 NUM	Numerator The numerator of the AUDIOPLL0 fractional loop divider. - The value of numerator must always be configured to be less than the value of the denominator. - The AUDIOPLL0NUM register can only be changed when the AUDIOPLL0 is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE AUDIOPLL0CTL0[MULT] should be greater than or equal to 20 when AUDIOPLL0NUM[NUM] is not 0.

6.5.2.15 Audio PLL0 Denominator (AUDIOPLL0DENOM)

Offset

Register	Offset
AUDIOPLL0DENOM	214h

Function

This register provides the denominator for the audio PLL fractional divider.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-0 DENOM	Denominator The denominator of the AUDIOPLL0 fractional loop divider. - The value of denominator must always be configured to be greater than the value of the numerator. - The AUDIOPLL0DENOM register can only be changed when the AUDIOPLL0 is disabled.

6.5.2.16 Audio PLL0 PFD (AUDIOPLL0PFD)

Offset

Register	Offset
AUDIOPLL0PFD	218h

Function

The AUDIOPLL0PFD register defines the control bits for the PFD0 - PFD3 clocks derived from the AUDIOPLL0.

The PFD output frequencies are governed by the following expression:

$$F_{PFDx} = F_{SYSPLL0\_OUT} * (18 / PFDx)$$

When changing PFD values, follow this recommended sequence:

- Ensure that the PFDx clock gets gated by writing a value of '1' to the PFDx_CLKGATE bit,
- Then program the new PFD value (values are limited to decimal values 12 - 35).
- Write a value of '0' to the PFDx_CLKGATE register to un-gate the PFDx clock,
- Next poll the PFDx_CLKRDY status flag until it (the status flag) is set to '1'.
- Clear the PFDx_CLKRDY status flag, by writing a '1' to that specific PFDx_CLKRDY bit.
- The PFD outputs can be no higher than 600MHz.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PFD3_	PFD3_							PFD2_	PFD2_						
W	CL...	CL...							CL...	CL...						
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PFD1_	PFD1_							PFD0_	PFD0_						
W	CL...	CL...							CL...	CL...						
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Fields

Field	Function
31 PFD3_CLKGATE	PFD3 Clock Gate 0b - PFD3 clock is not gated 1b - PFD3 clock is gated
30 PFD3_CLKRDY	PFD3 Clock Ready Status Flag Cleared by writing a '1'.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not ready 1b - Ready
29-24 PFD3	PLL Fractional Divider 3 The fractional divider value. Valid PFD values are decimal 12 - 35.
23 PFD2_CLKGAT E	PFD2 Clock Gate 0b - PFD2 clock is not gated 1b - PFD2 clock is gated
22 PFD2_CLKRDY	PFD2 Clock Ready Status Flag Cleared by writing a '1'. 0b - Not ready 1b - Ready
21-16 PFD2	PLL Fractional Divider 2 The fractional divider value. Valid PFD values are decimal 12 - 35.
15 PFD1_CLKGAT E	PFD1 Clock Gate 0b - PFD1 clock is not gated 1b - PFD1 clock is gated
14 PFD1_CLKRDY	PFD1 Clock Ready Status Flag Cleared by writing a '1'. 0b - Not ready 1b - Ready
13-8 PFD1	PLL Fractional Divider 1 The fractional divider value. Valid PFD values are decimal 12 - 35.
7 PFD0_CLKGAT E	PFD0 Clock Gate 0b - PFD0 clock is not gated 1b - PFD0 clock is gated
6 PFD0_CLKRDY	PFD0 Clock Ready Status Flag Cleared by writing a '1'. 0b - Not ready 1b - Ready
5-0 PFD0	PLL Fractional Divider 0 The fractional divider value. Valid PFD values are decimal 12 - 35.

6.5.2.17 Audio PLL Clock Divider (AUDIOPLLCLKDIV)

Offset

Register	Offset
AUDIOPLLCLKDIV	240h

Function

Contains controls for the Audio PLL Clock Divider.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESET	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished 1b - The Divider value has changed
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Audio PLL Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.18 DSP CPU Clock Divider (DSPCPUCLKDIV)

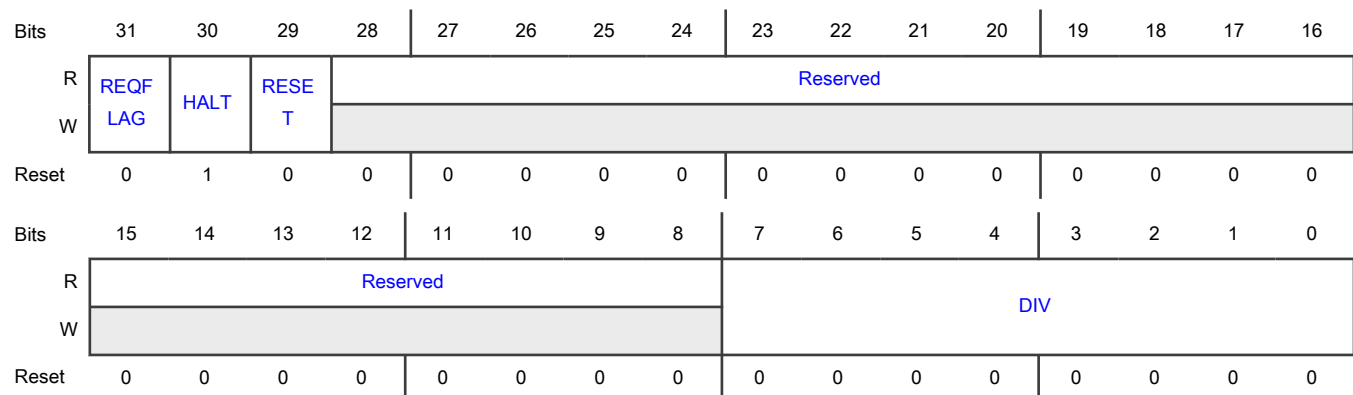
Offset

Register	Offset
DSPCPUCLKDIV	400h

Function

All bits are reset by all reset sources.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	DSP Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • . . . • 255: Divide by 256

6.5.2.19 DSP CPU Clock Select A (DSPCPUCLKSELA)**Offset**

Register	Offset
DSPCPUCLKSELA	430h

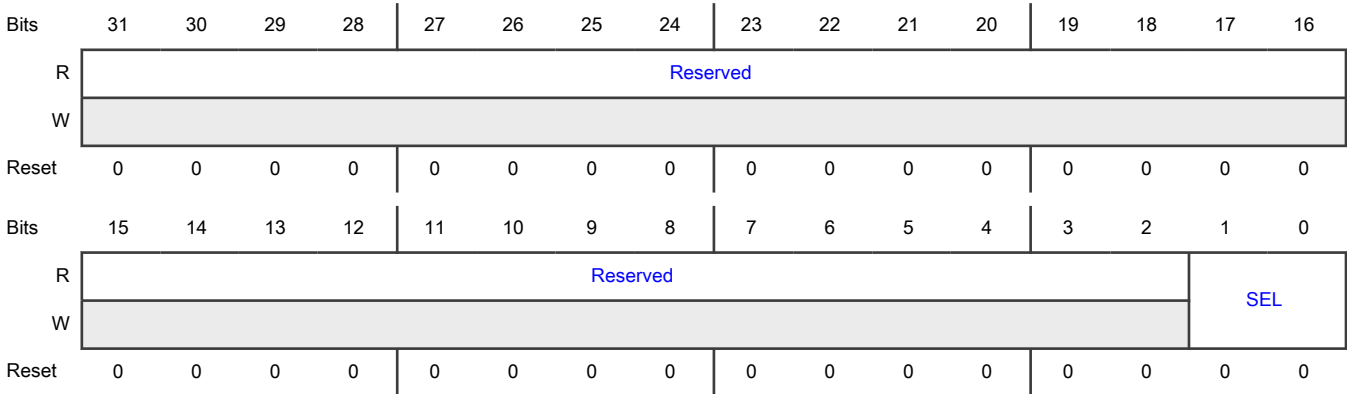
Function

Selects the DSP Main 1st Stage Control clock.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change. The default clock for this mux, “FRO Clock”, is OFF coming out of reset. Prior to changing this mux selection it is necessary to first enable the FRO in the PDRUNCFG0 register and set the FRO_DIV1_O_EN bit in the FRODIVOEN register.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 SEL	DSP Main 1st Stage Control Clock Source 00b - FRO_DIV1 Clock 01b - OSC_CLK Clock 10b - Low Power Oscillator Clock (LPOSC) 11b - Reserved

6.5.2.20 DSP CPU Clock Select B (DSPCPUCLKSELB)

Offset

Register	Offset
DSPCPUCLKSELB	434h

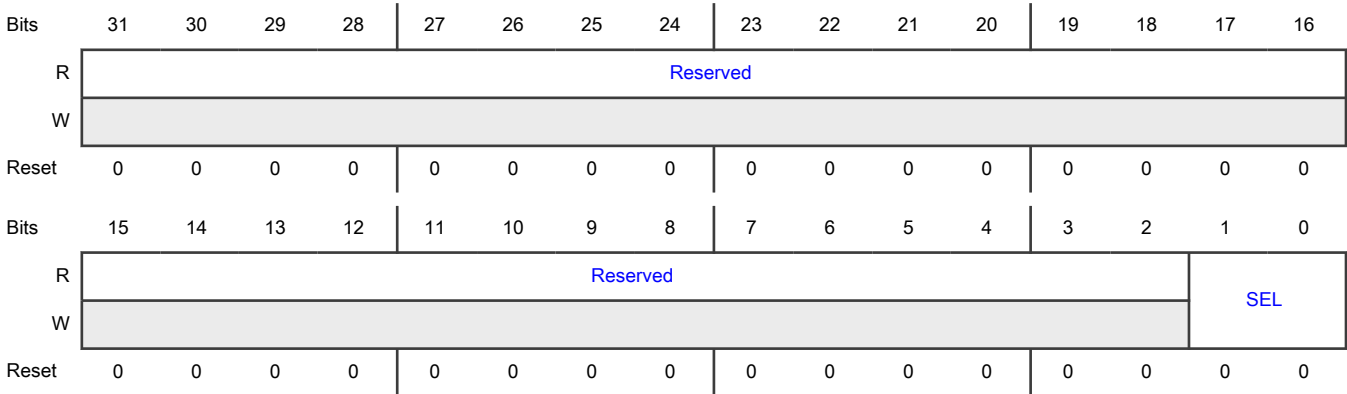
Function

Selects the DSP CPU Main clock.

NOTE

This clock selection is internally synchronized. The clock being switched from and the clock being switched to must both be running for the selection to change. The default clock for this mux, “FRO Clock”, is OFF coming out of reset. Prior to changing this mux selection it is necessary to first enable the FRO in the PDRUNCFG0 register and set the FRO_DIV1_O_EN bit in the FRODIVOEN register.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 SEL	Main Clock Source 00b - MAINCLKSELA 1st Stage Clock 01b - Main System PLL Clock 10b - DSP System PLL Clock 11b - RTC 32 KHz Clock

6.5.2.21 OS Event Timer Functional Clock Select (OSEVENTTFCLKSEL)

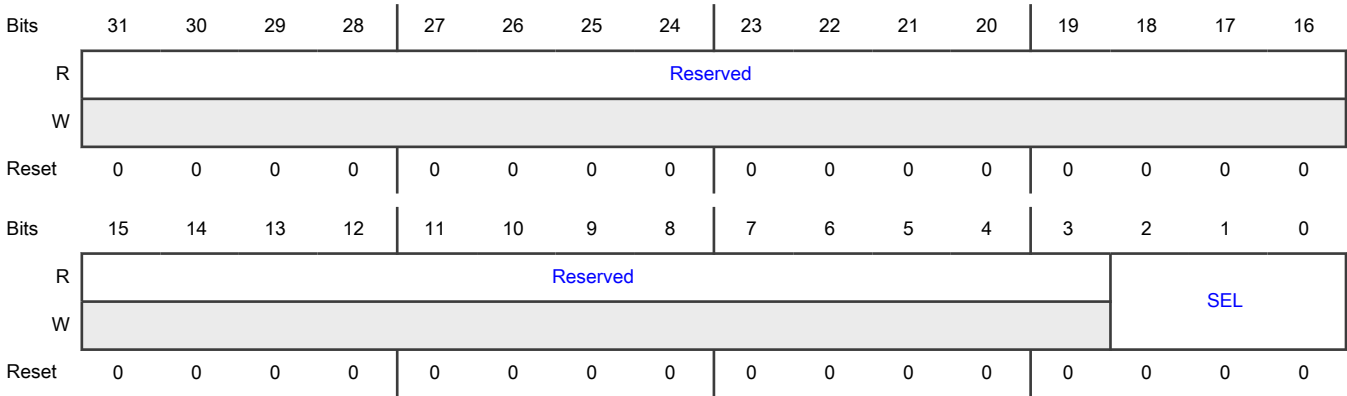
Offset

Register	Offset
OSEVENTTFCLKSEL	480h

Function

This register selects the clock source for the OS Event Timer.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	OS Event Timer Functional Clock Source 000b - Low Power Oscillator Clock (LPOSC) 001b - RTC 32 KHz Clock 010b - HCLK Free-Running Clock (Global Time Stamping) 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.22 Fractional Rate Generator 0 Clock Select (FRG0CLKSEL)

Offset

Register	Offset
FRG0CLKSEL	500h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm0.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 0 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.23 Fractional Rate Generator 0 Control (FRG0CTL)**Offset**

Register	Offset
FRG0CTL	504h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 0. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

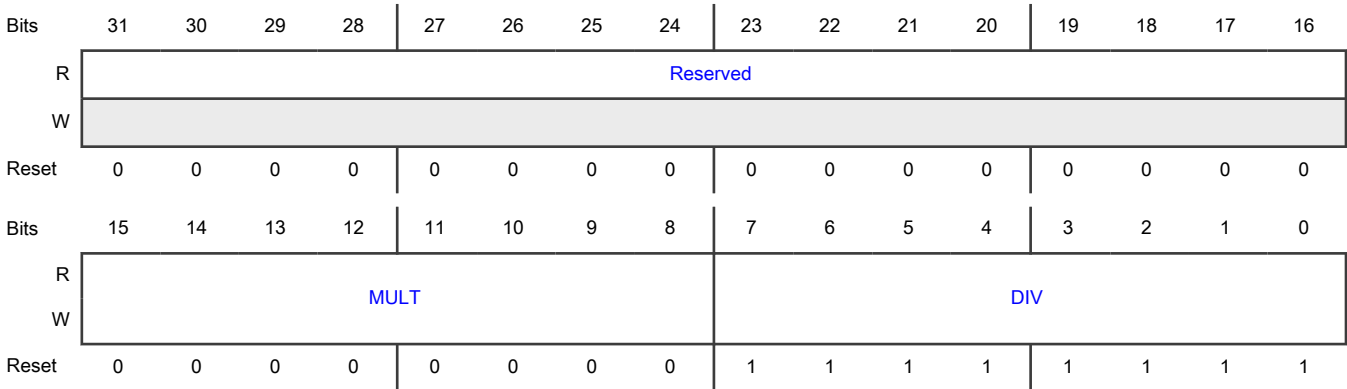
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.24 Flexcomm0 Clock Select (FC0FCLKSEL)

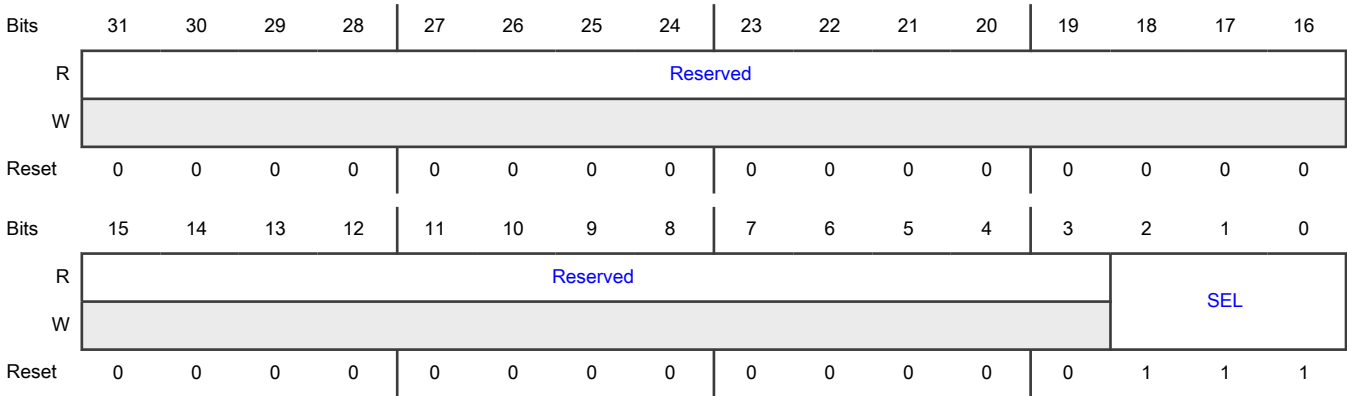
Offset

Register	Offset
FC0FCLKSEL	508h

Function

This register selects the clock source for Flexcomm0.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.25 Fractional Rate Generator 1 Clock Select (FRG1CLKSEL)

Offset

Register	Offset
FRG1CLKSEL	520h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm1.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 1 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.26 Fractional Rate Generator 1 Control (FRG1CTL)**Offset**

Register	Offset
FRG1CTL	524h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 1. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

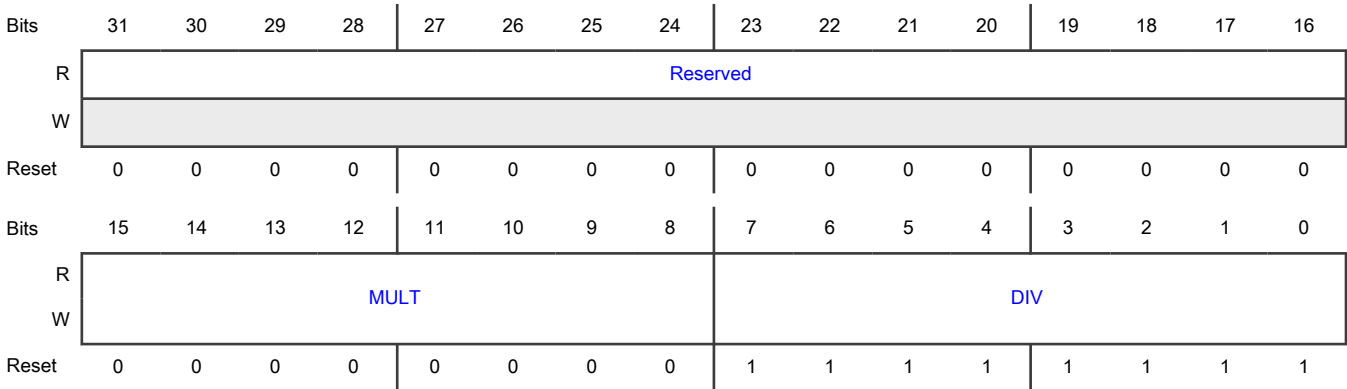
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.27 Flexcomm1 Clock Select (FC1FCLKSEL)

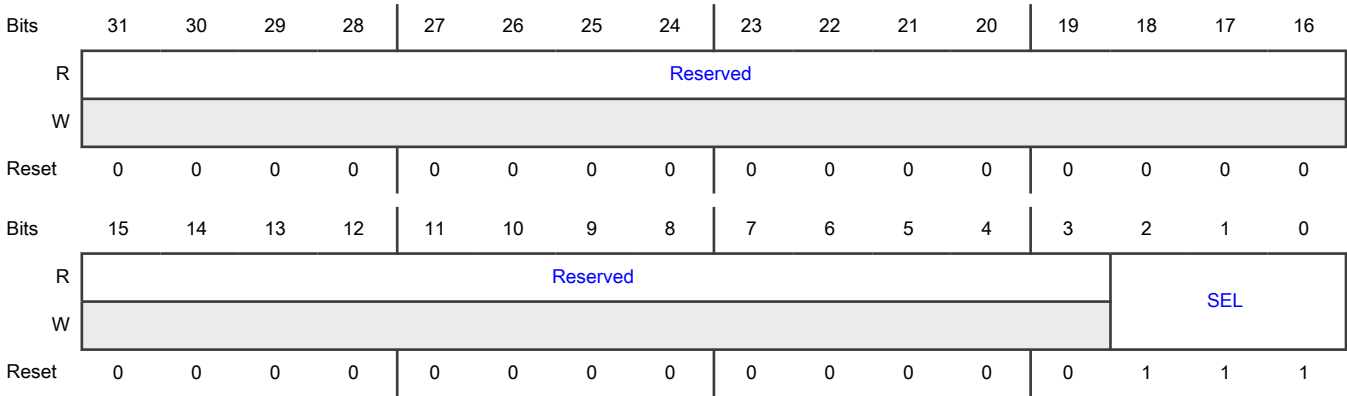
Offset

Register	Offset
FC1FCLKSEL	528h

Function

This register selects the clock source for Flexcomm1.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.28 Fractional Rate Generator 2 Clock Select (FRG2CLKSEL)

Offset

Register	Offset
FRG2CLKSEL	540h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm2.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 2 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.29 Fractional Rate Generator 2 Control (FRG2CTL)**Offset**

Register	Offset
FRG2CTL	544h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 2. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

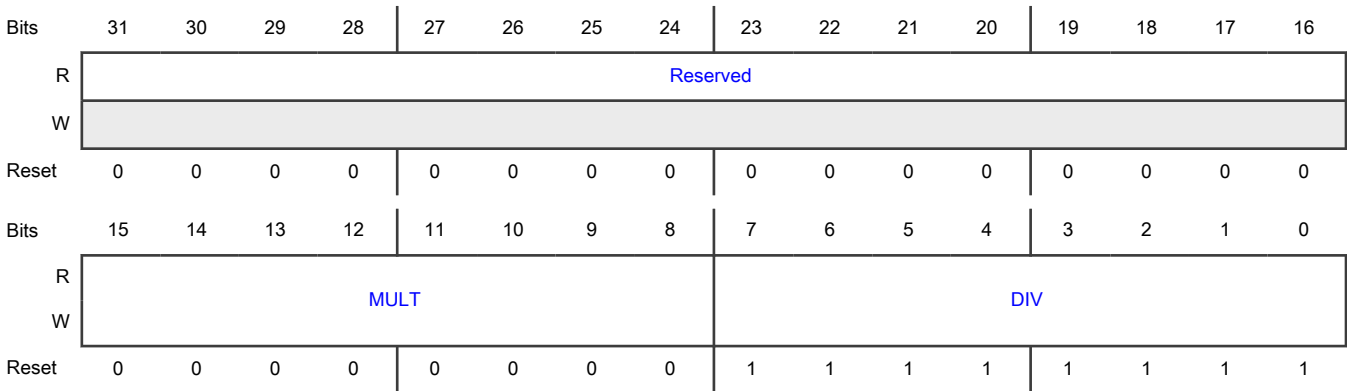
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.30 Flexcomm2 Clock Select (FC2FCLKSEL)

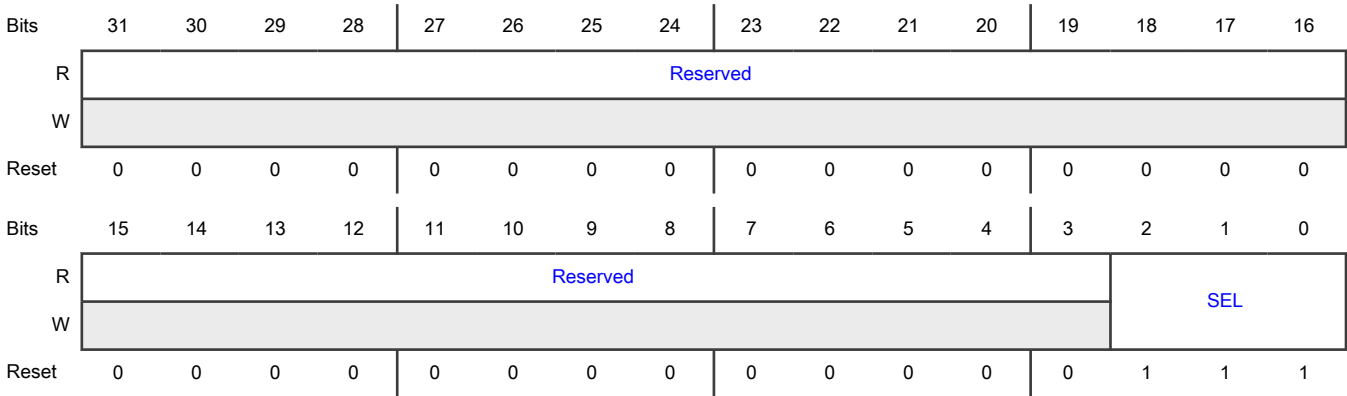
Offset

Register	Offset
FC2FCLKSEL	548h

Function

This register selects the clock source for Flexcomm2.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

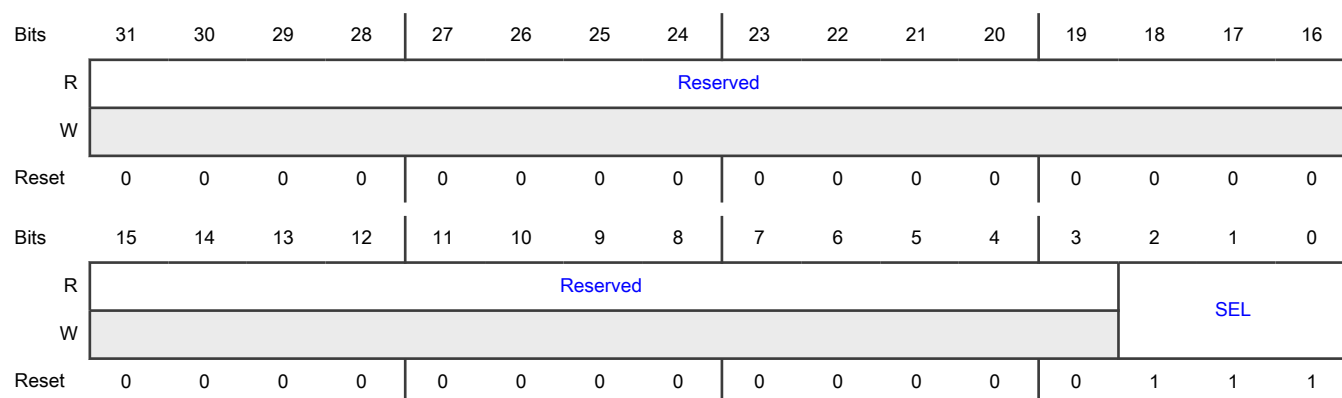
6.5.2.31 Fractional Rate Generator 3 Clock Select (FRG3CLKSEL)

Offset

Register	Offset
FRG3CLKSEL	560h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm3.

Diagram**Fields**

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 3 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.32 Fractional Rate Generator 3 Control (FRG3CTL)**Offset**

Register	Offset
FRG3CTL	564h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 3. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

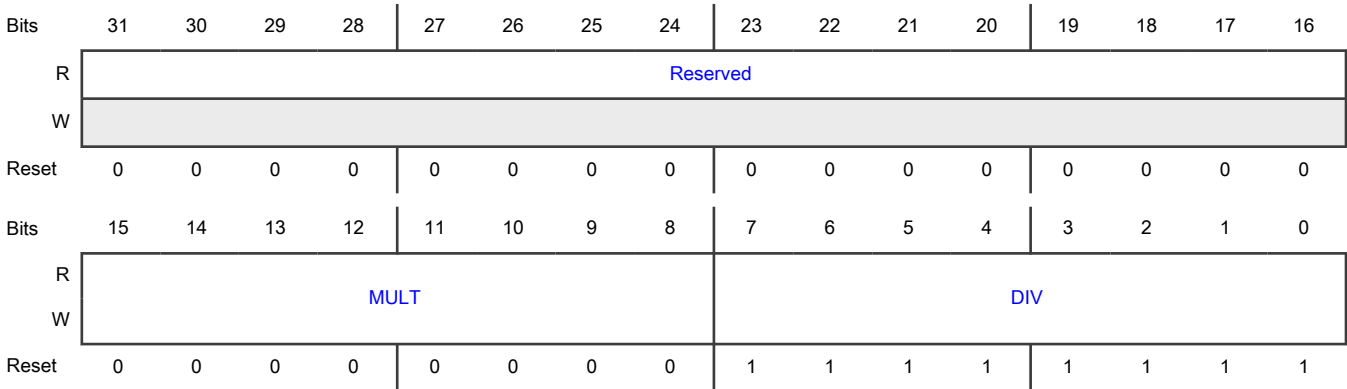
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.33 Flexcomm3 Clock Select (FC3FCLKSEL)

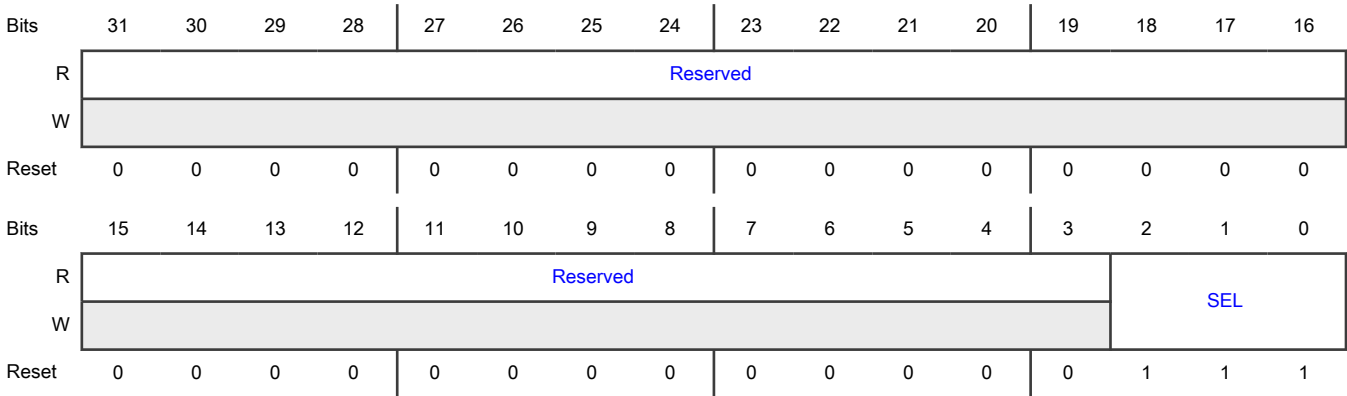
Offset

Register	Offset
FC3FCLKSEL	568h

Function

This register selects the clock source for Flexcomm3.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.34 Fractional Rate Generator 4 Clock Select (FRG4CLKSEL)

Offset

Register	Offset
FRG4CLKSEL	580h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm4.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 4 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.35 Fractional Rate Generator 4 Control (FRG4CTL)**Offset**

Register	Offset
FRG4CTL	584h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 4. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

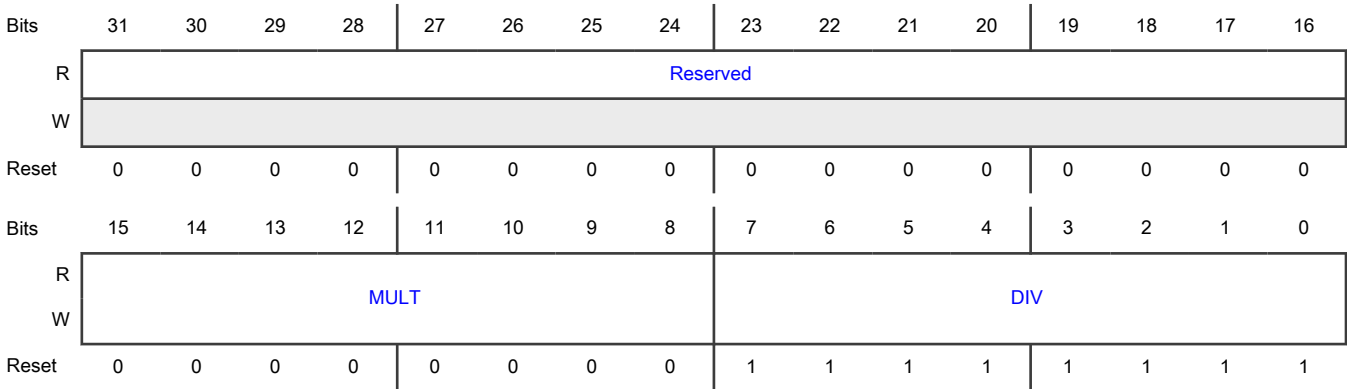
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.36 Flexcomm4 Clock Select (FC4FCLKSEL)

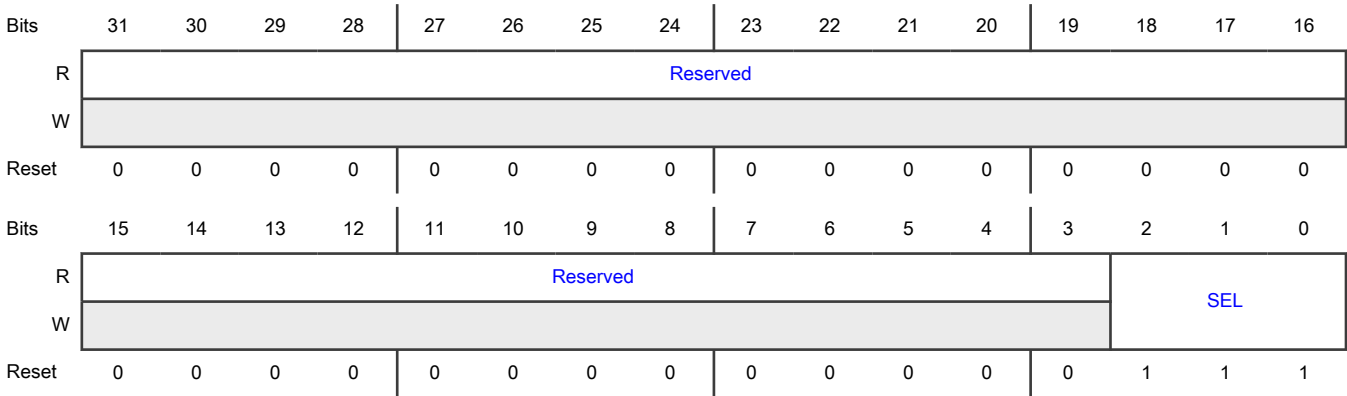
Offset

Register	Offset
FC4FCLKSEL	588h

Function

This register selects the clock source for Flexcomm4.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

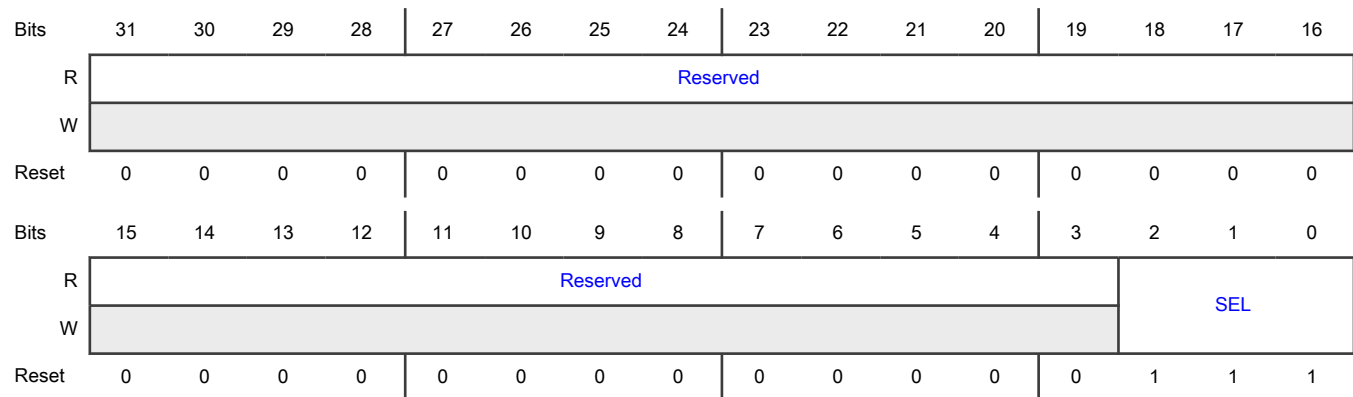
6.5.2.37 Fractional Rate Generator 5 Clock Select (FRG5CLKSEL)

Offset

Register	Offset
FRG5CLKSEL	5A0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm5.

Diagram**Fields**

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 5 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.38 Fractional Rate Generator 5 Control (FRG5CTL)**Offset**

Register	Offset
FRG5CTL	5A4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 5. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

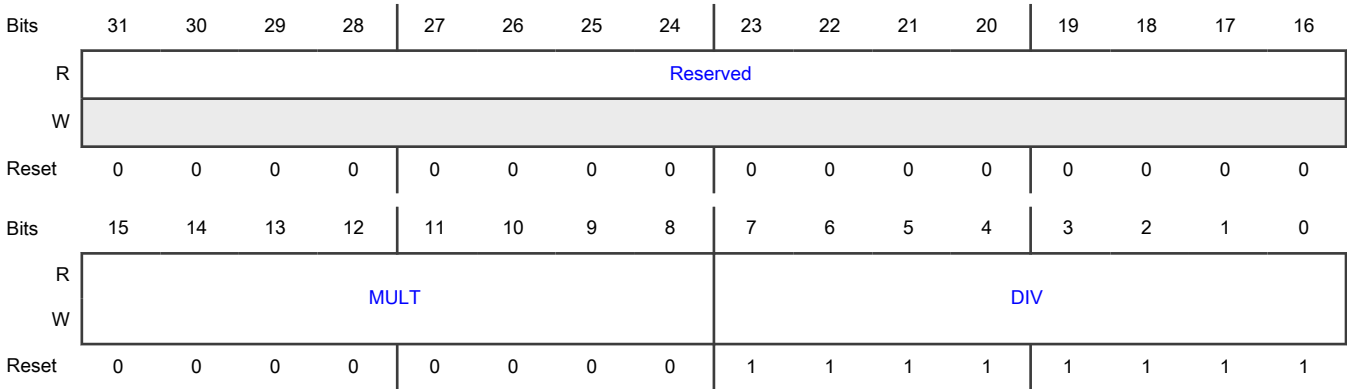
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.39 Flexcomm5 Clock Select (FC5FCLKSEL)

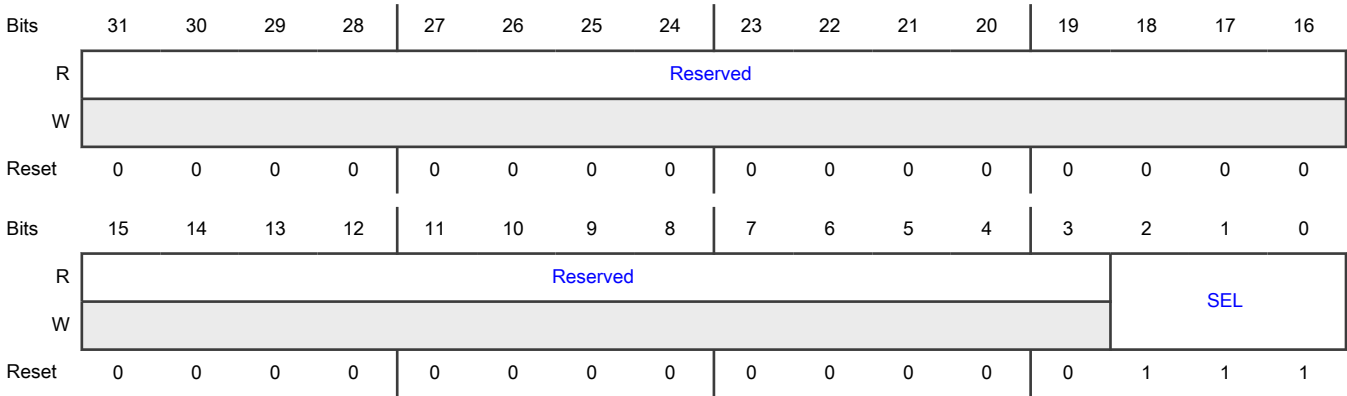
Offset

Register	Offset
FC5FCLKSEL	5A8h

Function

This register selects the clock source for Flexcomm5.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.40 Fractional Rate Generator 6 Clock Select (FRG6CLKSEL)

Offset

Register	Offset
FRG6CLKSEL	5C0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm6.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 6 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.41 Fractional Rate Generator 6 Control (FRG6CTL)**Offset**

Register	Offset
FRG6CTL	5C4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 6. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

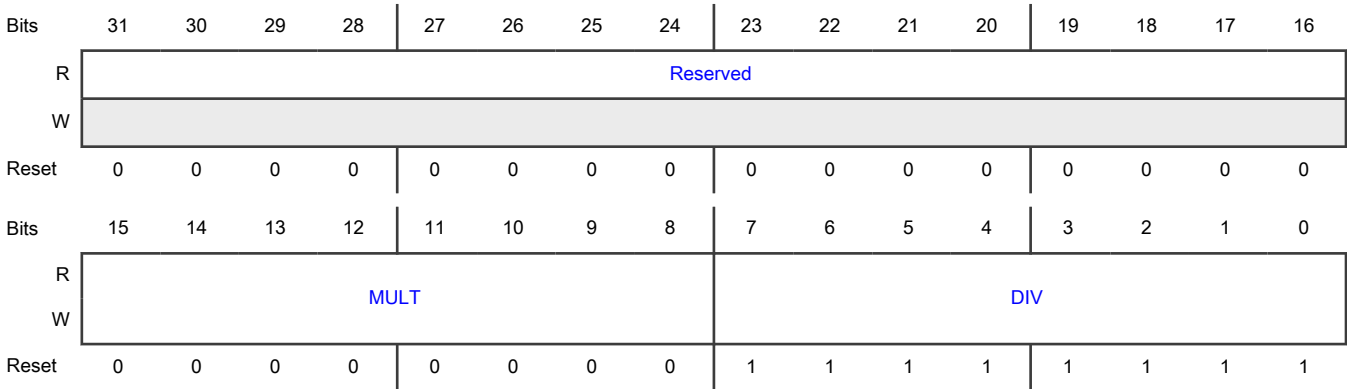
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.42 Flexcomm6 Clock Select (FC6FCLKSEL)

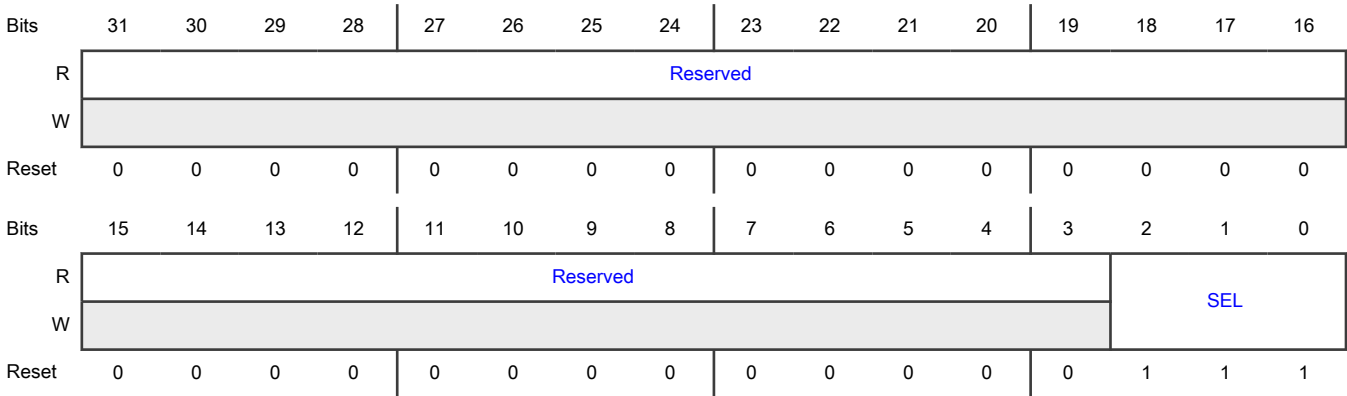
Offset

Register	Offset
FC6FCLKSEL	5C8h

Function

This register selects the clock source for Flexcomm6.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.43 Fractional Rate Generator 7 Clock Select (FRG7CLKSEL)

Offset

Register	Offset
FRG7CLKSEL	5E0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm7.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 7 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.44 Fractional Rate Generator 7 Control (FRG7CTL)**Offset**

Register	Offset
FRG7CTL	5E4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 7. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

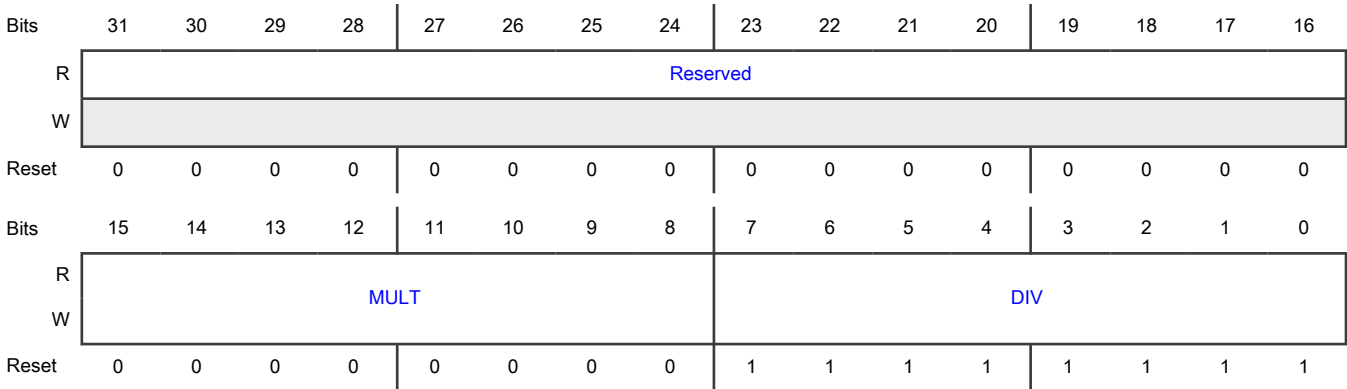
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.45 Flexcomm7 Clock Select (FC7FCLKSEL)

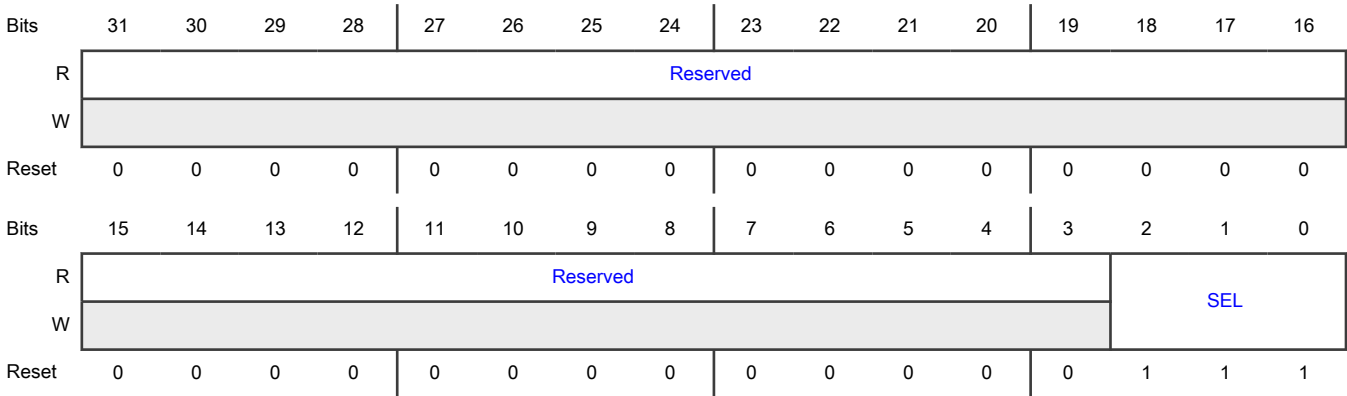
Offset

Register	Offset
FC7FCLKSEL	5E8h

Function

This register selects the clock source for Flexcomm7.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.46 Fractional Rate Generator 8 Clock Select (FRG8CLKSEL)

Offset

Register	Offset
FRG8CLKSEL	600h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm8.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 8 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.47 Fractional Rate Generator 8 Control (FRG8CTL)**Offset**

Register	Offset
FRG8CTL	604h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 8. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

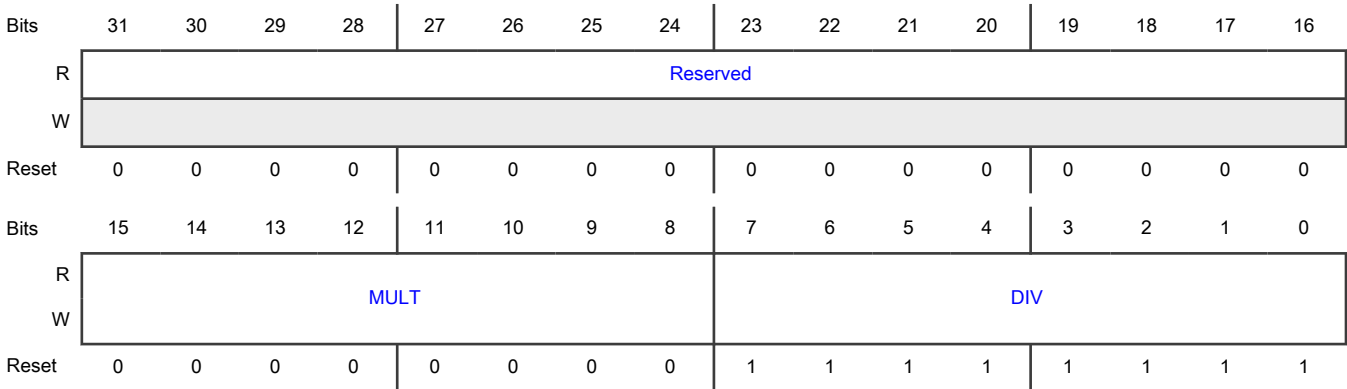
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.48 Flexcomm8 Clock Select (FC8FCLKSEL)

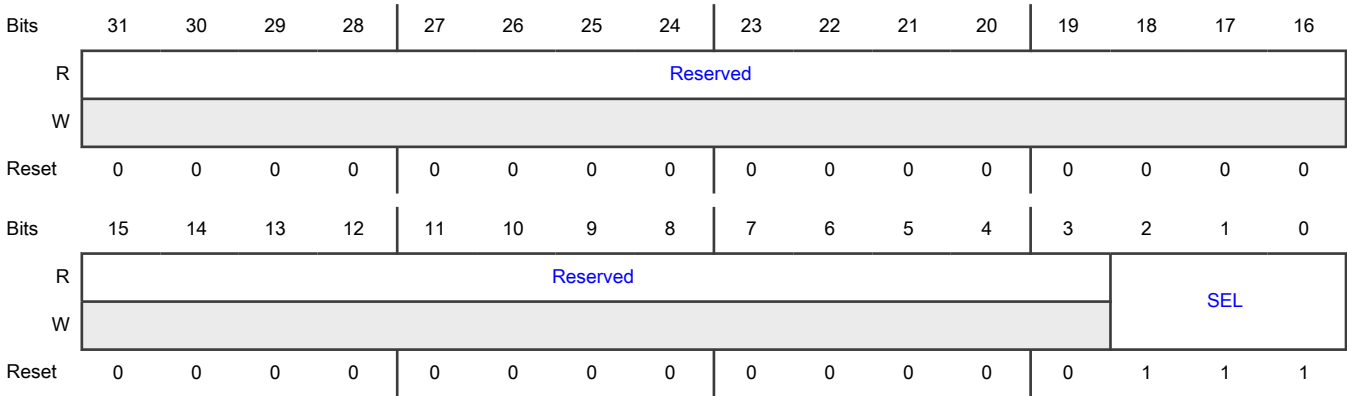
Offset

Register	Offset
FC8FCLKSEL	608h

Function

This register selects the clock source for Flexcomm8.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

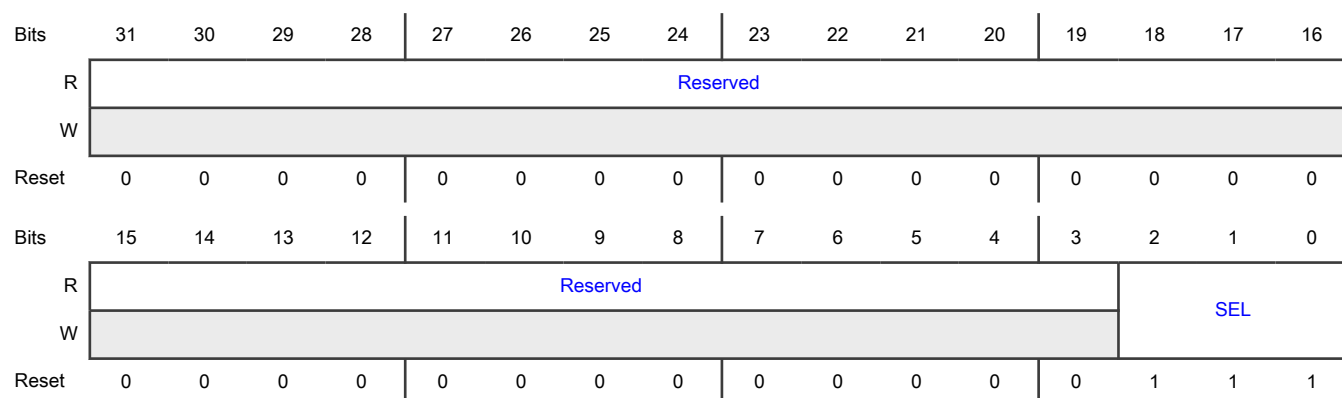
6.5.2.49 Fractional Rate Generator 9 Clock Select (FRG9CLKSEL)

Offset

Register	Offset
FRG9CLKSEL	620h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm9.

Diagram**Fields**

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 9 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.50 Fractional Rate Generator 9 Control (FRG9CTL)**Offset**

Register	Offset
FRG9CTL	624h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 9. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

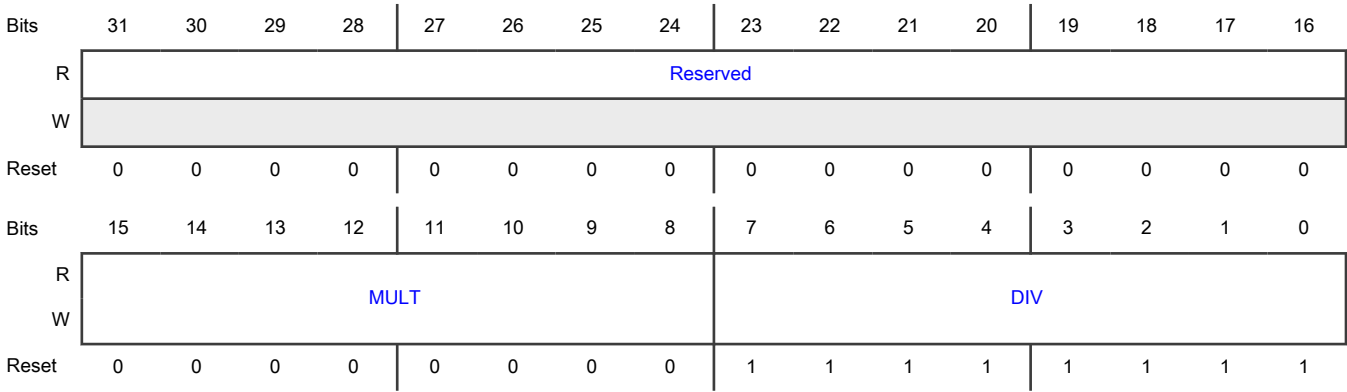
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.51 Flexcomm9 Clock Select (FC9FCLKSEL)

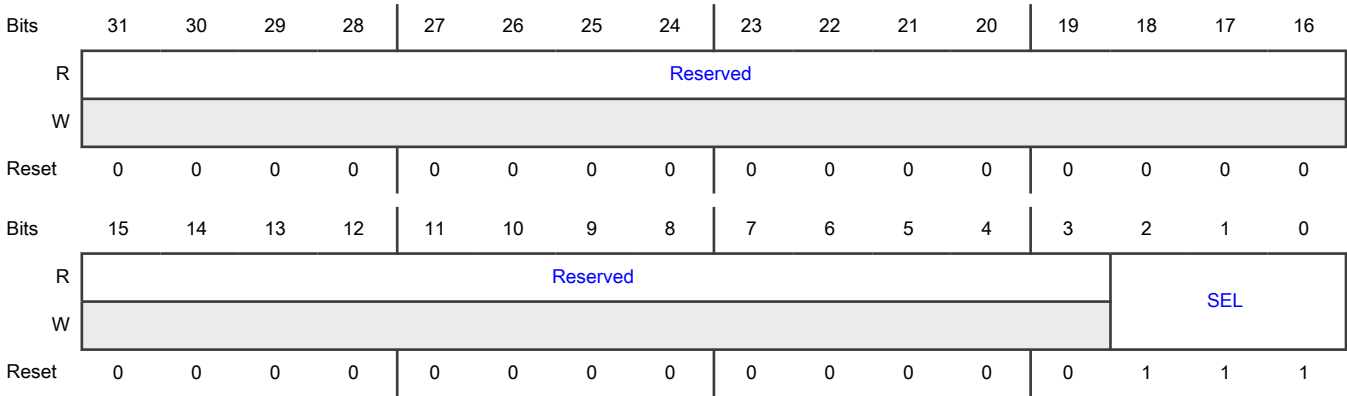
Offset

Register	Offset
FC9FCLKSEL	628h

Function

This register selects the clock source for Flexcomm9.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.52 Fractional Rate Generator 10 Clock Select (FRG10CLKSEL)

Offset

Register	Offset
FRG10CLKSEL	640h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm10.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 10 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.53 Fractional Rate Generator 10 Control (FRG10CTL)**Offset**

Register	Offset
FRG10CTL	644h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 10. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

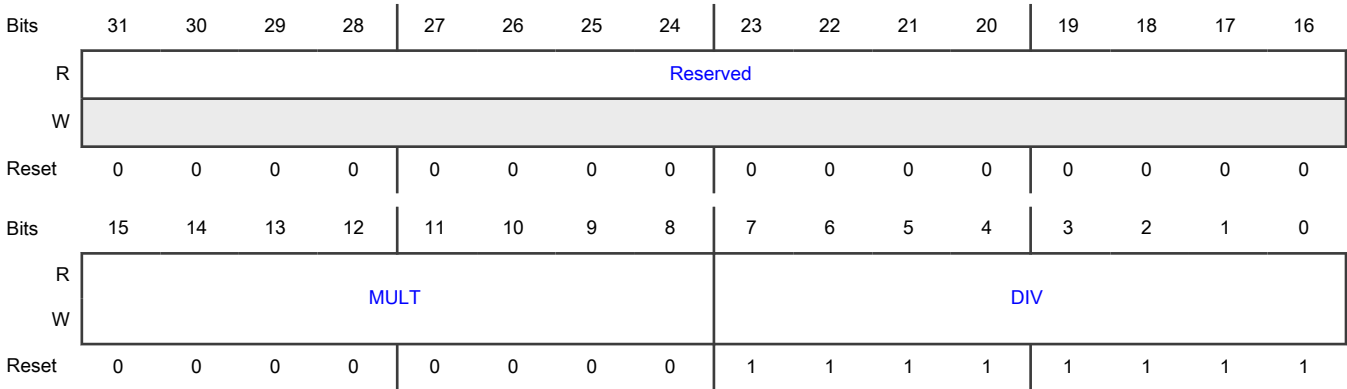
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.54 Flexcomm10 Clock Select (FC10FCLKSEL)

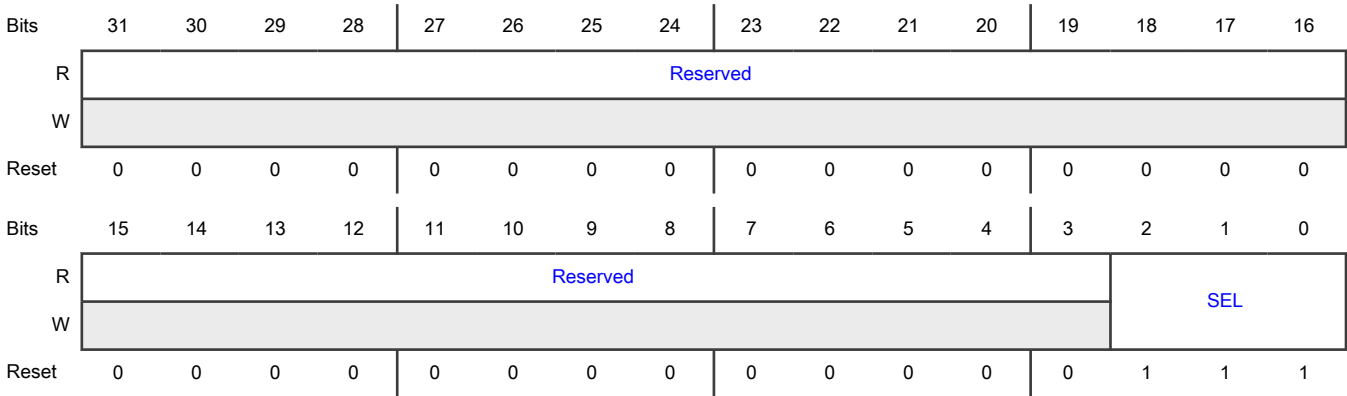
Offset

Register	Offset
FC10FCLKSEL	648h

Function

This register selects the clock source for Flexcomm10.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.55 Fractional Rate Generator 11 Clock Select (FRG11CLKSEL)

Offset

Register	Offset
FRG11CLKSEL	660h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm11.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 11 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.56 Fractional Rate Generator 11 Control (FRG11CTL)**Offset**

Register	Offset
FRG11CTL	664h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 11. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

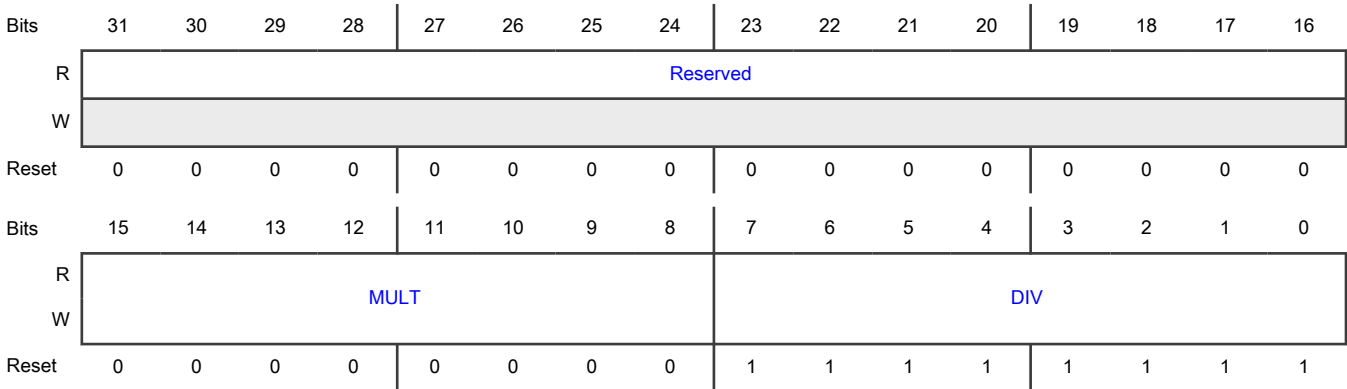
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.57 Flexcomm11 Clock Select (FC11FCLKSEL)

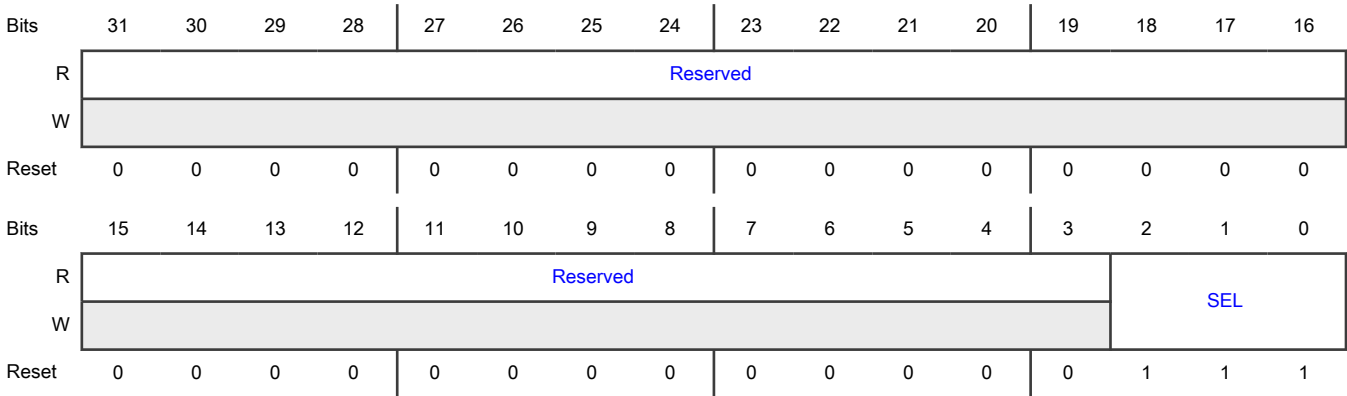
Offset

Register	Offset
FC11FCLKSEL	668h

Function

This register selects the clock source for Flexcomm11.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.58 Fractional Rate Generator 12 Clock Select (FRG12CLKSEL)

Offset

Register	Offset
FRG12CLKSEL	680h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm12.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 12 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.59 Fractional Rate Generator 12 Control (FRG12CTL)**Offset**

Register	Offset
FRG12CTL	684h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 12. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

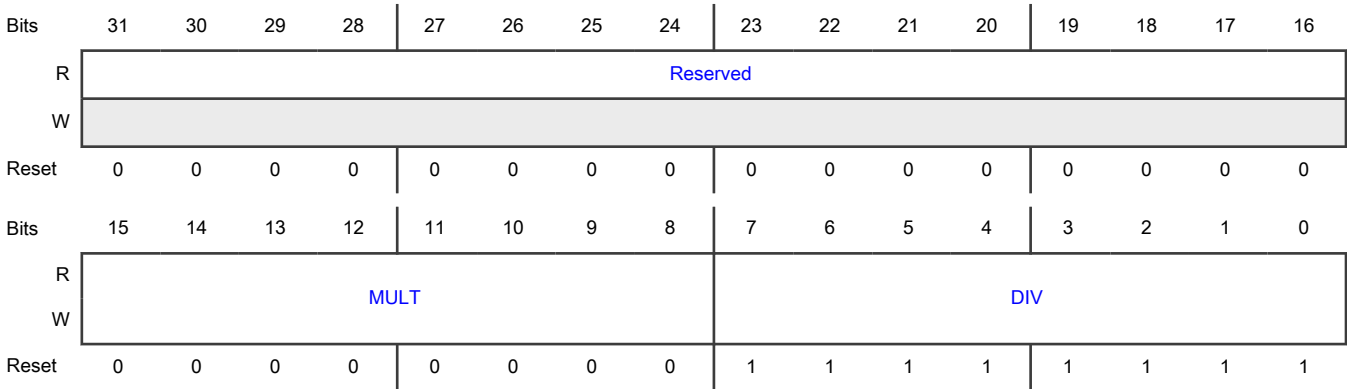
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.60 Flexcomm12 Clock Select (FC12FCLKSEL)

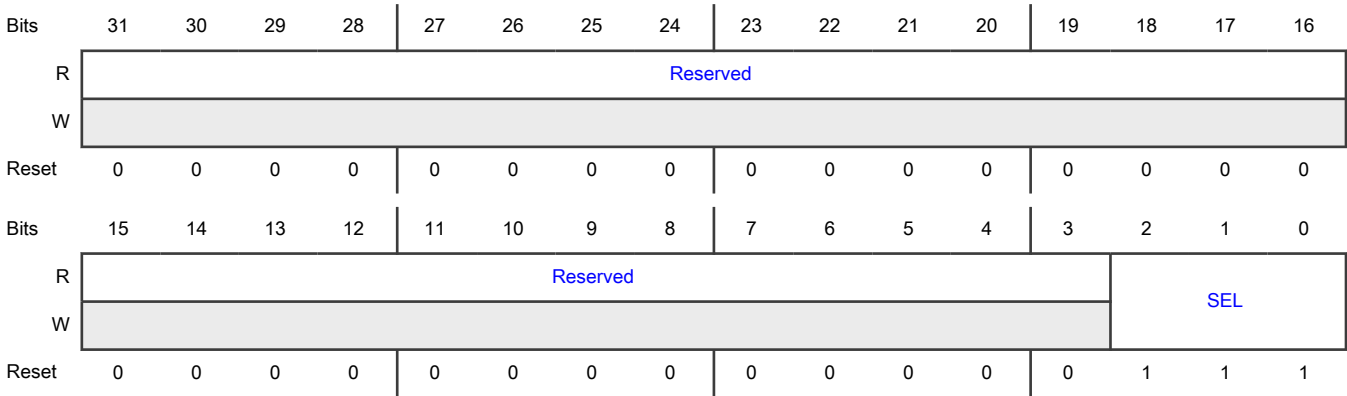
Offset

Register	Offset
FC12FCLKSEL	688h

Function

This register selects the clock source for Flexcomm12.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.61 Fractional Rate Generator 13 Clock Select (FRG13CLKSEL)

Offset

Register	Offset
FRG13CLKSEL	6A0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm13.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 13 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.62 Fractional Rate Generator 13 Control (FRG13CTL)**Offset**

Register	Offset
FRG13CTL	6A4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 13. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

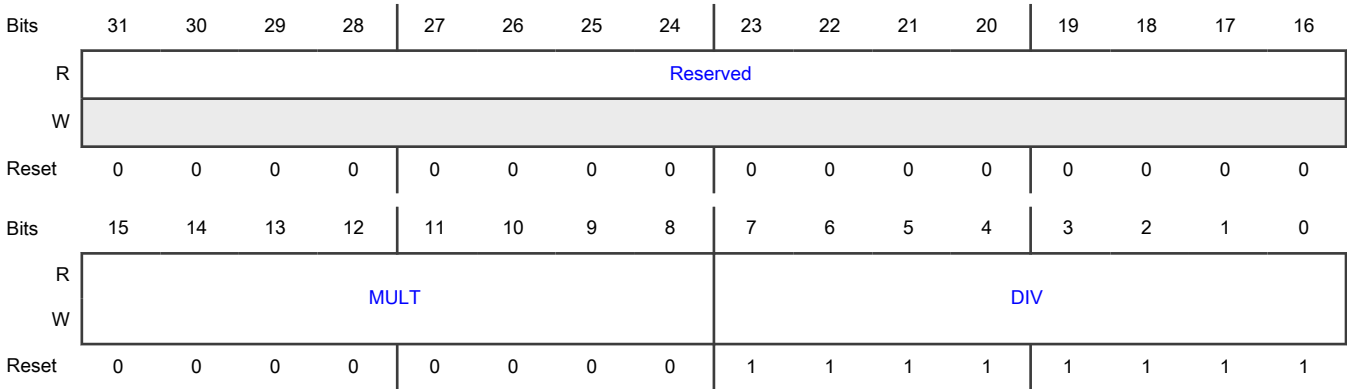
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.63 Flexcomm13 Clock Select (FC13FCLKSEL)

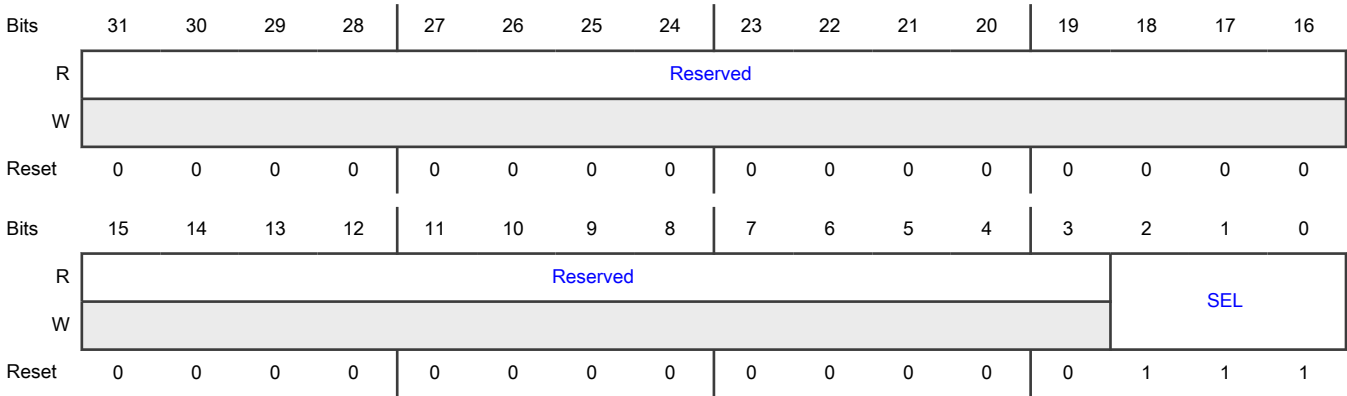
Offset

Register	Offset
FC13FCLKSEL	6A8h

Function

This register selects the clock source for Flexcomm13.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.64 Fractional Rate Generator 14 Clock Select (FRG14CLKSEL)

Offset

Register	Offset
FRG14CLKSEL	6C0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm14.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 14 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.65 Fractional Rate Generator 14 Control (FRG14CTL)**Offset**

Register	Offset
FRG14CTL	6C4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 14. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

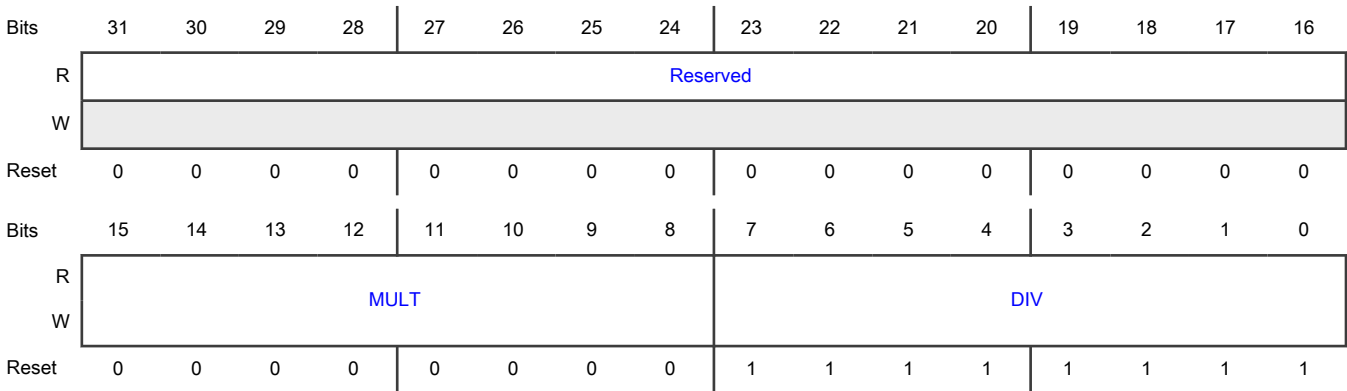
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.66 Flexcomm14 Clock Select (FC14FCLKSEL)

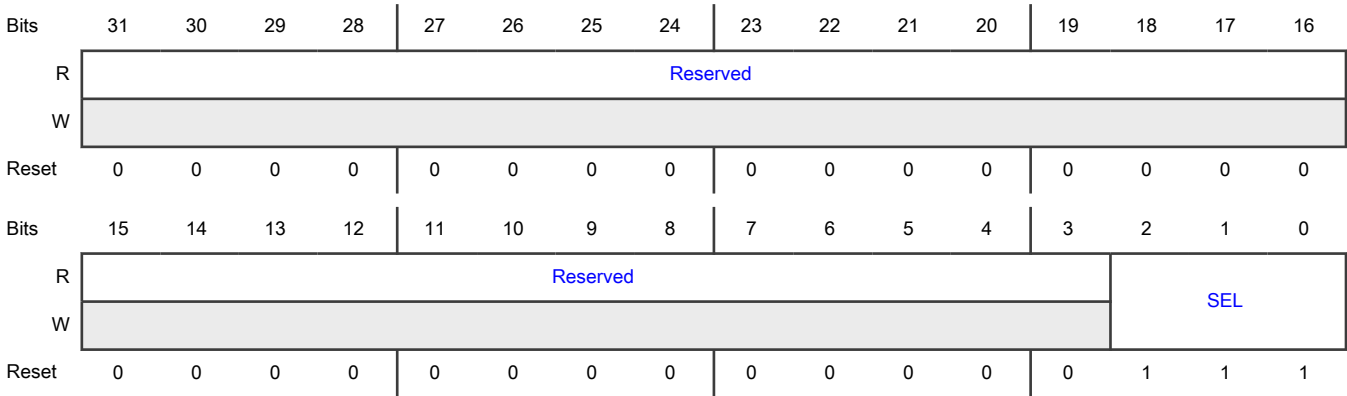
Offset

Register	Offset
FC14FCLKSEL	6C8h

Function

This register selects the clock source for Flexcomm14.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.67 Fractional Rate Generator 15 Clock Select (FRG15CLKSEL)

Offset

Register	Offset
FRG15CLKSEL	6E0h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm15.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 15 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.68 Fractional Rate Generator 15 Control (FRG15CTL)**Offset**

Register	Offset
FRG15CTL	6E4h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 15. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

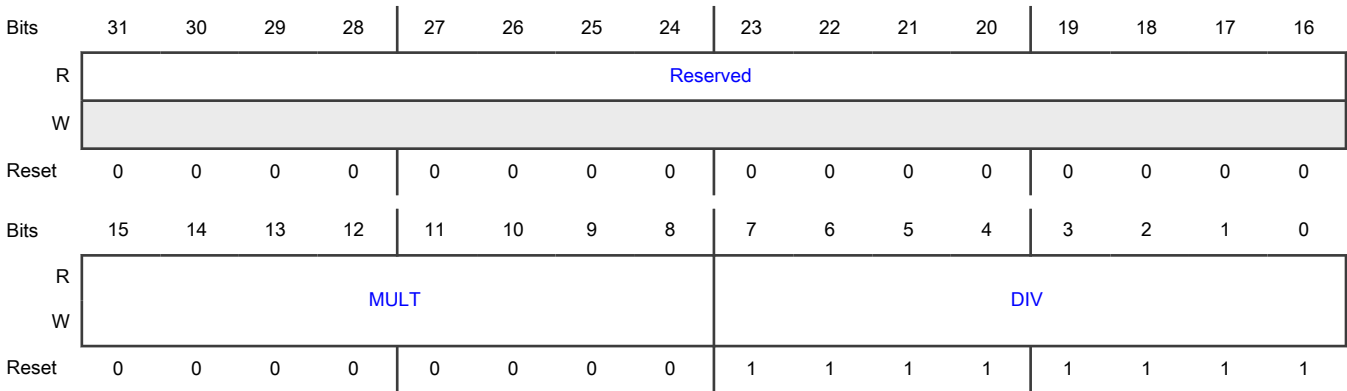
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.69 Flexcomm15 Clock Select (FC15FCLKSEL)

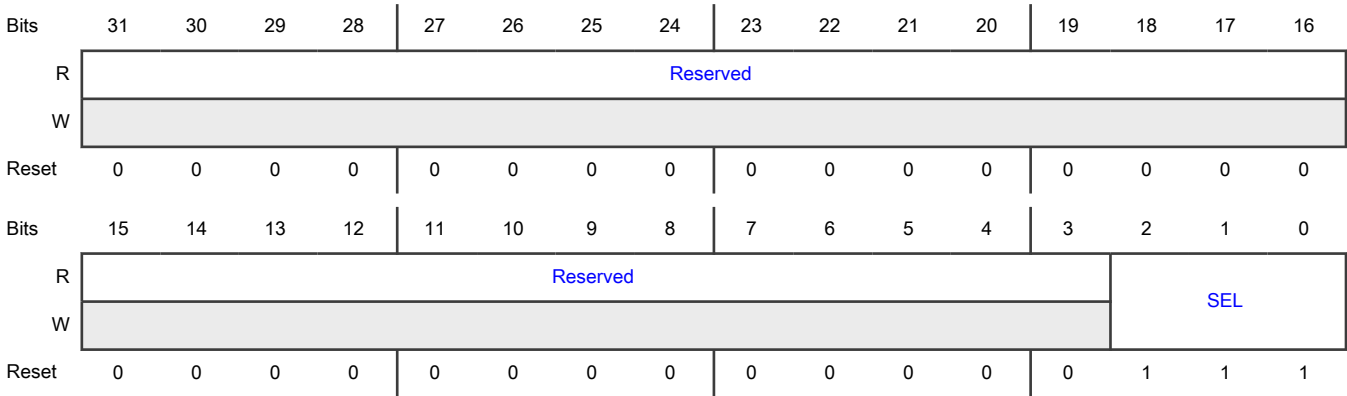
Offset

Register	Offset
FC15FCLKSEL	6E8h

Function

This register selects the clock source for Flexcomm15.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.70 Fractional Rate Generator 16 Clock Select (FRG16CLKSEL)

Offset

Register	Offset
FRG16CLKSEL	700h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm16.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 16 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.71 Fractional Rate Generator 16 Control (FRG16CTL)**Offset**

Register	Offset
FRG16CTL	704h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 16. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

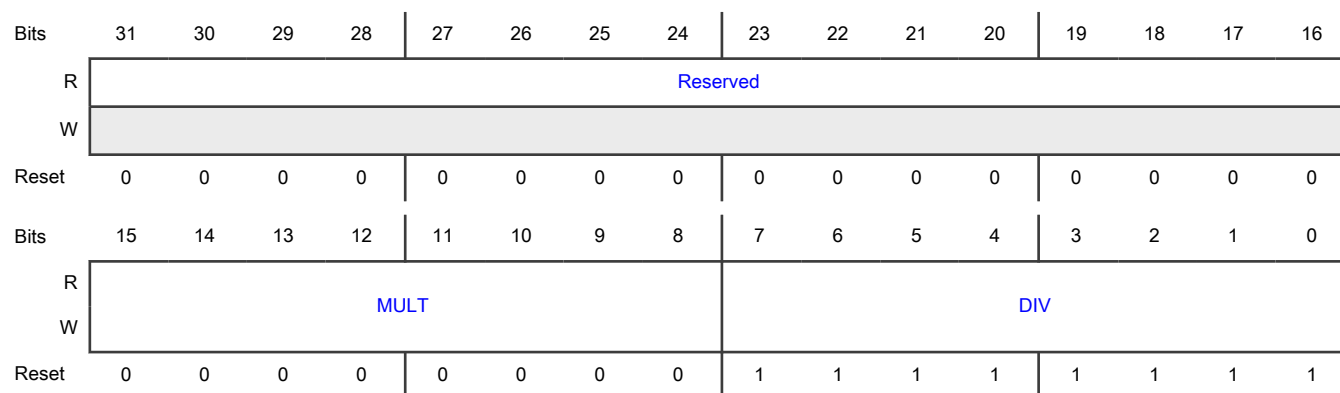
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value + 1. Always set to 0xFF to use with the FRG.

6.5.2.72 Flexcomm16 Clock Select (FC16FCLKSEL)

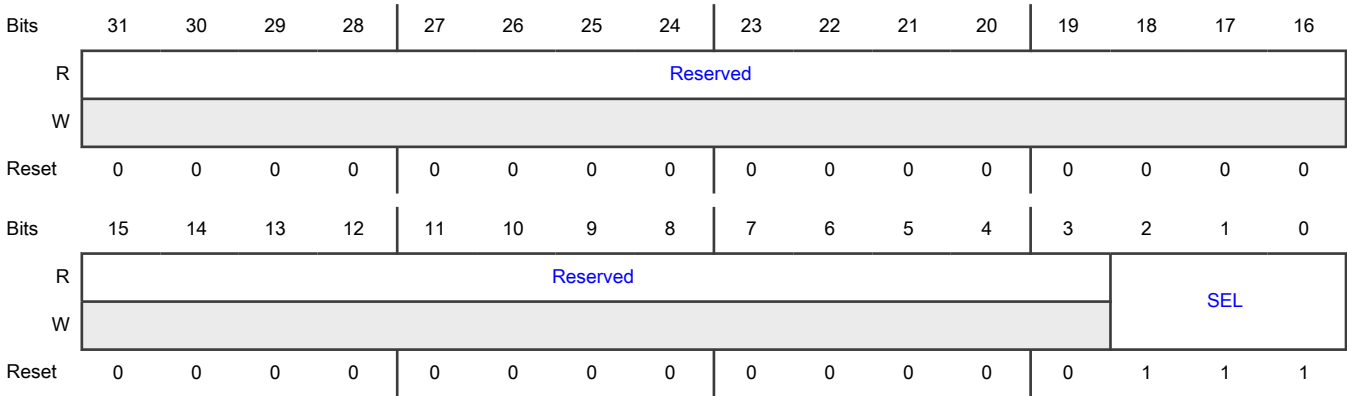
Offset

Register	Offset
FC16FCLKSEL	708h

Function

This register selects the clock source for Flexcomm16.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Flexcomm Functional Clock Source 000b - FRO_DIV4 clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FCn FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.73 Fractional Rate Generator 17 Clock Select (FRG17CLKSEL)

Offset

Register	Offset
FRG17CLKSEL	720h

Function

This register selects the clock source for the Fractional Rate Generator for Flexcomm17.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Fractional Generator 17 Clock Source 000b - Main Clock 001b - FRG PLL Clock 010b - FRO_DIV4 clock 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.74 Fractional Rate Generator 17 Control (FRG17CTL)**Offset**

Register	Offset
FRG17CTL	724h

Function

This register controls the Fractional Rate Generator (FRG) for Flexcomm Interface 17. All Flexcomm Interfaces have, as one of their possible clock sources, a common clock that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator. When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48MHz across voltage. For VDDCORE at 0.9V or above, the FRG output frequency can be up to 96MHz. The High Speed SPI interface, Flexcomm 14, can have a clock up to 50 MHz.

The output rate is:

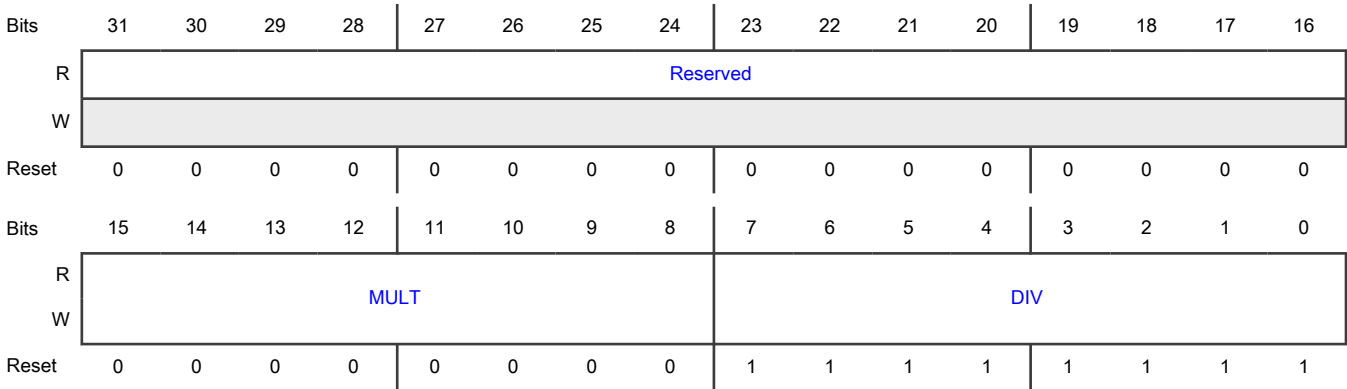
Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV).

The clock used by the fractional rate generator is selected via the related FRGSEL register. The MULT and DIV fields are both 8 bits in size. Because DIV is minus 1 encoded and MULT is not, the possible FRG range is from divide by 1 [1 + (0 / 256)] to divide by 1 + (255 / 256) (just under divide by 2).

NOTE

In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MULT	Numerator of the fractional divider: MULT is equal to the programmed value.
7-0 DIV	Denominator of the fractional divider: DIV is equal to the programmed value +1. Always set to 0xFF to use with the FRG.

6.5.2.75 FlexIO Clock Select (FLEXIOCLKSEL)

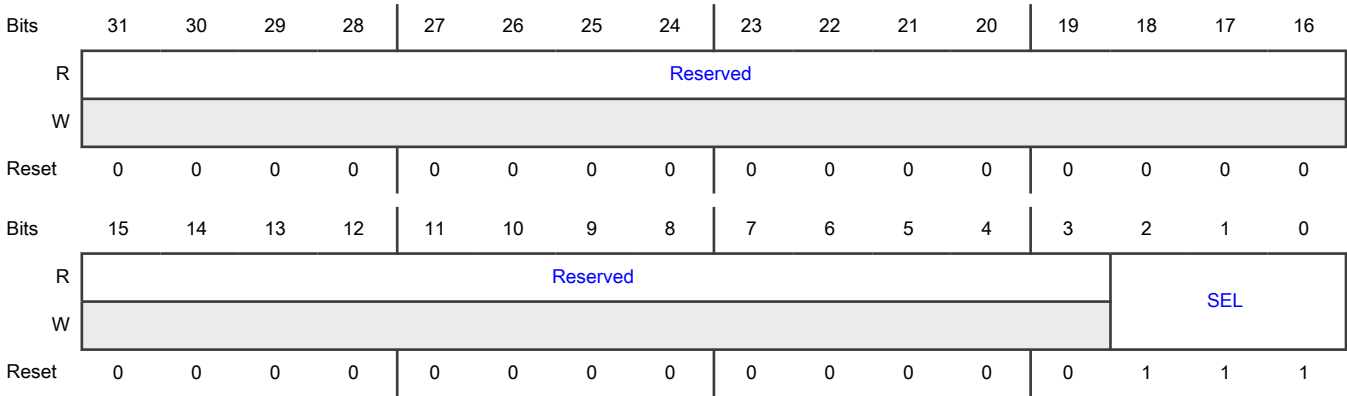
Offset

Register	Offset
FLEXIOCLKSEL	728h

Function

This register selects the clock source for the FlexIO Functional Clock.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	FlexIO Functional Clock Source 000b - FRO_DIV2 Clock 001b - Audio PLL Clock 010b - Master Clock In 011b - FC17 FRG Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.76 FlexIO Clock Divider (FLEXIOCLKDIV)

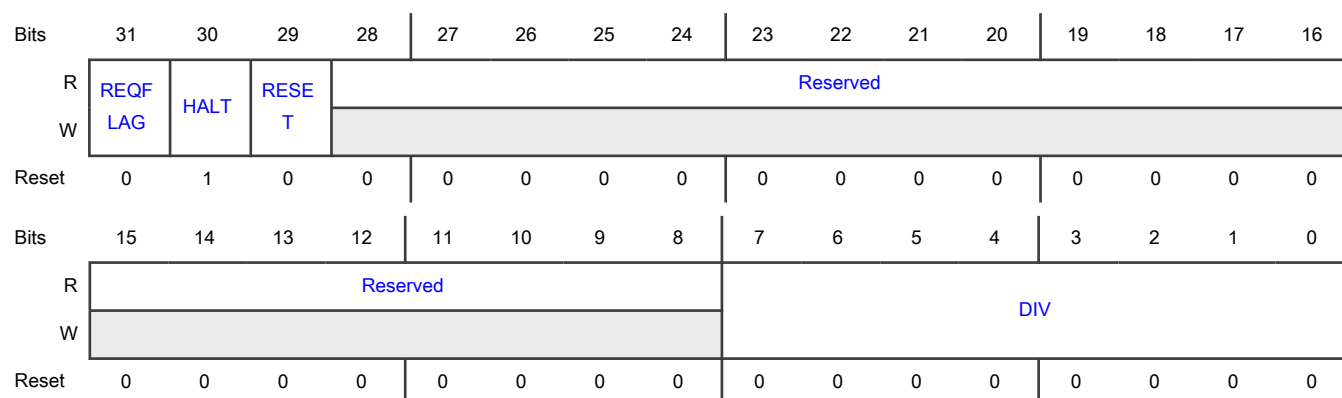
Offset

Register	Offset
FLEXIOCLKDIV	740h

Function

This register controls the divider for the FlexIO Functional Clock Divider.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	FLEXIO Clock Divider Value Selects the divider: 0: Divide by 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.77 Fractional Rate Generator PLL Clock Divider (FRGPLLCLKDIV)

Offset

Register	Offset
FRGPLLCLKDIV	760h

Function

This register controls the divider for the Fractional Rate Generator (FRG) that supplies one of the potential choices for the individual Flexcomm Interface FRGs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	FRG PLL Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.78 DMIC0 Functional Clock Select (DMIC0CLKSEL)

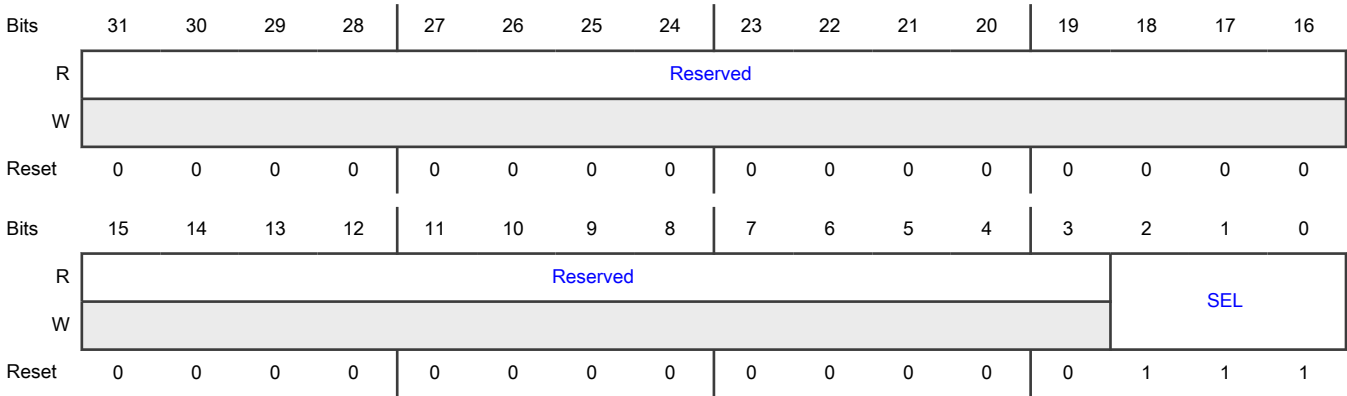
Offset

Register	Offset
DMIC0CLKSEL	780h

Function

This register selects the clock source for DMIC0.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	DMIC Functional Clock Source 000b - FRO Clock (Divided-by-4 selection) 001b - Audio PLL Clock 010b - Master Clock In 011b - Low Power Oscillator Clock (LPOSC) 100b - 32 KHz Wake Clock 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.79 DMIC0 Functional Clock Divider (DMIC0FCLKDIV)

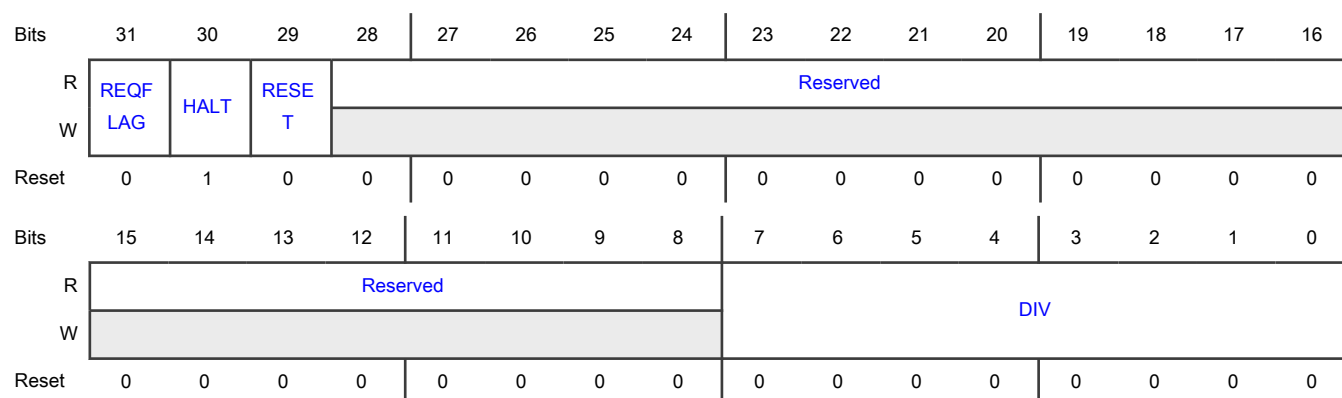
Offset

Register	Offset
DMIC0FCLKDIV	784h

Function

This register controls the divider for the DMIC0 function clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	32 KHz Wake Clock Divider Value Selects the divider: 0: Divide by 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.80 CT32BIT bit timer 0 Functional Clock Select (CT32BIT0FCLKSEL)

Offset

Register	Offset
CT32BIT0FCLKSEL	7A0h

Function

This register selects the clock source for CTIMER0.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	CT32BIT bit timer 0 Functional Clock Source 000b - Main Clock 001b - FRO_DIV1 Clock 010b - Audio PLL Clock 011b - Master Clock In

Table continues on the next page...

Table continued from the previous page...

Field	Function
	100b - 32 KHZ Wake Clock
	101b - Reserved
	110b - Reserved
	111b - None, output gated to reduce power

6.5.2.81 CT32BIT bit timer 1 Functional Clock Select (CT32BIT1FCLKSEL)

Offset

Register	Offset
CT32BIT1FCLKSEL	7A4h

Function

This register selects the clock source for CTIMER1.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	CT32BIT bit timer 1 Functional Clock Source 000b - Main Clock 001b - FRO_DIV1 Clock 010b - Audio PLL Clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011b - Master Clock In 100b - 32 KHZ Wake Clock 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.82 CT32BIT bit timer 2 Functional Clock Select (CT32BIT2FCLKSEL)

Offset

Register	Offset
CT32BIT2FCLKSEL	7A8h

Function

This register selects the clock source for CTIMER2.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	CT32BIT bit timer 2 Functional Clock Source 000b - Main Clock 001b - FRO_DIV1 Clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	010b - Audio PLL Clock
	011b - Master Clock In
	100b - 32 KHZ Wake Clock
	101b - Reserved
	110b - Reserved
	111b - None, output gated to reduce power

6.5.2.83 CT32BIT bit timer 3 Functional Clock Select (CT32BIT3FCLKSEL)

Offset

Register	Offset
CT32BIT3FCLKSEL	7ACh

Function

This register selects the clock source for CTIMER3.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	CT32BIT bit timer 3 Functional Clock Source 000b - Main Clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - FRO_DIV1 Clock
	010b - Audio PLL Clock
	011b - Master Clock In
	100b - 32 KHZ Wake Clock
	101b - Reserved
	110b - Reserved
	111b - None, output gated to reduce power

6.5.2.84 CT32BIT bit timer 4 Functional Clock Select (CT32BIT4FCLKSEL)

Offset

Register	Offset
CT32BIT4FCLKSEL	7B0h

Function

This register selects the clock source for CTIMER4.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0	CT32BIT bit timer 4 Functional Clock Source

Table continues on the next page...

Table continued from the previous page...

Field	Function
SEL	000b - Main Clock 001b - FRO_DIV1 Clock 010b - Audio PLL Clock 011b - Master Clock In 100b - 32 KHZ Wake Clock 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.85 Audio MCLK Clock Select (AUDIOMCLKSEL)

Offset

Register	Offset
AUDIOMCLKSEL	7C0h

Function

This register selects a clock to provide to the MCLK output function. In a system using I2S and/or digital microphone, this should be related to the clock used by those functions.

Diagram

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

Fields

Field	Function
31-3	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 SEL	Audio MCLK Clock Source Select 000b - FRO_DIV8 Clock 001b - AUDIO PLL Clock (Shared Domain) 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.86 Audio MCLK Clock Divider (AUDIOMCLKDIV)

Offset

Register	Offset
AUDIOMCLKDIV	7C4h

Function

This register determines the divider value for the MCLK output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider Status Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Audio MCLK Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.87 CLKOUT Clock Select 0 (CLKOUTSEL0)

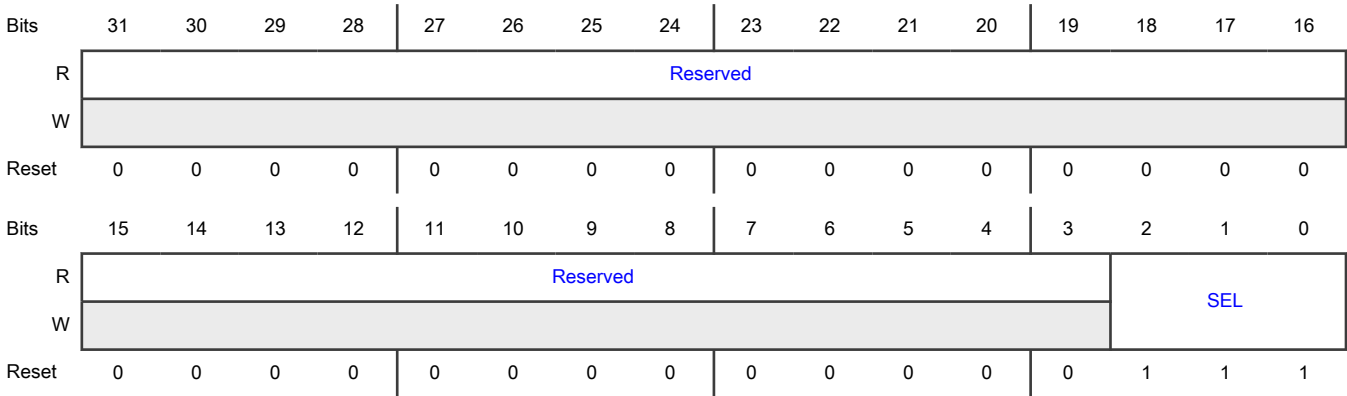
Offset

Register	Offset
CLKOUTSEL0	7E0h

Function

This register selects one of several clock sources to be made available to the final CLKOUT multiplexer.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Clock Output Select 1st Stage 000b - OSC_CLK Clock 001b - Low Power Oscillator Clock (LPOSC) 010b - FRO_DIV2 Clock 011b - Main Clock 100b - DSP Main Clock 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.88 CLKOUT Clock Select 1 (CLKOUTSEL1)

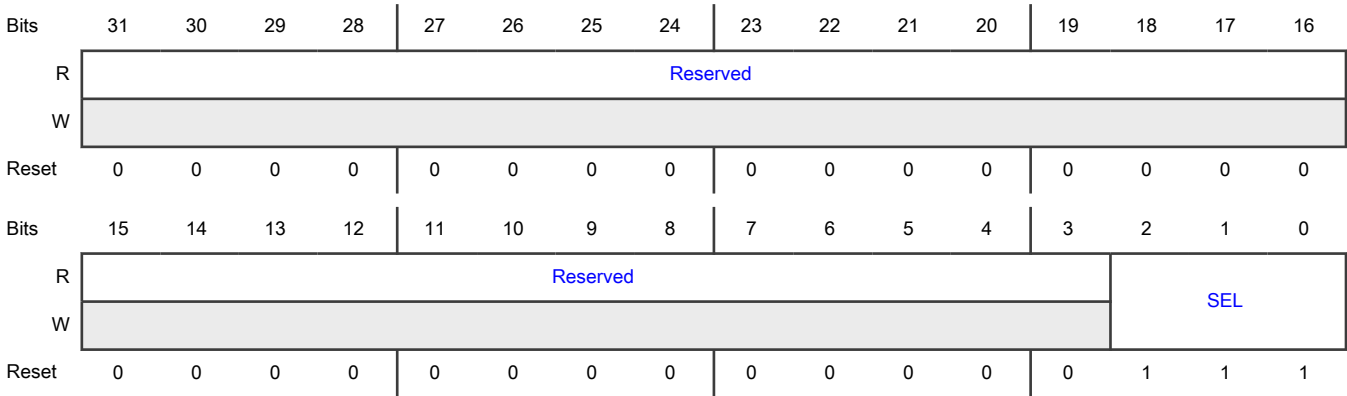
Offset

Register	Offset
CLKOUTSEL1	7E4h

Function

This register selects one of the internal clock sources to be output on the CLKOUT pin.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	Clock Out Source 000b - CLKOUTSEL0 Multiplexed Output 001b - Main System PLL Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - DSP PLL Clock 100b - SYSPLL0 AUX1_PLL_Clock 101b - AUDIO PLL Clock 110b - 32 KHz RTC Clock 111b - None, output gated to reduce power

6.5.2.89 CLKOUT Functional Clock Divider (CLKOUTFCLKDIV)

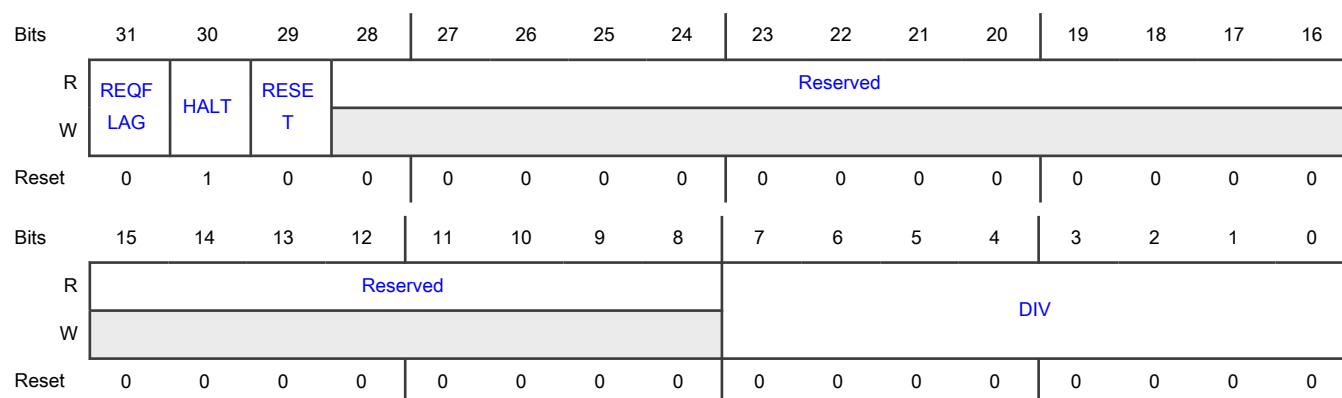
Offset

Register	Offset
CLKOUTFCLKDIV	7E8h

Function

This register determines the divider value for the clock signal on the CLKOUT pin.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock-Out Clock Divider Value Selects the divider: 0: Divide by 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.90 I3C0, I3C1 Functional Clock Select (I3C01FCLKSEL)

Offset

Register	Offset
I3C01FCLKSEL	800h

Function

This register selects the clock source for the I3C function clock.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C0, I3C1 Clock Source 000b - Main Clock 001b - FRO_DIV8 Clock 010b - Reserved 011b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	100b - Reserved
	101b - Reserved
	110b - Reserved
	111b - None, output gated to reduce power

6.5.2.91 I3C0, I3C1 Functional Slow Time Control Clock Select (I3C01FCLKSTCSEL)

Offset

Register	Offset
I3C01FCLKSTCSEL	804h

Function

This register selects the clock source for the I3C slow time control clock.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C0, I3C1 Clock Source 000b - I3C0 FCLK 001b - Low Power Oscillator Clock (LPOSC) 010b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	011b - Reserved
	100b - Reserved
	101b - Reserved
	110b - Reserved
	111b - None, output gated to reduce power

6.5.2.92 I3C0, I3C1 Functional Slow Time Control Clock Divider (I3C01FCLKSTCDIV)

Offset

Register	Offset
I3C01FCLKSTCDIV	808h

Function

This register controls the divider for the I3C slow time control clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished.
NOTE Clock being divided must be running for this status to change.	

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	I3C0, I3C1 Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.93 I3C0, I3C1 Functional Slow Clock Divider (I3C01FCLKSDIV)

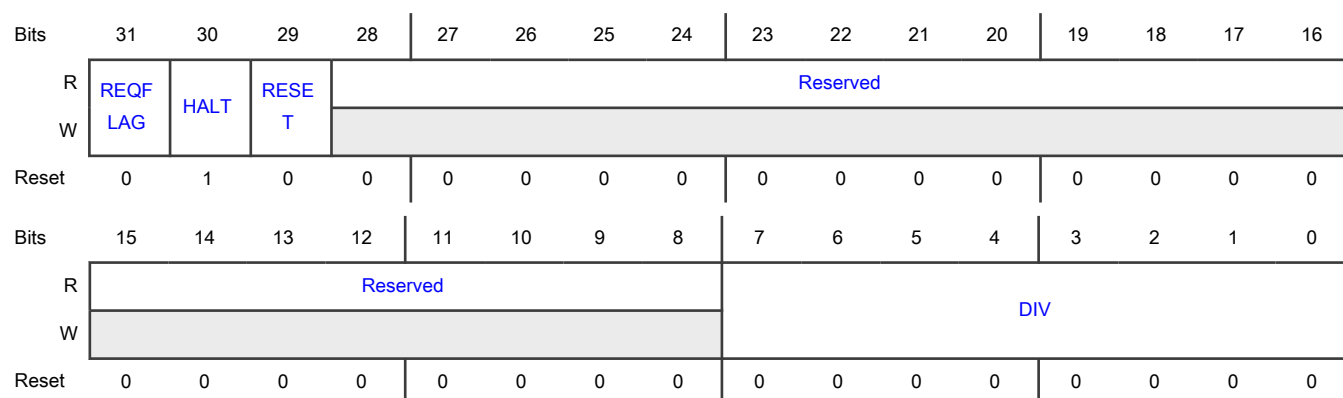
Offset

Register	Offset
I3C01FCLKSDIV	80Ch

Function

This register controls the divider for the slow clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halt the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	I3C0, I3C1 Clock Divider Value Selects the divider: 0: Divide by 1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.94 I3C0, I3C1 Functional Clock Divider (I3C01FCLKDIV)

Offset

Register	Offset
I3C01FCLKDIV	810h

Function

This register controls the divider for the I3C function clock.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF	HALT	RESE	Reserved												
W	LAG		T													
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider Status Flag Set (to 1) when a change is made to the divider value; cleared (to 0) when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
30 HALT	Halts the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter Can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	I3C0, I3C1 Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2 • ... • 255: Divide by 256

6.5.2.95 I3C01 Functional Clock Select (I3C01FCLKSTSTCLKSEL)

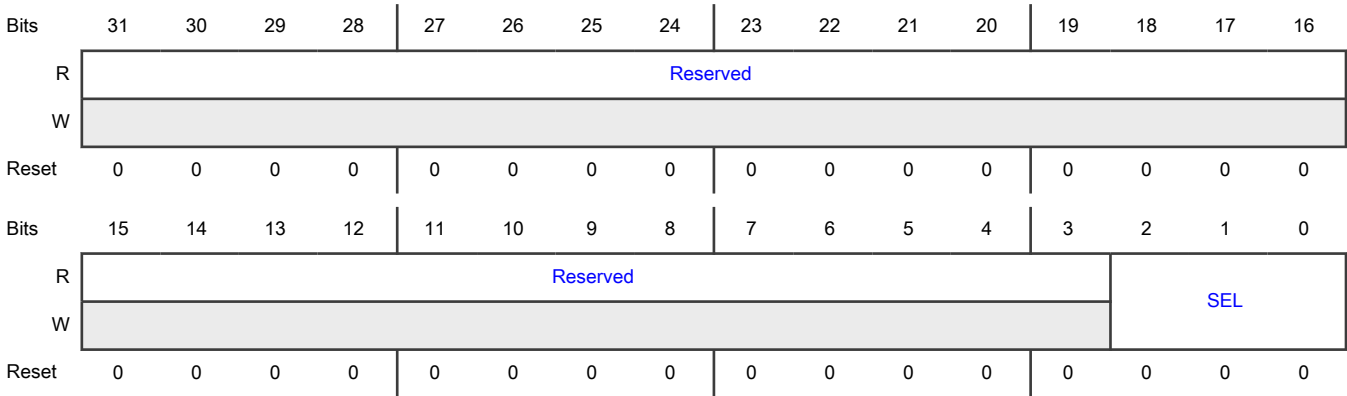
Offset

Register	Offset
I3C01FCLKSTSTCLKSEL	814h

Function

Selects the I3C0, I3C1 functional clock source.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	I3C0, I3C1 FCLK Test Clock Source 000b - Low Power Oscillator Clock (LPOSC) 001b - Reserved 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.96 Watchdog Timer 1 Functional Clock Select (WDT1FCLKSEL)

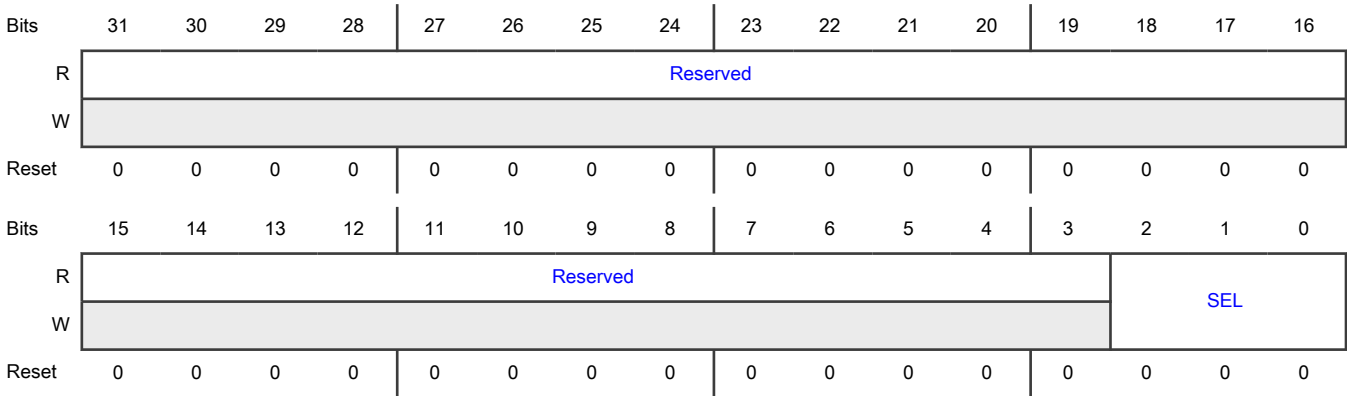
Offset

Register	Offset
WDT1FCLKSEL	820h

Function

Selects the WDT1 Functional Clock Source.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	WDT1 Functional Clock Source 000b - Low Power Oscillator Clock (LPOSC) 001b - Reserved 010b - Reserved 011b - Reserved 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.97 Analog Comparator 0 Clock Select (ACMP0CLKSEL)

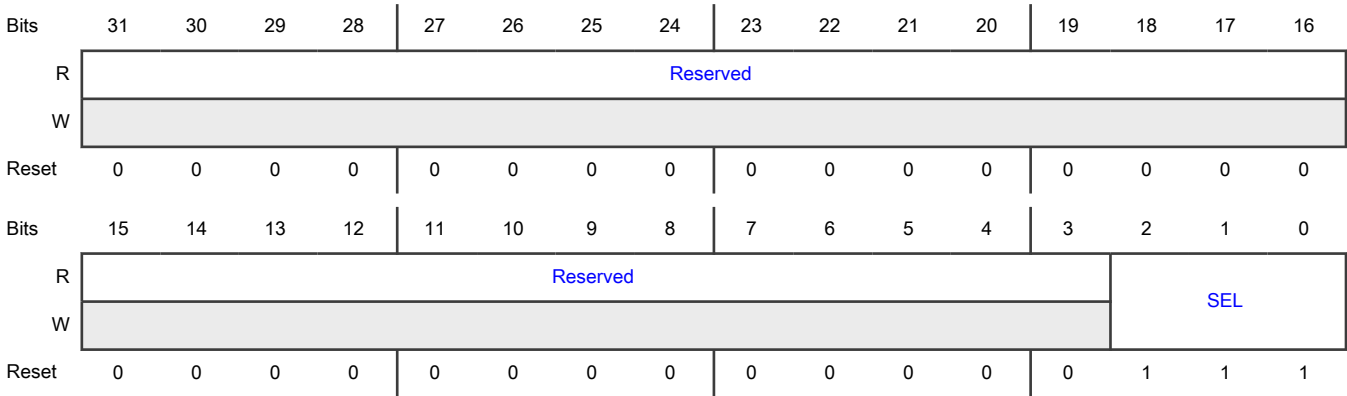
Offset

Register	Offset
ACMP0CLKSEL	830h

Function

This register selects the clock source for the analog comparator.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 SEL	ACMP0 Fast Functional Clock Source 000b - Main Clock 001b - FRO_DIV4 Clock 010b - SYSPLL0 AUX0_PLL_Clock 011b - SYSPLL0 AUX1_PLL_Clock 100b - Reserved 101b - Reserved 110b - Reserved 111b - None, output gated to reduce power

6.5.2.98 Analog comparator 0 FCLK divider (ACMP0FCLKDIV)

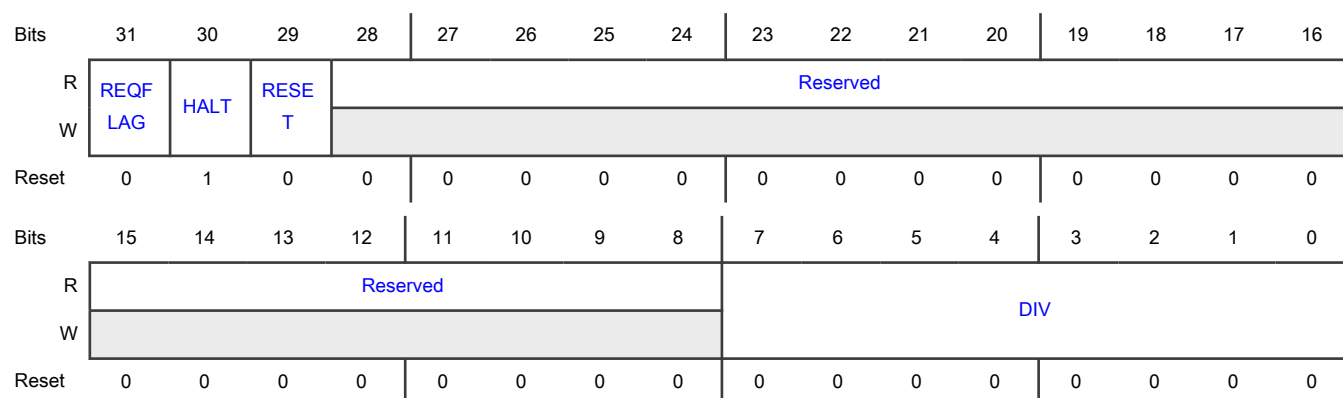
Offset

Register	Offset
ACMP0FCLKDIV	834h

Function

This register controls the divider for the analog comparator function clock.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Status Flag Set when a change is made to the divider value; cleared when the divider value change has finished. NOTE Clock being divided must be running for this status to change. 0b - The Divider change has finished (clock being divided must be running for this status to change). 1b - The Divider value has changed.
30 HALT	Halts the Divider Counter The intent is to allow the divider's clock source to be changed, without causing a glitch at the output. 0b - No effect 1b - Halt (stop) the Divider Counter
29 RESET	Reset the Divider Counter This bit can be used to make sure a new divider value is used immediately, instead of completing the previous count. 0b - No effect 1b - Reset the Divider Counter
28-8 —	Reserved
7-0 DIV	Clock Out Clock Divider Value Selects the divider: • 0: Divide by 1 • 1: Divide by 2

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• ... • 255: Divide by 256

Chapter 7

Power Control API

7.1 Overview

The power profiles and power control APIs can be implemented using the power library from the SDK software package.

7.1.1 Features

- Simple APIs to control power consumption and wake-up in all low power modes: sleep, deep-sleep, and deep power-down.
- Prepare the part to enter low power modes: deep-sleep, and deep power-down.
- Manage power consumption for sleep and active modes.
- Configure wake-up from deep-sleep via functions enabled by bits in the PDRUNCFG registers.

7.2 Functional Description

Control of device power consumption or entry to low power modes can be configured through simple calls to the low power profile API.

APIs exist to:

- Set up reduced power modes.

NOTE

Disable all interrupts before making calls to the power profile API. The interrupts can be re-enabled after the power profile API calls have completed.

7.2.1 POWER_UpdateOscSettlingTime

This routine sets up the oscillator stabilization time.

Table 18. POWER_UpdateOscSettlingTime API

Routine	POWER_UpdateOscSettlingTime
SDK prototype	void POWER_UpdateOscSettlingTime(uint32_t osc_delay);
Input parameter	Param0: osc_delay: Oscillator stabilization time in microseconds.
Result	None
Description	Update crystal oscillator settling time.

7.2.2 POWER_ApplyPD

This routine applies any updated bits in Syscon PDRUNCFG registers.

Table 19. POWER_ApplyPD API

Routine	POWER_ApplyPD
SDK prototype	void POWER_ApplyPD (void);

Table continues on the next page...

Table 19. POWER_ApplyPD API (continued)

Routine	POWER_ApplyPD
Input parameter	None
Result	None
Description	Apply updated PDRUNCFG bits in SYSCTL0.

7.2.3 POWER_EnablePD

This routine applies to enable PDRUNCFG bit in the SYSCTL0. Note that enabling the bit powers down the peripheral.

Table 20. POWER_EnablePD API

Routine	POWER_EnablePD
SDK prototype	void POWER_EnablePD(pd_bit_t en);
Input parameter	pd_bit_t en
Result	None
Description	Enable PDRUNCFG bits in SYSCTL0.

7.2.4 POWER_DisablePD

This routine applies to disable PDRUNCFG bit in the SYSCTL0. Note that enabling the bit powers up the peripheral.

Table 21. POWER_DisablePD API

Routine	POWER_DisablePD
SDK prototype	void POWER_DisablePD(pd_bit_t en);
Input parameter	pd_bit_t en
Result	None
Description	Disable PDRUNCFG bits in SYSCTL0.

7.2.5 POWER_ClearEventFlags

This routine allows clearing selected event flags in the PMC.

Table 22. POWER_ClearEventFlags API

Routine	POWER_ClearEventFlags
SDK prototype	void POWER_ClearEventFlags (uint32_t statusMask);
Input parameter	Param0: statusMask: A bitmask of event flags that are to be cleared.

Table continues on the next page...

Table 22. POWER_ClearEventFlags API (continued)

Routine	POWER_ClearEventFlags
Result	None
Description	Clears selected events in the PMC FLAGS register.

Table 23. Event flags (statusMask)

Flag name	Description
kPMC_FLAGS_PORCORE	POR triggered by the VDDCORE POR monitor 0 = no 1 = yes
kPMC_FLAGS_POR1V8	VDD1V8 power on event detected since last cleared. 0 = no 1 = yes
kPMC_FLAGS_PORAO18	VDD_AO1V8 power on event detected since last cleared. 0 = no 1 = yes
kPMC_FLAGS_LVDCORE	LVD tripped since last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_HVDCORE	HVD tripped since last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_HVD1V8	VDD1V8 HVD tripped since last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_RTC	RTC wake-up detected since last time flag was cleared. 0 = no 1 = yes
kPMC_FLAGS_AUTOWK	PMC Auto wake-up caused a deep-sleep wake-up and interrupt. 0 = no 1 = yes
kPMC_FLAGS_INTNPADF	Pad interrupt caused a wake-up or interrupt event since the last time this flag was cleared. 0 = no

Table continues on the next page...

Table 23. Event flags (statusMask) (continued)

Flag name	Description
	1 = yes
kPMC_FLAGS_RESETPAD	Reset pad wake-up caused a wake-up or reset event since the last time this bit was cleared. 0 = no 1 = yes
kPMC_FLAGS_DEEPPD	Deep power-down was entered since the last time this flag was cleared. 0 = no 1 = yes

7.2.6 POWER_GetEventFlags

This routine provides the state of event flags in the PMC.

Table 24. POWER_GetEventFlags API

Routine	POWER_GetEventFlags
SDK prototype	uint32_t POWER_GetEventFlags (void);
Input parameter	None
Result	PMC event flags register value.
Description	Get the PMC event flags state

7.2.7 POWER_EnableInterrupts

This routine allows enabling selected PMC interrupts.

Table 25. POWER_EnableInterrupts API

Routine	POWER_EnableInterrupts
SDK prototype	void POWER_EnableInterrupts (uint32_t interruptMask);
Input parameter	Param0: interruptMask: A bitmask of interrupts to enable.
Result	None
Description	Enable the selected PMC interrupt requests.

Table 26. PMC interrupts (interruptMask)

Interrupt name	Description
kPMC_INT_LVDCORE	VDDCORE Low-Voltage Detector Interrupt Enable.

Table continues on the next page...

Table 26. PMC interrupts (interruptMask) (continued)

Interrupt name	Description
kPMC_INT_HVDCORE	VDDCORE High-Voltage Detector Interrupt Enable.
kPMC_INT_HVD1V8	VDD1V8 High-Voltage Detector Interrupt Enable.
kPMC_INT_AUTOWK	PMC automatic wake-up enable and interrupt enable.
kPMC_INT_INTRPAD	Interrupt pad deep power-down and deep-sleep wake-up & interrupt enable.

7.2.8 POWER_DisableInterrupts

This routine allows disabling selected PMC interrupts.

Table 27. POWER_DisableInterrupts API

Routine	POWER_DisableInterrupts
SDK prototype	void POWER_DisableInterrupts (uint32_t interruptMask);
Input parameter	Param0: interruptMask: A bitmask of interrupts to disable. See POWER_EnableInterrupts .
Result	None
Description	Disable the selected PMC interrupt requests.

7.2.9 POWER_SetAnalogBuffer

This routine allows enabling or disabling the analog buffer to allow ADC and/or comparator operation.

Table 28. POWER_SetAnalogBuffer API

Routine	POWER_SetAnalogBuffer
SDK prototype	void POWER_SetAnalogBuffer (bool enable);
Input parameter	Param0: enable: Set to true to enable analog buffer for references, false to disable.
Result	None
Description	Set the PMC analog buffer for references.

7.2.10 POWER_SetPadVolRange

This routine configures pad voltage level. Selecting wide voltage range uses more power due to enabled voltage detector.

Note: use caution when calling this API. If the actual pad supply is beyond the set range, the device may be damaged.

Table 29. POWER_SetPadVolRange API

Routine	POWER_SetPadVolRange
SDK prototype	void POWER_SetPadVolRange (const power_pad_vrange_t *config);

Table continues on the next page...

Table 29. POWER_SetPadVolRange API (continued)

Routine	POWER_SetPadVolRange
Input parameter	Param0: power_pad_vrange_t: Value using the parameter structure.
Result	None
Description	Configures the pad voltage range to match the voltages applied to the VDDIO_0, VDDIO_1, and VDDIO_2 pins in the application.

Table 30. Structure of power_pad_vrange_t parameter

Name	Bits	Description
Vdde0Range	2	VDDE0 voltage range for VDDIO_0.
Vdde1Range	2	VDDE1 voltage range for VDDIO_1.
Vdde2Range	2	VDDE2 voltage range for VDDIO_2.
Vdde3Range	2	VDDE3 voltage range for VDDIO_3.
Vdde4Range	2	VDDE4 voltage range for VDDIO_4.
Reserved	26	Reserved

Table 31. Pad voltage range values (power_pad_vrange_val_t)

power_pad_vrange_val_t	Value	Description
kPadVol_171_360	0	Voltage from 1.71V to 3.60V (wide voltage range).
kPadVol_171_198	1	Voltage from 1.71V to 1.98V (low voltage range),
kPadVol_300_360	2	Voltage from 3.00V to 3.60V (high voltage range).

7.2.11 POWER_EnterRbb

This routine enters RBB (Reverse Body Bias) mode.

Table 32. POWER_EnterRbb API

Routine	POWER_EnterRbb
SDK prototype	void POWER_EnterRbb (void);
Input parameter	None
Result	None
Description	PMC Enter RBB mode function call.

7.2.12 POWER_EnterFbb

This routine enters FBB (Forward Body Bias) mode.

Table 33. POWER_EnterFbb API

Routine	POWER_EnterFbb
SDK prototype	void POWER_EnterFbb (void);
Input parameter	None
Result	None
Description	PMC Enter FBB mode function call.

7.2.13 POWER_EnterNbb

This routine enters NBB (Normal Body Bias) mode.

Table 34. POWER_EnterNbb API

Routine	POWER_EnterNbb
SDK prototype	void POWER_EnterNbb (void);
Input parameter	None
Result	None
Description	PMC exit RBB & FBB mode function call.

7.2.14 POWER_SetLdoVoltageForFreq

This routine sets the on-chip regulator to match power requirements based on the operating frequency of the CM33 and the HiFi4 DSP.

NOTE

If the LVD falling trip voltage is higher than the required core voltage for particular frequency, the LVD voltage will be decreased to safe level to avoid an unexpected LVD reset or interrupt event.

Table 35. POWER_SetLdoVoltageForFreq API

Routine	POWER_SetLdoVoltageForFreq
SDK prototype	void POWER_SetLdoVoltageForFreq (uint32_t main_clk_freq, uint32_t dsp_main_clk_freq);
Input parameter	Param0: main_clk_freq: Main clock frequency value in Hz. Param1: dsp_main_clk_freq: DSP main clock frequency value in Hz.
Result	None
Description	Set Ldo voltage function call.

7.2.15 POWER_SetLvdFallingTripVoltage

This routine sets the trip level for low voltage detection of vddcore falling voltage.

NOTE

At startup, the boot code sets a default level for the trip level. This API must be used to reduce the trip level prior to user code reducing the chip operating voltage.

Table 36. POWER_SetLvdFallingTripVoltage

Routine	POWER_SetLvdFallingTripVoltage
SDK prototype	void POWER_SetLvdFallingTripVoltage (power_lvd_falling_trip_vol_val_t volt);
Input parameter	Param0: volt: Target LVD voltage to set.
Result	None
Description	Sets the vddcore low voltage detection falling trip voltage.

Table 37. Trip voltage values (power_lvd_falling_trip_vol_val_t)

power_lvd_falling_trip_vol_val_t	Value	Description
kLvdFallingTripVol_720	0	Trip voltage 720 mV
kLvdFallingTripVol_735	1	Trip voltage 735 mV
kLvdFallingTripVol_750	2	Trip voltage 750 mV
kLvdFallingTripVol_765	3	Trip voltage 765 mV
kLvdFallingTripVol_780	4	Trip voltage 780 mV
kLvdFallingTripVol_795	5	Trip voltage 795 mV
kLvdFallingTripVol_810	6	Trip voltage 810 mV
kLvdFallingTripVol_825	7	Trip voltage 825 mV
kLvdFallingTripVol_840	8	Trip voltage 840 mV
kLvdFallingTripVol_855	9	Trip voltage 855 mV
kLvdFallingTripVol_870	10	Trip voltage 870 mV
kLvdFallingTripVol_885	11	Trip voltage 885 mV
kLvdFallingTripVol_900	12	Trip voltage 900 mV
kLvdFallingTripVol_915	13	Trip voltage 915 mV
kLvdFallingTripVol_930	14	Trip voltage 930 mV
kLvdFallingTripVol_945	15	Trip voltage 945 mV

7.2.16 POWER_EnterSleep

This routine enters Sleep mode.

Table 38. POWER_EnterSleep API

Routine	POWER_EnterSleep
SDK prototype	void POWER_EnterSleep (void);
Input parameter	None
Result	None
Description	Configures and enters in Sleep low power mode.

7.2.17 POWER_EnterDeepSleep

This routine enters Deep-Sleep mode.

Table 39. POWER_EnterDeepSleep API

Routine	POWER_EnterDeepSleep
SDK prototype	void POWER_EnterDeepSleep (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDSLEEPCFG0 through PDSLEEPCFG3 registers to identify functions to be powered on during Deep-sleep mode.
Result	None
Description	PMC Deep-sleep function call.

7.2.18 POWER_EnterDeepPowerDown

This routine enters Deep Power-Down mode.

Table 40. POWER_EnterDeepPowerDown API

Routine	POWER_EnterDeepPowerDown
SDK prototype	void POWER_EnterDeepPowerDown (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDSLEEPFG0 through PDSLEEPCFG3 registers to identify functions to be powered on during Deep-sleep mode
Result	None
Description	PMC Deep Power-down function call.

7.2.19 POWER_EnterFullDeepPowerDown

This routine enters Full Deep Power-Down mode.

Table 41. POWER_EnterFullDeepPowerDown API

Routine	POWER_EnterFullDeepPowerDown
SDK prototype	void POWER_EnterFullDeepPowerDown (const uint32_t exclude_from_pd[4]);
Input parameter	Param0: exclude_from_pd[4]: Bit mask of the PDSLEEPCFG0 through PDSLEEPCFG3 registers to identify functions to be powered on during Deep-sleep mode.
Result	None
Description	PMC Full Deep Power-down function call.

7.2.20 POWER_EnterPowerMode

This routine allows entering reduced power modes.

Table 42. POWER_EnterPowerMode API

Routine	POWER_EnterPowerMode
SDK prototype	void POWER_EnterPowerMode (power_mode_cfg_t mode, const uint32_t exclude_from_pd[4]);
Input parameter	Param0: power_mode_cfg_t mode: Power mode. Param1: exclude_from_pd[4]: Bit mask of the PDSLEEPCFG0 through PDSLEEPCFG3 registers to identify functions to be powered on during the power mode selected.
Result	None
Description	Power Library API to enter different power mode.

Table 43. Power modes (power_mode_cfg_t)

Power mode name	Description
kPmu_Sleep	Sleep mode.
kPmu_Deep_Sleep	Deep-sleep mode.
kPmu_Deep_PowerDown	Deep power-down mode.
kPmu_Full_Deep_PowerDown	Full deep power-down mode.

7.2.21 EnableDeepSleepIRQ

This routine allows enabling selected interrupts for Deep-Sleep mode.

Table 44. EnableDeepSleepIRQ API

Routine	EnableDeepSleepIRQ
SDK prototype	void EnableDeepSleepIRQ (IRQn_Type interrupt);

Table continues on the next page...

Table 44. EnableDeepSleepIRQ API (continued)

Routine	EnableDeepSleepIRQ
Input parameter	Param0: interrupt: this is an interrupt number used to set a bit in SYSCTL0_STARTEN0 or SYSCTL0_STARTEN1, see Table 9 in NVIC chapter.
Result	None
Description	Enable specific interrupt for wake-up from deep-sleep mode.

7.2.22 DisableDeepSleepIRQ

This routine allows disabling selected interrupts for Deep-Sleep mode.

Table 45. DisableDeepSleepIRQ API

Routine	DisableDeepSleepIRQ
SDK prototype	void DisableDeepSleepIRQ (IRQn_Type interrupt);
Input parameter	Param0: interrupt: this is an interrupt number used to clear a bit in SYSCTL0_STARTEN0 or SYSCTL0_STARTEN1, see Table 9 in NVIC chapter.
Result	None
Description	Disable specific interrupt for wake-up from deep-sleep mode.

7.2.23 POWER_GetLibVersion

This routine returns the version of the Power API library.

Table 46. POWER_GetLibVersion API

Routine	POWER_GetLibVersion
SDK prototype	<pre>uint32_t POWER_GetLibVersion (void);</pre>
Input parameter	None
Result	Version number of the power library
Description	Power Library API to return the library version.

7.2.24 POWER_SetDeepSleepClock

This routine applies to the clock source of main clock before entering deep sleep.

Table 47. POWER_SetDeepSleepClock

Routine	POWER_SetDeepSleepClock
SDK prototype	void POWER_SetDeepSleepClock(power_deep_sleep_clk_t clk);
Input parameter	clk : clock source of main clock.
Result	None
Description	Set deep sleep clock source of main clock.

Chapter 8

Power Management Controller (PMC)

8.1 Chip-specific PMC information

Table 48. Reference links to related information

Topic	Related module	Reference
Full description	PMC	Power Management Controller
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

8.1.1 Module instances

This device has one instance of the PMU module.

8.2 Overview

The Power Management module on the chip includes power switches and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes. Power modes apply to the Core and its subsystems, and are controlled by the Core.

To save power, the MCU is designed to allow lower voltages for VDDCORE supply when clocking at lower frequencies or when in Deep Sleep mode. And VDDCORE supply can be removed completely when in the power-down modes. When using the on-chip regulator to supply VDDCORE, the PMC controls the voltage level. When using an external PMIC, the PMC signals mode changes to the PMIC using the PMIC_MODE pins.

8.3 Features

The power management module includes the following features:

- Support for following power management modes in order from highest to lowest power consumption:
 1. Active
 2. Sleep
 3. Deep-sleep
 4. Deep power-down
 5. Full deep power-down
- Power usage controlled by SYSCTL register settings, regulator settings controlled via a Power API, and the operating mode of the CPU.
- Power to the chip supplied via two power domains.
 - The main power domain VDDCORE supplies power to the core, peripheral, memories, inputs and outputs.
 - An always-on power domain powered by VDD_AO1V8, includes the RTC and wake-up timer. This domain always has power as long as sufficient voltage is supplied to VDD_AO1V8. The PMIC_IRQ_N pin is in the VDD_AO1V8 domain, and provides a wake source even when other supply rails are removed. Please refer the chip datasheet for details.

8.4 Functional Description

8.4.1 Power Modes

The Power Management module implements five basic power modes.

- Active
- Sleep
- Deep Sleep
- Deep Power Down
- Full Deep Power Down

The voltage levels, regulation mode, body biasing, IP power gating are very flexibly defined by software during active and deep sleep modes. Deep power down mode returns to active mode whenever a wakeup signal or an always on domain POR occurs.

The power modes of the chip are controlled by the SYSCTL PDRUNCFG and PDSLEEPCFG registers.

8.4.1.1 Active mode

In Active mode, clocks to the CPU, memories and peripherals are enabled. The chip is in Active mode after reset and the default power configuration is determined by the boot values of the PDRUNCFG and PSCCTL registers. The power configuration can be changed during run time. Most blocks may be in a normal power, low power, or off state, as defined in software. After reset DSP, MIPI and most of memory is disabled.

8.4.1.1.1 Power configuration in Active mode

Power consumption in Active mode is determined by the following:

- The PSCCTL0-PSCCTL2 (See CLKCTL0 and CLKCTL1 registers) control which memories and peripherals are running. In order to save power, functions that are not needed by the application should be turned off. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on when they will be needed.
- The power to various analog blocks (RAMs, PLL, oscillators) can be controlled individually through the PDRUNCFG registers. As with clock controls, these blocks should generally be turned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock sources for system functions may be selected from among several sources (see CLKCTL). One or more PLLs may be used as well (see CLKCTL). Less power is consumed at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If a PLL is not needed, it should be turned off to save power. Also, running PLLs at a lower VCO frequency saves power.
- Many peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements.

8.4.1.2 Sleep mode

In sleep mode, the clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the PSCCTL registers, continue operation during sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static, other than automatic peripheral operation.

The POWER_EnterSleep API provides an easy way to enter sleep mode.

8.4.1.2.1 Programming sleep mode

The following steps allow entry to sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part.
2. Alter PDRUNCFG registers if needed to reflect any functions that should be on or off during sleep mode.
3. Call the power API to enter sleep mode.

8.4.1.2.2 Power configuration in sleep mode

Power consumption in sleep mode is configured by the same settings as in Active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the PDRUNCFG and PSCCTL registers.

8.4.1.2.3 Wake-up from sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor, or when a reset occurs. After a wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG and PSCCTL registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

8.4.1.3 Deep Sleep mode

Deep-sleep mode is configurable and can potentially turn off nearly all on-chip power consumption other than the on-chip power supply, with the cost of a longer wake-up time. VDDCORE supply voltage can also be reduced to the minimum retention level in this mode.

Deep-sleep mode affects the entire system, the clocks to CPUs are shut down and, if not configured, the peripherals receive no internal clocks. Individual blocks may be in a normal power, low power, or off state, as defined in software. Device registers, and SRAMs that are not shut down maintain their contents. Some device features can be automatically disabled by setting up the value of the PDSLEEPCFG registers (see SYSCTL). Entry to this mode can only be accomplished by the CM33. Through the power profiles API, selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, and Micro-tick Timer can be left running in deep sleep mode.

8.4.1.3.1 Entering Deep Sleep Mode

In deep-sleep mode, the system clock to the processor is disabled. Analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock and all peripheral clocks are disabled. Primary clock sources are disabled by default. Power consumed by analog peripherals and dynamic power used by the processor itself, memory systems and related controllers, and internal buses (other than those specifically designated to remain functional) is eliminated. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

GPIO Pin Interrupts, GPIO Group Interrupts, and selected peripherals such as USB, DMIC, SPI, I2C, USART, WWDT, RTC, and Micro-tick Timer and primary clock sources (such as FRO), the RTC oscillator, and the watchdog oscillator can be left running in deep sleep mode. In some cases, DMA can operate in deep-sleep mode.

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled in the call to the power API function.

The following steps allow entry to deep-sleep mode:

1. Enable potential wake-up interrupts in the NVIC.
2. Call the POWER_EnterDeepSleep API with parameters to enable selected functions and enable wake-up sources.

Clock and wakeup source considerations:

1. If the main clock oscillator source is powered down by PDSLEEPCFG during deep sleep then
 - PDSLEEPCFG0[MAINCLKSHUTOFF] must be set
 - MAINCLK source must be FRO_DIV4 or 1M_LPOSC when entering Deep Sleep
2. Else if the main clock oscillator source remains powered/running during deep sleep then
 - If PDSLEEPCFG0[MAINCLKSHUTOFF] = 1 then
 - MAINCLKSAFETY must be set to proper delay value
 - The oscillator source can be any source
 - Else
 - MAINCLKSAFETY has no effect BUT any changes caused by differences in PDSLEEPCFG & PDRUNCFG will still occur and take the same amount of time calculated by the MAINCLKSAFETY.
 - If the wakeup is due to a HWWAKE event and there is an overlapping normal wakeup event, then the CPU will start executing code immediately. This should be avoided if memories are being turned back on and the CPU has no access. A proper software coding should avoid this situation where a peripheral trigger both type of wakeup simultaneously.
 - Else if the wakeup is due to a HWWAKE (This may not be a useful case – the main clock is already running, and the wakeup wouldn't wakeup the ARM CPU).
 - Then, the configuration change due to PDRUNCFG will occur while the peripherals continue to run.
 - Else (the wakeup is due to a normal wakeup/interrupt)
 - The CPU stays in sleep until the PMC config changes to PDRUNCFG

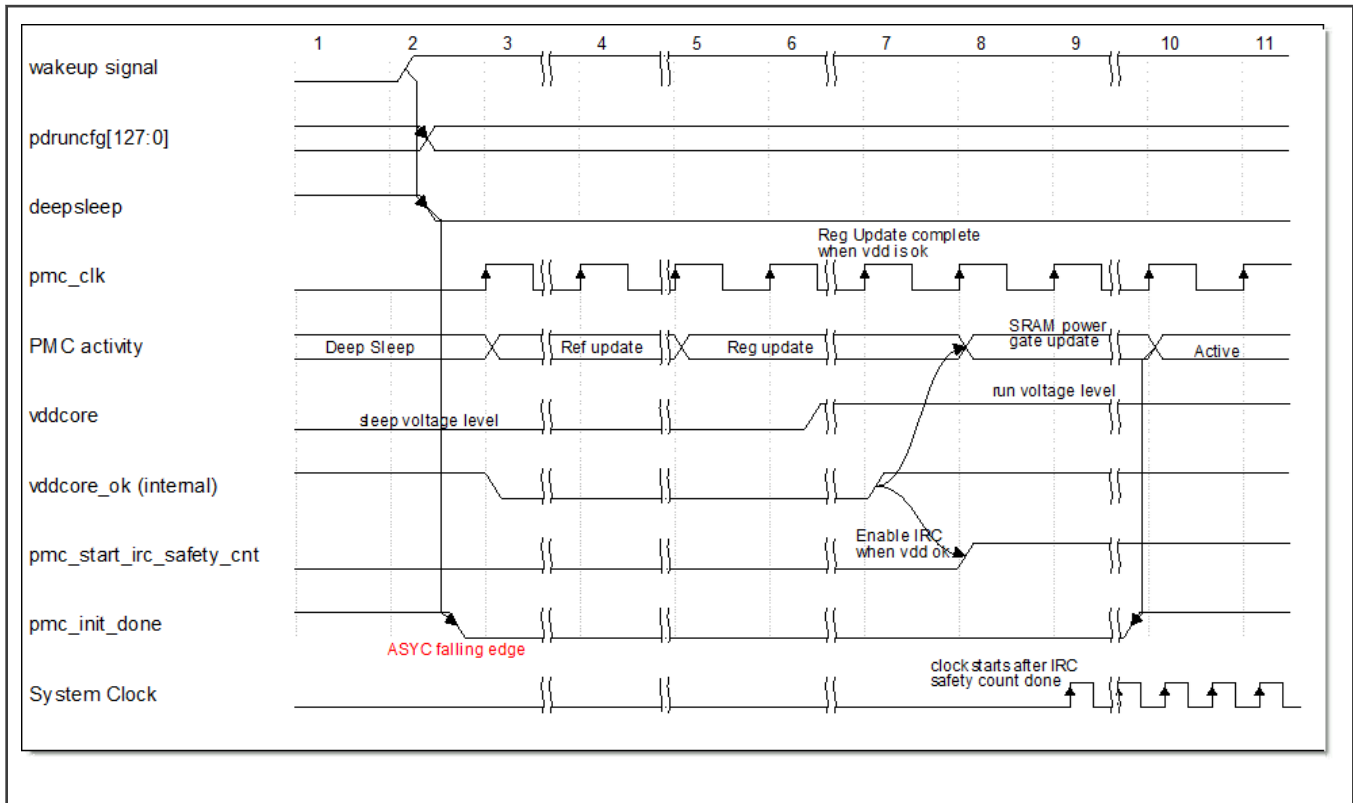
8.4.1.3.2 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled in the call to the power API function.

8.4.1.3.3 Wakeup from Deep Sleep

The chip can wake up from deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in PINT_SEL registers (See GPIO). Each pin interrupt must also be enabled in the NVIC.
- Using a watchdog interrupt or RTC interrupt, enabled during the reduced power mode via the power API. Each interrupt must also be enabled in the NVIC.
- Using a reset from the RESET pin or WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Each interrupt must also be enabled in the NVIC.
- GPIO interrupts. The interrupts must also be enabled in the NVIC.
- RTC alarm signal or wake-up signal (See RTC). Interrupts must also be enabled in the NVIC.



8.4.1.4 Deep Powerdown Mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC.

During deep power-down mode, the contents of the SRAM and registers (other than those in the RTC) are not retained. All functional pins are tri-stated in deep power-down mode as long as chip power supplied externally.

The SYSCTL block is in the vddcore domain, so all register contents are lost when the chip enters deep powerdown or full deep powerdown mode. On rising edge of a wakeup source, the wakeup mode is entered and a POR is generated to the vddcore domain. On entry to deep powerdown mode, the DEEP_PD and PMIC_MODE* bit values from the PDSLEEPCFG0 register are latched into the always on domain to maintain deep pd mode and the state of the MODE output pins.

8.4.1.4.1 Enter Deep Powerdown

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC. The SRAM and registers (other than those in the RTC) are not retained. All functional pins are tri-stated in deep power-down mode as long as VDDIO power rails are supplied externally.

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the VDD_AO1V8 pin.

The following steps must be performed to enter deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the VAL bits in WAKE register (See RTC). This starts the high resolution timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the POWER_EnterDeepPowerDown API function.

8.4.1.4.2 Power configuration in deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the VDD_AO1V8 pin.

8.4.1.4.3 Wakeup from Deep Powerdown

Wake-up from deep power-down can be accomplished via the reset pin, PMIC_IRQ_N pin or the RTC. The chip goes through the entire reset process when the RTC times out.

- The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip boots.
- All registers will be in their reset state except those retained by VDD_AO1V8 supply like RTC.

8.4.1.4.4 Programming deep power-down mode using the RTC for wake-up:

The following steps must be performed to enter deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the RTC VAL register. This starts the high res timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the POWER_EnterDeepPowerDown API function.

8.4.1.5 Full Deep Powerdown Mode

External supplies are powered off except for VDD_AO1V8. Wakeup sources (RTC, PMIC_IRQ_N pin, RESETn pin) trigger a POR in the vdd1v8 and vddcore domains.

8.4.1.6 Peripheral configuration in reduced power modes

Table 49. Peripheral configuration in reduced power modes

Peripheral/Clock	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down
1m_lposc	Software configured	Software configured	Off
192M FRO	Software configured	Software configured	Off
Crystal oscillator	Software configured	Software configured	Off
RTC and RTC oscillator	Software configured	Software configured	Software configured NOTE 32khz_clkout signal output not available.
System PLL	Software configured	Software configured	Off
Audio PLL	Software configured	Software configured	Off
SRAM memory arrays	Software configured	Software configured	Off
SRAM periphery	Software configured	Software configured	Off
Boot ROM	On	Off	Off
Other digital peripherals	Software configured	Software configured	Off
ADC	Software configured	Software configured	Off

Table continues on the next page...

Table 49. Peripheral configuration in reduced power modes (continued)

Peripheral/Clock	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down
Analog Comparator	Software configured	Software configured	Off
MIPI	Software configured	Software configured	Off
DSP	Software configured	Software configured	Off
OTP	Software configured	Software configured	Off

8.4.1.7 Wake-up process

The chip always wakes up to the active mode. To wake up from the reduced power modes, one or more wake-up sources must be configured. PMIC_MODE pins are cleared by one of these wakeup sources

- Reset
- PMIC_IRQ_N
- RTC wakeup

NOTE

In the table below:

- The related interrupt must be enabled in the NVIC.
- Enable related function in STARTEN0 or STARTEN1 register.

Table 50. Wake-up sources for reduced power modes

Power mode	Wake-up source	Comment
Sleep	Any peripheral that can cause an interrupt in sleep mode	
	HWWAKE	Flexcomm Interfaces and DMIC subsystem activity.
Deep-sleep	Pin interrupts	
	Watchdog interrupt	Only WDT0 can generate a wake-up from deep-sleep mode.
	Watchdog reset	Only WDT0 can generate a chip reset.
	Reset pin	No configuration needed.
	RTC 1 Hz alarm timer	
	RTC_ALARM, RTC_WAKE	
	Micro-tick timer	The Micro-tick timer is specifically targeted for ultra-low power wake-up from deep-sleep mode.
	OS Event Timer	

Table continues on the next page...

Table 50. Wake-up sources for reduced power modes (continued)

Power mode	Wake-up source	Comment
	Flexcomm USART	Interrupt from USART in slave or 32 kHz mode.
	Flexcomm SPI	Interrupt from SPI in slave mode.
	Flexcomm I2C	Interrupt from I2C in slave mode.
	Flexcomm I2S	Interrupt from I2S in slave mode.
	I3C	Interrupt from I3C in slave mode.
	USB need clock	Interrupt from USB when activity is detected that requires a clock. See SYSCTL0[USB0CLKSTAT].
	DMA	See DMA controller for details of DMA-related interrupts.
	DMA controllers	
	DMIC	
	HWWAKE	Certain Flexcomm Interface and DMIC subsystem activity. See SYSCTL0[HWWAKE].
	FlexSPI	
	SDIO	
	HASH-AES	
	CASPER	
	PowerQuad	
	ADC	
	DSP	
	ACMP	
Deep power-down	RTC_ALARM, RTC_WAKE	Enable related function in the and STARTEN0 or STARTEN1 register.
	Reset pin	No configuration needed.

8.4.2 Always-On Domain

The Always-On domain is a low-power domain supplied by the VDD_AO1V8 pins. This domain contains the RTC, eight 32-bit general-purpose registers GPREG[7:0], supports the PMIC_MODE pins, and the RESETN. This domain contains a POR circuit that will reset the logic in this domain.

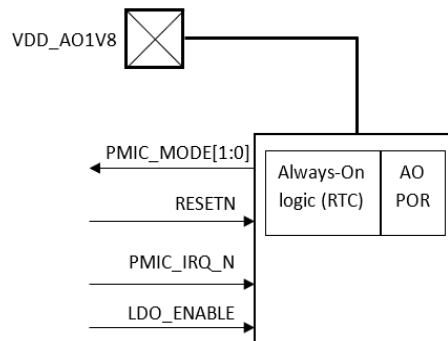


Figure 22. Always-On Domain

Pins in the Always-On domain:

- **PMIC_MODE[1:0]** – push-pull output pins. When using external PMIC, output pins that signal power mode transitions to PMIC, and which of 4 programmable power states to change to.
- **RESETN** – active-low input pin that resets the MCU.
- **PMIC_IRQ_N** – active-low input pin with wake-up and interrupt capability in any power mode. Intended for external PMIC, but can be used for any external wake-up/interrupt source.
- **LDO_ENABLE** – active-high input pin to configure PMC to use on-chip regulator to supply VDDCORE or external PMIC. When using PMIC to supply VDDCORE, LDO_ENABLE must be tied low, and this will tri-state the output of the on-chip regulator.

8.4.2.1 PMIC_MODE pins and register bits

The **PMIC_MODE** pins are an option when using external PMIC. They can eliminate the latency when communicating with PMIC, like through I2C, for quick power state transitions. The PMIC can be configured to use the **PMIC_MODE** pins to switch between 4 pre-configured power states. The state of the **PMIC_MODE** pins is controlled by the PMC and is configured by the **PMIC_MODE** bits. The state of these pins do not have any defined function and are completely controlled by software. The **PMIC_MODE** pins are controlled by software in Active mode, and by PMC hardware when the MCU transitions from Active into Deep Sleep or powerdown modes, and on wakeup from these power modes. In Active and Sleep modes, the **PMIC_MODE** pins are controlled with the **PDRUNCFG0[PMIC_MODE]** bits. For Deep Sleep and the powerdown modes, **PDSLEEPCFG0[PMIC_MODE]** bits are set before entering the low-power mode. When the PMC transitions into the low-power mode, it will change the **PMIC_MODE** pins to the state in the **PDSLEEPCFG0[PMIC_MODE]** bits. When waking from Deep Sleep mode, the **PMIC_MODE** pins will revert to the state in the **PDRUNCFG0[PMIC_MODE]** bits. When waking from the powerdown modes, or during initial power-up of the **VDD_AO1V8** supply, **PDRUNCFG0[PMIC_MODE]** bits and **PMIC_MODE** pins will all be logic low. The **PMICCFG** register is configured by firmware for the PMC to define if the **VDD1V8** and **VDDCORE** supplies are shutdown in 3 of the **PMIC_MODE** states. **PMIC_MODE = 00** is hardwired for **VDDCORE** and **VDD1V8** supplied, but the other 3 possible **PMIC_MODE**s are user defined. See **PMICCFG** register description for more details.

8.4.3 Temperature Sensor

To enable the temperature sensor, the sensor must be powered by clearing **PMC_TEMPSNS_PD** in **PDRUNCFG0/PDSLEEPCFG0** registers. Set the **APPLYCFG** bit to activate the **PDRUNCFG0** change to enable the temperature sensor.

NOTE

The temperature sensor is automatically disabled/powered down when the chip is in deep powerdown or full deep powerdown mode.

8.4.3.1 Using the Temperature Sensor

To configure system for measurement:

- Enable PMC Temperature Sensor (PDRUNCFG0[PMC_TEMPSNS_PD]=0), set the APPLYCFG bit to trigger PDRUNCFG0 change, wait for the ACTIVEFSM bit to go low, then another 40us for the temperature sensor to settle.
- Select ADC differential input channel connected to PMC Temperature Sensor.
- Configure ADC sampling time to (at least) 15us.

Measurement Procedure:

- Set TSENSOR[TSENSM]=3b'000, perform AD conversion, and record C1_000
- Set TSENSOR[TSENSM]=3b'001, perform AD conversion, and record C1_001
- Set TSENSOR[TSENSM]=3b'011, perform AD conversion, and record C1_011
- Set TSENSOR[TSENSM]=3b'010, perform AD conversion, and record C1_010
- Set TSENSOR[TSENSM]=3b'110, perform AD conversion, and record C1_110
- Set TSENSOR[TSENSM]=3b'111, perform AD conversion, and record C1_111
- Set TSENSOR[TSENSM]=3b'101, perform AD conversion, and record C1_101
- Set TSENSOR[TSENSM]=3b'100, perform AD conversion, and record CM_100
- Set TSENSOR[TSENSM]=3b'101, perform AD conversion, and record C2_101
- Set TSENSOR[TSENSM]=3b'111, perform AD conversion, and record C2_111
- Set TSENSOR[TSENSM]=3b'110, perform AD conversion, and record C2_110
- Set TSENSOR[TSENSM]=3b'010, perform AD conversion, and record C2_010
- Set TSENSOR[TSENSM]=3b'011, perform AD conversion, and record C2_011
- Set TSENSOR[TSENSM]=3b'001, perform AD conversion, and record C2_001
- Set TSENSOR[TSENSM]=3b'000, perform AD conversion, and record C2_000

Calculate die-temperature using the formula:

$$T[\text{in } ^\circ\text{C}] = 370.98 * (\text{CM\_TEMP} / \text{CM\_VREF}) - 273.15$$

- TSENS_CAL is an 8-bit signed calibration constant retrieved from the OTP fuse word index 0d93.
- $\text{CM\_VREF} = \text{CM\_CTAT} + (953.36 + \text{TSENS\_CAL}) * \text{CM\_TEMP} * 2^{-11}$
- $\text{CM\_CTAT} = (2 * \text{C1\_001} - \text{C1\_011} + 2 * \text{C2\_001} - \text{C2\_011} + 2 * \text{C1\_101} - \text{C1\_111} + 2 * \text{C2\_101} - \text{C2\_111}) * 2^{-2}$
- $\text{CM\_TEMP} = (2 * \text{C1\_000} - \text{C1\_010} + 2 * \text{C2\_000} - \text{C2\_010} + 4 * \text{CM\_100} - \text{C1\_110} - \text{C2\_110}) * 2^{-2}$

NOTE

Multiplications and divisions by 2 may be performed through simple shift-register operations. Refer to ADC chapter for information on how to configure the ADC HP bandgap must be enabled and Analog Buffer disabled whenever PMC Temperature Sensor is used.

8.4.4 Vddcore LVD

When using an external PMIC, on initial power up and on wakeup from deep sleep, the PMC logic uses the LVD status to determine when VDDCORE is high enough to allow the MCU to operate. There is no other handshake with the PMIC to indicate to the PMC that VDDCORE is OK for operation. So on wakeup from deep sleep, the MODE outputs change from the PDSLEEPCFG0[PMIC_MODE] state to the PDRUNCFG0[PMIC_MODE] state and the PMIC should respond by raising VDDCORE from a level less than LVD to a level greater than LVD. The PMC state machine will wait for the core LVD status to go low to indicate that VDDCORE > LVDCORECTRL.LVDCORELVL + 20mV, then the PMC will allow the MCU to enter active mode.

On initial power up, the LVDCORECTRL.LVDCORELVL is set to 0xC until the Boot ROM initializes it to 0x8.

LVDCORECTRL.LVDCORELVL sets the LVD trip point between 0.72V and 0.945V falling trip point. The LVD has hysteresis, so

once tripped by falling VDDCORE, VDDCORE must then rise 20mV higher than the selected trip point for the LVD to go back to inactive.

For example, when the LVD trip is set to the minimum and the application is using an external PMIC, the VDDCORE must rise above 0.74V for the MCU to start/wake-up. This is $0.72 + 20\text{mV}$ hysteresis when set to the minimum VDDCORE trip level (rising trip level). The LVD will continue to indicate VDDCORE is ok until VDDCORE drops below 0.72V (falling trip level). The current state of the LVD output can be read from STATUS.LVDCORE. Anytime the LVD trips (VDDCORE < LVDCORECTRL.LVDCORELVL) FLAGS.LVDCORE is set. The flag bit remains set until software clears it by writing a 1 to FLAGS.LVDCORE, i.e. the bit is “write 1 to clear”. Please refer to the data sheet for specifications pertaining to the LVD.

The LVD is active all the time, except deep power down mode when VDDCORE is off. The LVD's power mode can be low power or high power and is controlled by LVDCORE_LP bits in PDRUNCFG0 and PDSLEEPCFG0 registers, 1 => low power mode => slower reaction time. In LP mode it only uses the low power references, to enable the rest of the PMC to go into LP mode.

The LVD can trigger a reset, interrupt, or a wakeup depending on the setting of CTRL.LVDCORERE (LVD VDDCORE reset enable), CTRL.LVDCOREIE (LVD VDDCORE interrupt enable), ARM NVIC interrupt enable bit (ISER), and SYSTL0_STARTEN1.PMIC bit.

- If LVDCORERE=1 then any time VDDCORE drops below the trip level, except in deep power down mode, the MCU will reset. This is the default configuration.
- If LVDCORERE=0 and LVDCOREIE=1 and NVIC interrupt is enabled, then anytime VDDCORE drops below the trip point, the PMC will trigger an interrupt. The PMC interrupt output remains active until FLAGS.LVDCORE is cleared by writing a 1.
- If LVDCORERE=0 and LVDCOREIE=1 and NVIC interrupt is enabled and SYSTL0_STARTEN1[PMIC]=1, then when VDDCORE < LVD trip, the part will wake-up from sleep or deep sleep mode and respond to the PMC interrupt.

8.4.5 Optimizing power with PMC

Several circuits in the PMC can be powered down when unused. The following functions are controlled by these bits in the PDRUNCFG0/PDSLEEPCFG0 registers:

- PMC temperature sensor controlled by PMC_TEMPSNS_PD. This sensor can be measured by the ADC using the SYSTL0.TEMPSNSORCTL register.
- Body biasing generators controlled by RBBSRAM_PD, FBB_PD, and RBB_PD.
- High-Voltage Detectors controlled by HVDCORE_PD and HVD1V8_PD

Also, some PMC circuits can be switched to a Low-Power (LP) mode. The intent is that these circuits can be in LP mode when the MCU is in Deep Sleep or Deep Power Down modes. Otherwise, these circuits need to be in their High-Power (HP) mode. In the PDRUNCFG0/PDSLEEPCFG0 registers, the POR and LVD power modes are controlled by LVDCORE_LP and PORCORE_LP bits. The LVD and POR react slower to voltage changes when in LP mode, vs HP mode. And the VDDCOREREG_LP bit controls the internal regulator for VDDCORE. In LP mode it has lower leakage at the expense of more ripple.

The PMC includes some references used by other circuits, which can also be in LP mode using the PMCREF_LP bit. The PMC will force the references into HP mode (and ignore the PMCREF_LP setting) if needed, specifically when the following circuits are powered or in HP mode: HVD1V8_PD=0, PORCORE_LP=0, LVDCORE_LP=0, HVDCORE_PD=0, FBB_PD=0, or PMC_TEMPSNS_PD=0. Therefore, the PMCREF_LP bit can be set for LP mode most of the time. The cases when PMCREF_LP should be cleared and in HP mode are when either PLL is enabled or the FRO is enabled. Also, it might be needed if the ADC or the Comparators need the high power reference voltages or currents while the PMC is in LP mode. When the PMC references are in LP mode, their outputs are turned off completely, and any analog block depending on them will not be functional.

8.4.6 Body Bias

Body Biasing adjusts the transistor threshold voltage V_t . Three different body bias modes are supported:

- Normal Body Bias NBB should be used at very low frequencies < 60MHz.
- Forward Body Bias FBB lowers the V_t , which allows the device to turn quickly, used for high performance.
- Reverse Body Bias RBB raises required V_t increasing the reaction time t_{ON} but lowering the leakage current.

- The variation of three bits in the PDRUNCFG0/PDSLEEPCFG0 register determines the body bias mode. The power library API complies with the specific process required when switching between body bias modes. FBB is the default mode out of RESET and should be the return state after waking from Deep Sleep.

8.5 Memory Map and register definition

This section includes the Power Management module memory map and detailed descriptions of registers.

8.5.1 Power Management register descriptions

PMU

8.5.1.1 PMU memory map

PMC0 base address: 4013_5000h

Offset	Register	Width (In bits)	Access	Reset value
4h	Status (STATUS)	32	R	0000_0000h
8h	Wakeup, Interrupt, Reset Flags (FLAGS)	32	RW	0017_0000h
Ch	PMC control register (CTRL)	32	RW	0020_0000h
10h	PMC controls used during run mode (RUNCTRL)	32	RW	0000_0026h
14h	PMC controls used during deep sleep mode (SLEEPCTRL)	32	RW	0000_0026h
18h	PMC Active vddcore LVD monitor trip adjust (LVDCORECTRL)	32	RW	0000_000Ah
24h	PMC Automatic wakeup from deepsleep mode (AUTOWKUP)	32	RW	0000_0000h
28h	PMIC Power Mode Select Control Configuration (PMICCFG)	32	RW	0000_0073h
2Ch	PMC GPIO VDDIO Range Selection Control (PADVRANGE)	32	RW	0000_0000h
30h	PMC Memory sequencer Control (MEMSEQCTRL)	32	RW	0000_003Fh
60h	PMC Temperature Sensor Control (TSENSOR)	32	RW	0000_0000h

8.5.1.2 Status (STATUS)

Offset

Register	Offset
STATUS	4h

Function

Current values of these items, not latched, but synchronized to PMC clock. See the FLAGS register for latched versions for sources that enabled to generate resets or interrupts.

NOTE

The reset value depends on the cause of the most recent reset. This register is handled by the SDK power library.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved				Reserved	Reserved				ACTIV EF...		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29 —	Reserved
28 —	Reserved
27 —	Reserved
26-25 —	Reserved
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 —	Reserved
20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
19-11 —	Reserved
10-8 —	Reserved
7 —	Reserved
6-1 —	Reserved
0 ACTIVEFSM	General sequencer and finite state machine status 0b - All PMC finite state machines are idle. OK to set APPLYCFG to trigger the PMC state machines. 1b - One or more PMC finite state machines are active, do not set APPLYCFG or write to any PDRUNCFG or CTRL register values that are used by the PMC state machines.

8.5.1.3 Wakeup, Interrupt, Reset Flags (FLAGS)

Offset

Register	Offset
FLAGS	8h

Function

It stores information about events such as resets, interrupts, deep powerdown mode wakeup and so on. The bits are set by hardware and cleared by writing 1.

NOTE

This register is handled by the SDK power library.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DEEP PDF	RESE TNP...	INTNP ADF	AUTO WKF	RTCF	Reserved			HVD1 V8F	Reserv ed	HVDC OREF	Reserv ed	LVDC OREF	Reserv ed	PORA O18F	POR1 V8F	PORC OREF
W	W1C	W1C	W1C	W1C	W1C				W1C		W1C		W1C		W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserved																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31 DEEPPDF	Deep powerdown wakeup flag 0b - No deep powerdown wakeup since last time flag was cleared. 1b - Deep powerdown was entered since the last time this flag was cleared. Write 1 to clear
30 RESETNPADF	Reset pad flag 0b - No reset detected since last time this flag was cleared. 1b - Reset pad wakeup caused a wakeup or reset event since the last time this bit was cleared. Write 1 to clear.
29 INTNPADF	PMIC_IRQ_N Interrupt pin flag 0b - No interrupt detected since flag last cleared. 1b - Pad interrupt caused a wakeup or interrupt event since the last time this flag was cleared. Write 1 to clear.
28 AUTOWKF	PMC Auto Wakeup Interrupt flag 0b - No PMC Auto Wakeup Interrupt detected since last time cleared. 1b - PMC Auto wakeup caused a deep sleep wakeup and interrupt. Write 1 to clear.
27 RTCF	RTC Wakeup from deep powerdown mode flag 0b - No RTC wakeup detected since last time flag was cleared. 1b - RTC wakeup caused a deep powerdown wakeup. Write 1 to clear.
26-25 —	Reserved
24 HVD1V8F	vdd1v8 High-Voltage Detector Flag This flag is set when vdd1v8 > the trip voltage of the vdd1v8 HVD monitor. The monitor must be powered up by PDRUNCFG. Flag can't be cleared if vdd1v8 is still > trip point.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - vdd1v8 HVD has not tripped since last clear 1b - vdd1v8 HVD tripped since last time this bit was cleared. Write 1 to clear
23 —	Reserved
22 HVDCOREF	vddcore High-Voltage Detector Flag This flag is set when vddcore > the trip voltage of the vddcore HVD monitor. The monitor must be powered up by PDRUNCFG. Flag will not be cleared if vddcore is still > trip point. 0b - vddcore HVD has not tripped since last clear 1b - vddcore HVD tripped since last time this bit was cleared. Write 1 to clear
21 —	Reserved
20 LVDCOREF	vddcore Low-Voltage Detector Flag This flag is set when vddcore < the trip voltage of the vddcore LVD monitor. Flag cannot be cleared if vddcore is still < trip point. 0b - vddcore LVD has not tripped since last clear 1b - vddcore LVD tripped since last time this bit was cleared. Write 1 to clear
19 —	Reserved
18 PORA018F	VDD_AO1V8 power on reset flag 0b - No VDD_AO1V8 power on event detected since last cleared. 1b - VDD_AO1V8 power on detect caused a reset. Write 1 to clear.
17 POR1V8F	vdd1v8 power on reset flag NOTE This bit must be cleared prior to entering deep sleep / deep powerdown / full deep powerdown mode. If it is not cleared, the MODE output pins will be cleared shortly after entering the reduced power mode, but a wakeup will not be triggered. 0b - No vdd1v8 power on event detected since last cleared. 1b - vdd1v8 power on detect caused a reset or deep power down wakeup. Write 1 to clear.
16 PORCOREF	vddcore POR Flag 0b - vddcore POR was not tripped since the last cleared. 1b - POR triggered by the vddcore POR monitor. Write 1 to clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-0 —	Reserved

8.5.1.4 PMC control register (CTRL)

Offset

Register	Offset
CTRL	Ch

Function

Since the register is in the vddcore domain the bits have no effect in deep powerdown mode or on wakeup from deep powerdown mode. They go to their reset value and the RBB_PD and FBB_PD bits in PDRUNCFG0 are set to 1 (their reset value) on wakeup from deep powerdown (See SYSCTL).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		INTRP	AUTO	Reserved		HVD1	HVD1	HVDC	HVDC	LVDC	LVDC	Reserv	OTPS		Reserved
W			AD...	WKEN			V8RE	V8IE	ORE...	ORE...	ORE...	ORE...		WREN		
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Reserved				Reserved		CLKDI	
W									Reserved				BUFE		VEN	APPLY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29 INTRPADEN	PMIC_IRQ_N enable Interrupt pad deep powerdown and deep sleep wake up & interrupt enable 0b - Interrupt pad low has no effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Interrupt pad low triggers an interrupt and deep sleep wakeup or deep powerdown wakeup event.
28 AUTOWKEN	PMC automatic wakeup enable and interrupt enable 0b - Auto wakeup interrupt and counter disabled 1b - Auto wakeup interrupt generated when PMC sequencer finishes and AUTOWAKE counter = 0 after entering deep sleep mode (but not deep powerdown mode). Interrupt will wake up the M33.
27-26 —	Reserved
25 HVD1V8RE	vdd1v8 High-Voltage Detector Reset Enable 0b - vdd1v8 HVD reset disabled 1b - vdd1v8 HVD causes reset
24 HVD1V8IE	vdd1v8 High-Voltage Detector Interrupt Enable 0b - vdd1v8 HVD interrupt disabled 1b - vdd1v8 HVD causes interrupt and wakeup from deep sleep or deep power down mode.
23 HVDCORERE	vddcore High-Voltage Detector Reset Enable 0b - vddcore HVD reset disabled 1b - vddcore HVD causes reset
22 HVDCOREIE	vddcore High-Voltage Detector Interrupt Enable 0b - vddcore HVD interrupt disabled 1b - vddcore HVD causes interrupt and wakeup from deep sleep.
21 LVDCORERE	vddcore Low-Voltage Detector Reset Enable 0b - vddcore LVD reset disabled 1b - vddcore LVD causes reset
20 LVDCOREIE	vddcore Low-Voltage Detector Interrupt Enable 0b - vddcore LVD interrupt disabled 1b - vddcore LVD causes interrupt and wakeup from deep sleep.
19 —	Reserved
18 OTPSWREN	OTP Switch RBB enable
17-16	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15-9 —	Reserved
8-5 —	Reserved
4 BUFEN	Enable analog buffer for references or ATX2 0b - Disabled 1b - Enabled
3-2 —	Reserved
1 CLKDIVEN	Internal clock divider enable When enabled, this bit increases the PMC register access delay (reads or writes). Likewise, it has no effect on power or timing during deep sleep or deep power down since it is disabled automatically in those modes, including mode entry and exit. 0b - 16MHz clock selected 1b - 4MHz clock selected
0 APPLYCFG	Apply updated PMC PDRUNCFG bits Apply updated PMC PDRUNCFG bits (SRAM power gates, RBB, FBB, LVD, and HVD control bits) and/or RUNCTRL setting. Please refer PDRUNCFG description (See SYSCTL). 0b - Always reads 0. Write 0 has no effect 1b - Write 1 = initiate update sequencing of PMC state machines

8.5.1.5 PMC controls used during run mode (RUNCTRL)

Offset

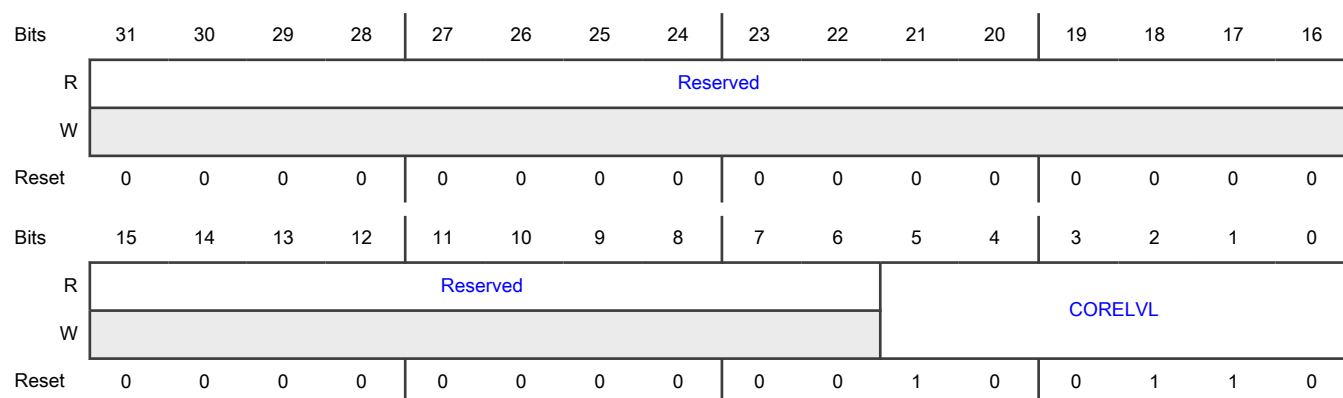
Register	Offset
RUNCTRL	10h

Function

Writes to this register do NOT affect the on-chip regulator output voltage UNTIL the APPLYCFG bit is set or until a wakeup event occurs.

NOTE

This register is handled by the SDK power library.

Diagram**Fields**

Field	Function
31-6 —	Reserved
5-0 CORELVL	Vddcore voltage value when using on-chip regulator and SYSCCTL is in run mode. NOTE This voltage level should be set higher than LVD threshold level in LVDCORELVL. 00_0001b - 0.6V 01_0000b - 0.7V 01_0011b - 0.8V 10_0110b - 1.0V 11_0010b - 1.138V

8.5.1.6 PMC controls used during deep sleep mode (SLEEPCTRL)**Offset**

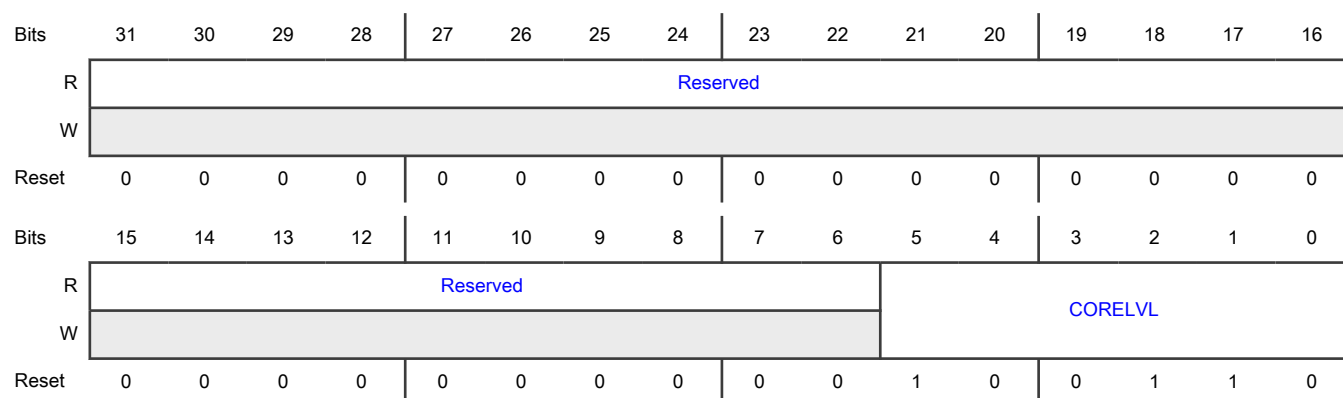
Register	Offset
SLEEPCTRL	14h

Function

This register is identical to RUNCTRL, except it is used when the M33 is in deep sleep mode. This allows the SYSCCTL block to change the voltage levels when the chip goes from run to deep sleep and when it wakes up and returns to run mode.

NOTE

This register is handled by the SDK power library.

Diagram**Fields**

Field	Function
31-6 —	Reserved
5-0 CORELVL	Vddcore voltage value when using on-chip regulator and SYSCTL is in sleep mode. 0x32 is the maximum value that can be written. If a higher write value is written, then 0x32 is written instead. 00_0000b - 0.595833V 10_0110b - 1.007498V = 0.595833 + 0x26 10.8333mV 11_0010b - 1.138V

8.5.1.7 PMC Active vddcore LVD monitor trip adjust (LVDCORECTRL)**Offset**

Register	Offset
LVDCORECTRL	18h

Function

Active vddcore LVD monitor trip adjust.

NOTE

This register is handled by the SDK power library.

Diagram

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

Fields

Field	Function
31-4 —	Reserved
3-0 LVDCORELVL	Vddcore LVD falling trip voltage, in steps of 15mV 0000b - 0.720V 0111b - 0.825V = 0.720V + 7 x 15mV 1111b - 0.945V

8.5.1.8 PMC Automatic wakeup from deepsleep mode (AUTOWKUP)**Offset**

Register	Offset
AUTOWKUP	24h

Function

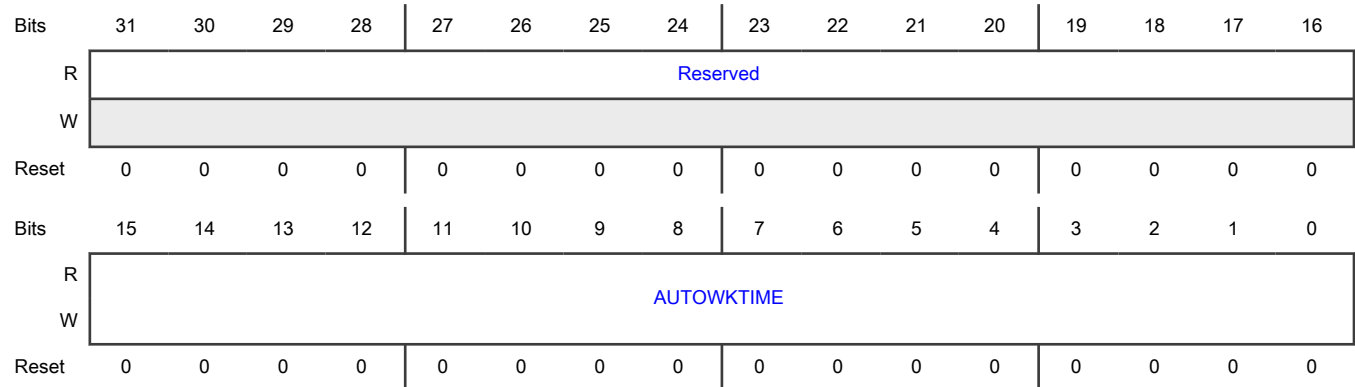
When this feature is enabled by CTRL.AUTOWKEN, the PMC will generate a wakeup and interrupt to the M33 to exit deep sleep mode when the PMC mode sequencer has completed entry into deep sleep mode and the optional delay timer has counted down to zero. The wakeup delay timer is a preloaded down counter implemented in 1.8V domain logic and clocked by the PMC 16MHz oscillator. The value in this register is preloaded into the counter when deep sleep mode is entered. When the state machine reaches "deep sleep idle", the down counter starts and when the count reaches zero the pmc_intr goes active. This is implemented in the vdd1v8 domain which may be turned off during deep powerdown mode, so this is only useful in deep sleep modes. This feature allows the system to stop clock to enter/exit FBB or RBB then wakeup while maintaining the BB mode. Use in conjunction with the PDWAKECFG register in SYSCTL.

NOTE

This register is handled by the SDK power library.

NOTE

On POR or Deep Power Down/Full Deep Power Down, this value is set to 0xC until the Boot ROM initializes it to 0x8.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 AUTOWKTIME	Auto wake up delay timer. Added delay after sequencer delay delay time = <value>/16MHz 0000_1111_1111_1111b - Delay time = 0x0FFF/16MHz (example)

8.5.1.9 PMIC Power Mode Select Control Configuration (PMICCFG)**Offset**

Register	Offset
PMICCFG	28h

Function

It lets PMC know when vddcore or vdd1v8 will power off/onConfigures the PMC to respond to changes on PDRUNCFG0[PMIC_MODE] select pin values. Defines run (all supplies on), deep powerdown (vddcore off), and full deep powerdown (vdd1v8 and vddcore off). Used in isolation logic for signals from vddcore and vdd1v8 domain going into the VDD_AO1V8 domain.

NOTE

VDD1V8M0 and VDDCOREM0 is expected to be 1 during active mode otherwise a POR will be triggered.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								VDD1 V8M3	VDD1 V8M2	VDD1 V8M1	VDD1 V8M0	VDDC ORE...	VDDC ORE...	VDDC ORE...	VDDC ORE...
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1

Fields

Field	Function
31-8 —	Reserved
7 VDD1V8M3	vdd1v8 state in PMIC mode 3 0b - Off 1b - Powered
6 VDD1V8M2	vdd1v8 state in PMIC mode 2 0b - Off 1b - Powered
5 VDD1V8M1	vdd1v8 state in PMIC mode 1 0b - Off 1b - Powered
4 VDD1V8M0	vdd1v8 state in PMIC mode 0 0b - Off 1b - Powered
3 VDDCOREM3	vddcore state in PMIC mode 3 Note that this must be 0 if VDD1V8M3 = 0 0b - Off 1b - Powered
2 VDDCOREM2	vddcore state in PMIC mode 2 Note that this must be 0 if VDD1V8M2 = 0 0b - Off 1b - Powered

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 VDDCOREM1	vddcore state in PMIC mode 1 Note that this must be 0 if VDD1V8M1 = 0 0b - Off 1b - Powered
0 VDDCOREM0	vddcore state in PMIC mode 0 Note that this must be 0 if VDD1V8M0 = 0 0b - Off 1b - Powered

8.5.1.10 PMC GPIO VDDIO Range Selection Control (PADVRANGE)

Offset

Register	Offset
PADVRANGE	2Ch

Function

This register controls the FSGPIOs bias voltage filler cells in the pad ring. The filler cell block has a detector to sense the vdd value, 1.8V or 3.3V range, but this consumes current. To save power, the filler cell can be configured to the 1.8V or the 3.3V range, and the detector is disabled. At reset it configures the filler cells into detect mode. The range bits for VDDIO_0, VDDIO_1, VDDIO_2, VDDIO_3, VDDIO_4, drive vdd1v8 domain outputs that connect to the corresponding range bits of the pad bias cells.

NOTE

VDDIO_3 is the only supply that can support 3.3V.

NOTE

This register is handled by the SDK power library. If not using the SDK power library, it is strongly recommended that the default values for VDDIO_RANGE0/1/2/4 are changed to 01 in order to reduce current consumption.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				VDDIO_4RANG		VDDIO_3RANG		VDDIO_2RANG		VDDIO_1RANG		VDDIO_0RANG			
W					E		E		E		E		E			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-10 —	Reserved
9-8 VDDIO_4RANG E	VDDIO4RANGE 00b - 1.71 - 1.98V. Continuous mode. Consumes static current to detect VDDIO_4 level. To reduce power consumption, change this value to 01. 01b - 1.71 - 1.98V, vdde detector off 10b - Not allowed 11b - Not allowed (hardware translates to 00 = continuous mode)
7-6 VDDIO_3RANG E	VDDIO3RANGE 00b - 1.71 - 3.6V. Continuous mode. Consumes static current to detect VDDIO_3 level 01b - 1.71 - 1.98V, vdde detector off 10b - 3.00 - 3.6V, vdde detector off 11b - Not allowed (hardware translates to 00 = continuous mode)
5-4 VDDIO_2RANG E	VDDIO2RANGE 00b - 1.71 - 1.98V. Continuous mode. Consumes static current to detect VDDIO_2 level. To reduce power consumption, change this value to 01. 01b - 1.71 - 1.98V, vdde detector off 10b - Not allowed 11b - Not allowed (hardware translates to 00 = continuous mode)
3-2 VDDIO_1RANG E	VDDIO1RANGE It is recommended that the user change this value to 01 to reduce power consumption. 00b - 1.71 - 1.98V. Continuous mode. Consumes static current to detect VDDIO_1 level. 01b - 1.71 - 1.98V, vdde detector off 10b - Not allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Not allowed (hardware translates to 00 = continuous mode)
1-0 VDDIO_0RANGE	VDDIO_0RANGE 00b - 1.71 - 1.98V. Consumes static current to detect VDDIO_0 level. To reduce power consumption, change this value to 01. 01b - 1.71 - 1.98V, vddio detector off 10b - Not allowed 11b - Not allowed (hardware translates to 10)

8.5.1.11 PMC Memory sequencer Control (MEMSEQCTRL)

Offset

Register	Offset
MEMSEQCTRL	30h

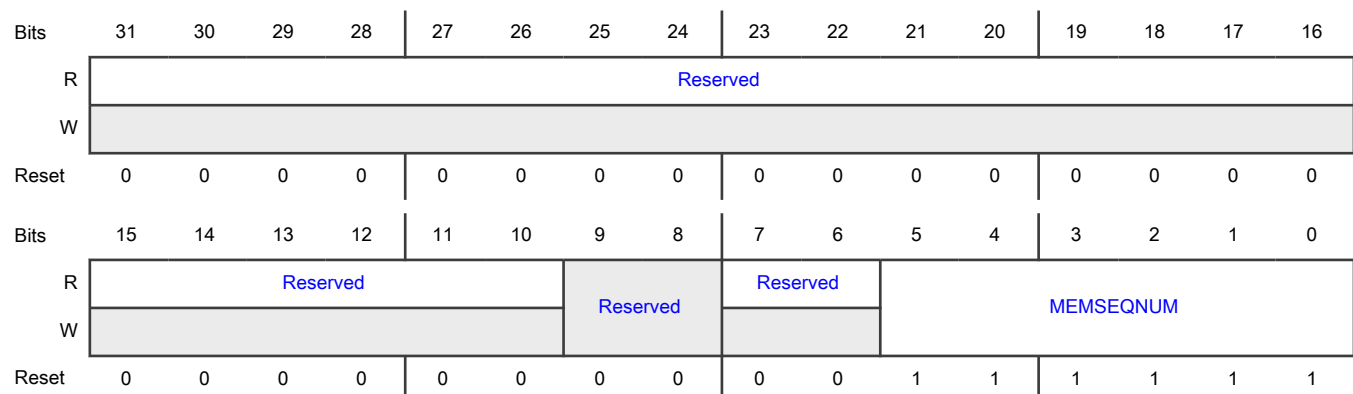
Function

This register defines the number of memory power gate switches that are turned on at a time. The memory sequencer state machine starts turning on the memories in PDRUNCFG1, starting at bit 0, then the memories in PDRUNCFG2/3 starting at partition 0. This is so the smaller ones get connected first, thus adding to the VDDCORE decoupling capacitance as more are turned on so larger memories later cause less delta V.

NOTE

The value written to this register should not be changed. This register is handled by the SDK power library. The user should not modify the power library source code. Changes to the power library source code can cause application failure. NXP is not responsible for any change to the power library code and is not obligated to provide support.

Diagram



Fields

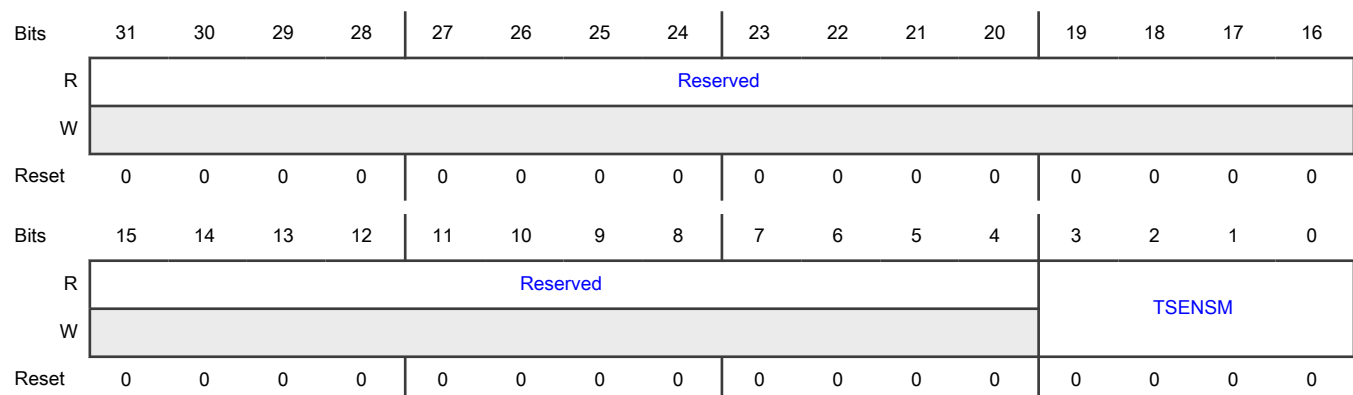
Field	Function
31-10 —	Reserved
9-8 —	Reserved
7-6 —	Reserved
5-0 MEMSEQNUM	Number of memories to turn on/off at a time. 00_0000b - For main system SRAM partitions, 1st array power then periphery power (400ns worst case delay) 00_0001b - Turn on 1st memory partition at a time, periphery and array power switches at the same time. 00_0101b - Turn on 5th memory partitions in parallel, periphery and array. 11_1111b - All memories are switched on/off at the same time

8.5.1.12 PMC Temperature Sensor Control (TSENSOR)

Offset

Register	Offset
TSENSOR	60h

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 TSENSM	Temperature sensor mode select Selects a circuit configuration according to the procedure described in Using the Temperature Sensor . Note that all circuit configurations will be selected, in sequence, during the temperature measurement procedure, as means to dynamically cancel various internal error sources. Please refer SYSCTL[TEMPSENSORCTRL] to select PMC temperature sensor.

Chapter 9

Processor Configurations

9.1 ARM Cortex-M33 Details

ARM Limited publishes a Generic User Guide for the Cortex®-M33. This is available on their website at: arm.com

9.1.1 Cortex-M33 implementation options

The Cortex™-M33 CPU provides a number of implementation options explained below.

Table 51. Cortex-M33 options

Feature	Description
FPU	A Floating Point Unit is included. The FPU supports single-precision floating-point computation functionality in compliance with the ANSI/IEEE Standard 754-2008. The FPU provides add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also performs a variety of conversions between fixed-point, floating-point, and integer data formats.
DSP	Digital Signal Processing extension instructions are included.
SECEXT	ARMv8-M security extensions are included.
CPIF	A coprocessor interface is included.
MPU	Both a secure and non-secure Memory Protection Unit are included, each with 8 regions. The MPU provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis.
SAU	A Security Attribution Unit with 8 regions is included.
NUMIRQ	79 interrupt slots are implemented for the Cortex-M33. Not all interrupts are available on all part numbers.
IRQLVL	3 interrupt priority bits are implemented on the Cortex-M33, providing 8 priority levels.
DBGLVL	Four watchpoint and eight breakpoint comparators.
ITM	Instrumentation Trace Macrocell (ITM) as well as DWT triggers and counters are included.
ETM	The Embedded Trace Macrocell is included.
MTB	The Micro Trace Buffer is not included
WIC	The Wake-up Interrupt Controller is not included. Similar functionality is provided by other mechanisms in the device.
CTI	The Cross Trigger Interface unit is included.

NOTE

NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M33.

9.2 Fusion DSP Details

Option	Value
Fusion F1 FP	Selected
Fusion F1 Reduced MAC Latency	Selected
Fusion F1 AVS	Selected
Fusion F1 16-bit Quad MAC DSP	Selected
Size of L0 Loop Buffer (in bytes)	256
Memory Protection/MMU	Region protection
16-bit MAC with 40-bit Accumulator	Selected
Number of AR register for call windows	32
Byte ordering (endianness)	Little Endian
Generate exception on unaligned load/store address	Take Exception
L32R Hardware support option	Normal L32R
Pipeline length	7
GPIO32: 32-bit GPIO interface	Selected
Debug	Selected
Debug: Instruction address breakpoint registers	2
Debug: Data address breakpoint registers	2
Debug: On-chip Debug (OCD)	Selected
Debug: External Debug Interrupt	Selected
Debug: Number of Performance Counters	2
Trace Port (address trace and pipeline status)	Selected
Add data trace	Selected
Interrupt count	32
Interrupt: Count of interrupt priority levels	4
Interrupt: Timer count	2
Interrupt: EXCM priority level (highest priority of efficiently C-callable handlers)	2
Interrupt: Debug interrupt level	4

Chapter 10

Fusion DSP

10.1 Overview

This section provides an introduction to the Fusion DSP.

NOTE

This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations.

10.1.1 Features

- Cadence Xtensa Fusion F1 Audio DSP processor, running at frequencies of up to 275 MHz.
- Hardware Floating Point Unit.
- Serial Wire Debug (shared with Cortex-M33 Control Domain CPU).

10.2 Memory Map

The general features of the Fusion DSP memory map are discussed in the following sections.

The Fusion DSP includes an AHB bus master interface. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSTL0[AHBMATRIXPRIOR] register.

10.2.1 RAM

The Fusion F1 can access the entire main system RAM, and the TCM memories are physically shared system memories.

10.2.1.1 Tightly Coupled Memories (TCMs)

The Fusion F1 uses the Shared System RAM as TCMs for both code (IRAM0 / IRAM1) and data (DRAM0 / DRAM1). This shared memory is divided in several partitions allowing parallel accesses to different partitions. Therefore IRAM0 and DRAM0 are configured to access different partitions of the shared RAM, and IRAM1 / DRAM1 are configured to access other different partitions. And the shared RAM can be accessed by other masters too (Cortex M33, DMA controllers, etc).

NOTE

Under no circumstance should the DSP code ("IRAM"/"TCM Code") and data ("DRAM"/"TCM Data") regions be located in the same shared ram partition. Doing so will cause a permanent lock-up.

10.2.1.2 Main SRAM

The Fusion F1 has access to the entire main SRAM space, unless security is enabled and parts are marked for secure-only access.

10.2.2 AHB and APB peripherals

The Fusion F1 accesses system peripherals via its AHB master interface using the same busing as the Cortex-M33. Access to some peripherals is not possible due to the AHB matrix connections. Available peripherals can be seen in [Block Diagrams](#), [System memory map](#) and [AHB Peripherals](#).

If system security is enabled, other peripherals may become unavailable since the Fusion F1 does not have modes similar to those supported by the Cortex-M33. System security can selectively lock out any non-secure master from any number of specific peripherals.

All APB peripheral are intrinsically accessible to the Fusion F1, but some may be made unavailable through security settings.

10.3 Reset behavior

The Fusion F1 is held in reset after a device reset. Before the Fusion F1 begins execution, the Cortex-M33 must load code into a memory accessible by the Fusion F1, configure (at a minimum) the Fusion F1 reset vector, and release the Fusion F1 from reset. API calls support various portions of this process.

The vector table base address for Fusion F1 is configured by Cortex-M33 using the mapped register `DSP_VECT_REMAP`. Please refer to the section for more details.

10.4 Security

The Fusion F1 does not include any mechanism for supporting system security of the type implemented by the Cortex-M33. So, if TrustZone is enabled and/or other system security features are enabled, the Fusion F1 will not have access to any secured memory regions or peripherals. System security can be very specific, controlled to the level of individual peripherals and individual memory partitions.

In some cases, peripherals are grouped on AHB matrix ports with the idea that all of those peripherals will likely not need to be accessed by the Fusion F1. If that is the case, the whole matrix port can be secured for simplicity.

In the case of clock control, reset control, and other system registers, each type of register is split into two parts. Because security resolution does extend all the way to individual registers, an attempt has been made to separate these system registers into groups that are likely to be secured (`CLKCTL0`, `RSTCTL0`, and `SYSCTL0`), and those that might need to be accessed by the Fusion F1 or other non-secure bus masters (`CLKCTL1`, `RSTCTL1`, and `SYSCTL1`).

- All groups can be secured, allowing access only to secure bus masters
- All can be left unsecured, allowing access to all bus masters
- Groups can be selectively secured to protect more critical system controls, while allowing access to other masters, including the Fusion F1.

10.5 Interrupts

The connection of interrupts to the Fusion DSP is shown in [Fusion DSP interrupt priority assignments](#). Most interrupts are user selectable using the input multiplexing feature described in [Fusion DSP interrupt input multiplexer](#). In addition to giving a choice of interrupts needed, selecting interrupts allows allocating interrupt priorities to fit the application. L1 interrupts are the lowest priority, L3 are the highest.

One of the Fusion DSP interrupt sources (from interrupt 5 through interrupt 31) can be selected as the Fusion DSP NMI source by configuring the Fusion DSP NMI source selection.

NOTE

The Fusion DSP interrupts 5 through 31 only accepts level interrupt, so modules that have self-clearing edge detection interrupts, like PINT, may not trigger the interrupt depending on module clock speed.

10.6 DMA

There are two DMA controllers that can be used in a number of ways. One of two main intended use cases is to allocate DMA0 to the CM33, and DMA1 to the Fusion F1. This case can apply to systems where there is no need to differentiate security between the CPUs or between different tasks, and simplifies keeping track of which CPU is using which DMA channels. It can also apply if non-secure code and bus masters share DMAC1 with the Fusion F1 to handle non-secure data.

10.7 Inter-CPU communications

The Fusion F1 and the CM33 can communicate with each other via various means. Specific hardware is provided to facilitate communication.

A way to send simple messages is available with the Message Unit peripheral, which provides dedicated Fusion F1 to CM33 and CM33 to Fusion F1 channels (see [Messaging Unit](#)). In addition, the Semaphore peripheral (see [Semaphore SEMA42](#)) can be used to communicate the allocation of resources.

The DSPWAKE interrupt may be used to wake up the CM33 when it is in reduced power mode, typically so that some service may be requested by the Fusion F1. The Fusion F1 can alternatively use SYSTL1[RXEVULSEGEN] to generate a pulse on the CM33 RXEV input to wake it up if it is stopped in a WFE instruction.

Chapter 11

AXI Switch

11.1 Overview

AXI is an AMBA switch for the GPU and LCDIF. Through the switch, the two controllers can access memories at the same time. It has three slave interfaces and ten master interfaces. All the ten NIC301 AMIB interfaces are AXI type. An AXI to AHB bridge from Arm called XHB400 is used to convert each AMIB interface to AHB.

11.2 Memory map and register definition

This section includes the AXI SWITCH memory map and detailed descriptions of all registers.

11.2.1 AXI SWITCH AMIB register descriptions

Port S0 AMIB registers are at address GPV0_BASE + 0x2000. GPV0_BASE is the base address of AXI SWITCH.

11.2.1.1 AXI_SWITCH_AMIB memory map

GCDC_SWITCH0.AXI_SWITCH_AMIB base address: 4028_2000h

Offset	Register	Width (In bits)	Access	Reset value
8h	Bus Matrix Issuing Functionality Modification. (FN_MOD_BM_ISS)	32	RW	0000_0000h
24h	Bypass Merge (FN_MOD2)	32	RW	0000_0000h
108h	Issuing Functionality Modification (FN_MOD)	32	RW	0000_0000h

11.2.1.2 Bus Matrix Issuing Functionality Modification. (FN_MOD_BM_ISS)

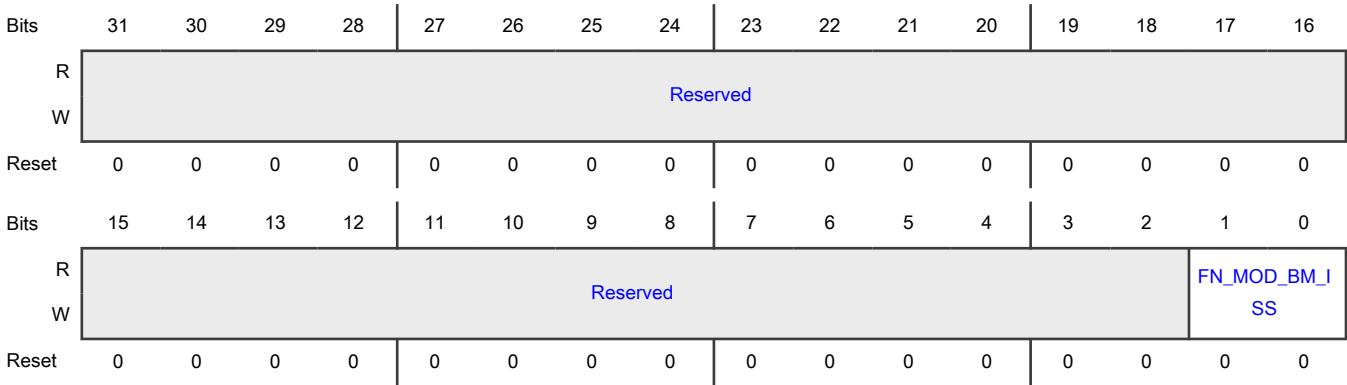
Offset

Register	Offset
FN_MOD_BM_ISS	8h

Function

This register sets the issuing capability of the preceding switch arbitration scheme to 1.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 FN_MOD_BM_I SS	Read channel QoS value Legal programmable values are 0, 1, 2 and 3. 0: Default setting 1: Set read issuing capability to one outstanding transaction 2: Set write issuing capability to one outstanding transaction 3: Set both read and write issuing capability to one outstanding transaction NOTE It is recommended to keep its reset value to obtain the best performance.

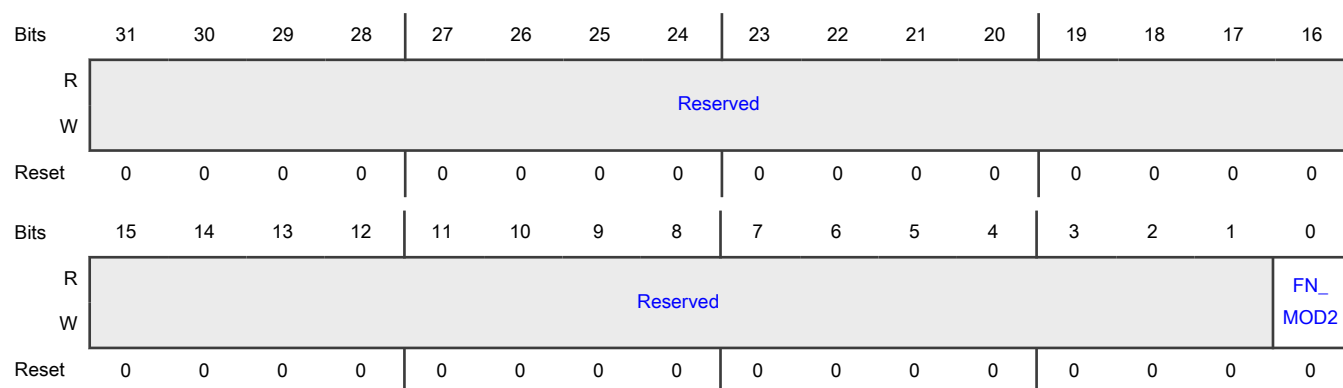
11.2.1.3 Bypass Merge (FN_MOD2)

Offset

Register	Offset
FN_MOD2	24h

Function

If this register is set, the network does not alter any transactions that could pass through legally without alteration.If bypass_merge is asserted, the network does not perform any packing of beats to match the optimum SIZE, up to the output data width SIZE.

Diagram**Fields**

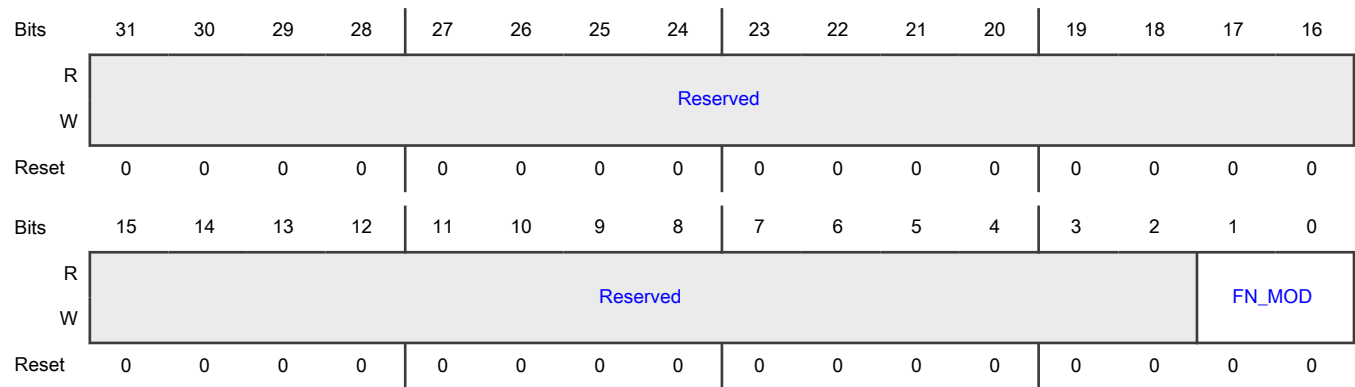
Field	Function
31-1 —	Reserved
0 FN_MOD2	Bypass Merge Legal programmable values are 0, 1, 2 and 3. 0: Not bypass merging functionality 1: Bypass merging functionality NOTE It is recommended to keep its reset value to obtain the best performance.

11.2.1.4 Issuing Functionality Modification (FN_MOD)**Offset**

Register	Offset
FN_MOD	108h

Function

This register sets the block issuing capability to one outstanding transaction.

Diagram**Fields**

Field	Function
31-2 —	Reserved
1-0 FN_MOD	Bypass Merge Legal programmable values are 0, 1, 2 and 3. 0: Default setting 1: Set read issuing capability to one outstanding transaction 2: Set write issuing capability to one outstanding transaction 3: Set both read and write issuing capability to one outstanding transaction NOTE It is recommended to keep its reset value to obtain the best performance.

11.2.2 AXI SWITCH ASIB register descriptions

Port M0 ASIB registers are at address GPV0_BASE + 0x42000.

Port M1 ASIB registers are at address GPV0_BASE + 0x43000.

11.2.2.1 AXI_SWITCH_ASIB memory map

GCDC_SWITCH0.AXI_SWITCH_ASIB base address: 402C_2000h

Offset	Register	Width (In bits)	Access	Reset value
100h	Read channel QoS value (READ_QOS)	32	RW	0000_0000h
104h	WRITE channel QoS value (WRITE_QOS)	32	RW	0000_0001h
108h	Issuing Functionality Modification (FN_MOD)	32	RW	0000_0000h

11.2.2.2 Read channel QoS value (READ_QOS)

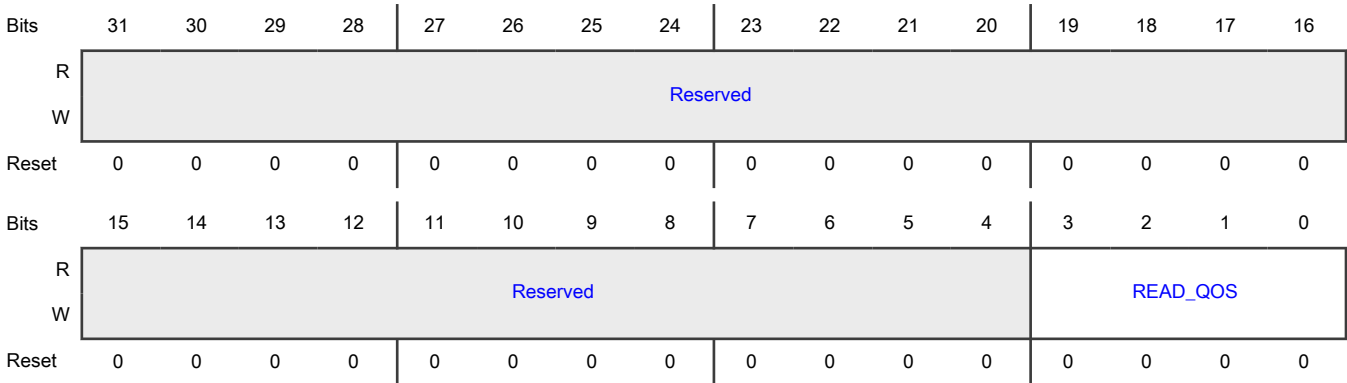
Offset

Register	Offset
READ_QOS	100h

Function

Set the priority of master's read.

Diagram



Fields

Field	Function
31-4 —	Reserved
3-0 READ_QOS	Read channel QoS value The priority level is used when the master's read transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority. 0: the lowest arbitration priority 1: higher than 0 2: higher than 1, 0 3: higher than 2, 1, 0 ... 15: the highest priority

11.2.2.3 WRITE channel QoS value (WRITE_QOS)

Offset

Register	Offset
WRITE_QOS	104h

Function

Set the priority of master's write.

Diagram

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

Fields

Field	Function
31-4 —	Reserved
3-0 WRITE_QOS	Write channel QoS value The priority level is used when the master's write transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority. 0: the lowest arbitration priority 1: higher than 0 2: higher than 1, 0 3: higher than 2, 1, 0 ... 15: the highest priority

11.2.2.4 Issuing Functionality Modification (FN_MOD)

Offset

Register	Offset
FN_MOD	108h

Function

Issuing override. This register sets the block issuing capability to one outstanding transaction.

NOTE

It is recommended to keep its reset value to obtain the best performance.

Diagram

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

Fields

Field	Function
31-2 —	Reserved
1-0 WRITE_QOS	Write channel QoS value Sets block issuing capability to one transaction and you can configure the bits as follows: 0: Default setting. 1: Set read issuing capability to one outstanding transaction. 2: Set write issuing capability to one outstanding transaction. 3: Set both read and write issuing capability to one outstanding transaction.

Chapter 12

IO PAD Controller (IOPCTL)

12.1 Chip-specific IOPCTL information

Table 52. Reference links to related information

Topic	Related module	Reference
Full description	IOPCTL	IOPCTL
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

12.1.1 Module instances

This device has one instance of the IOPCTL module. IOPCTL0.

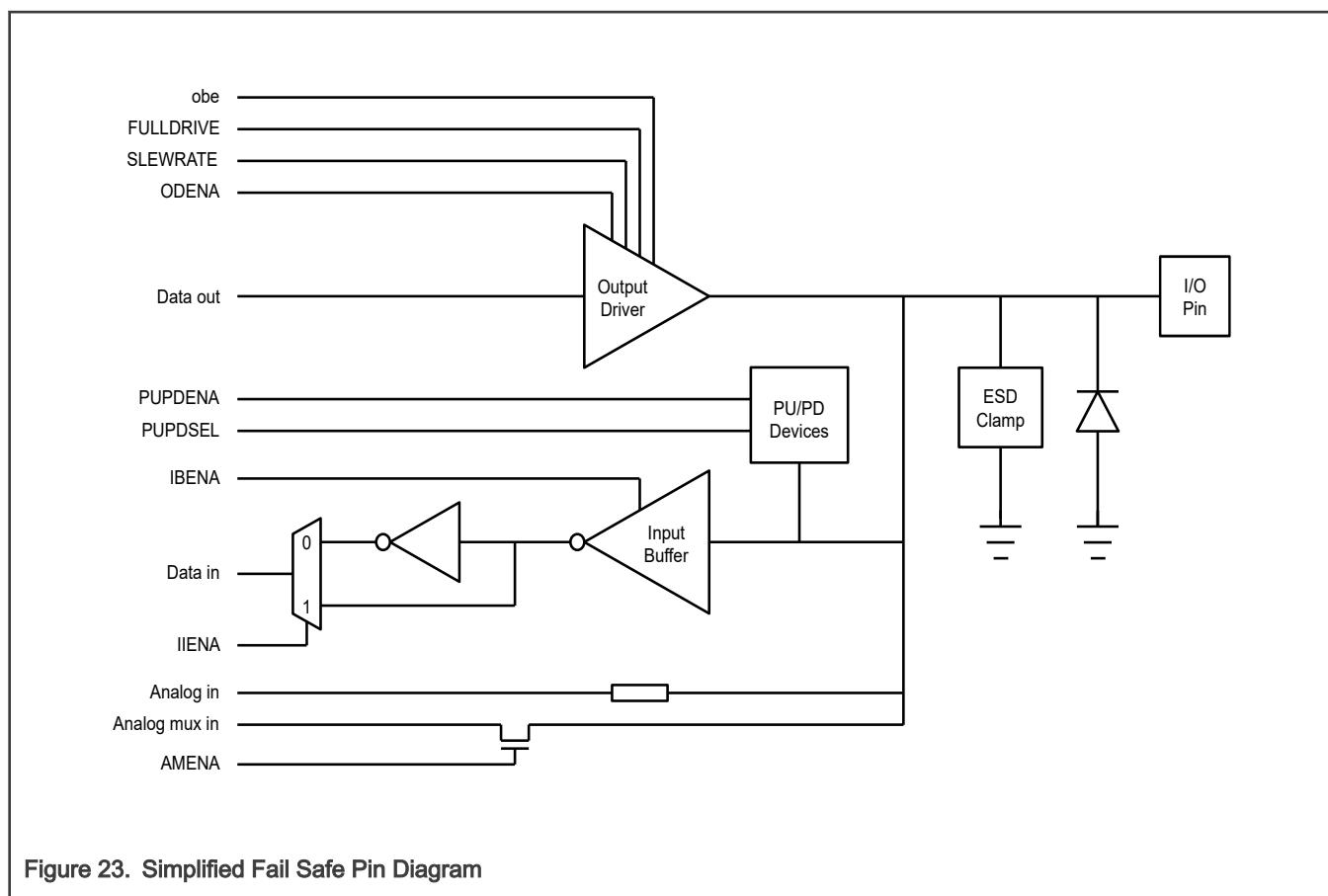
12.2 Overview

This section describes the IO PAD Controllers on the chip. Each port pin (PIOa_n, PMIC_I2C_SCL and PMIC_I2C_SDA) has an IOPCTL register assigned to control the characteristics of the pin. This chip has two types of GPIO pads: Fail-Safe pads and High-Speed pads. A Fail-Safe pad requires that the pad will not interfere with the bus when the VDDIO domain is not powered, which is a typical requirement for chips on an I2C bus. The main difference in the structure of these two pads is that the High-Speed pads have diodes to VDD and VSS, while the Fail-Safe pads only have diodes to VSS, as shown in the pin diagrams below.

On this chip, the IO pins associated with power domains VDDIO_0/1/2/4 can only support input voltages up to 1.8V when powered. IO pins in the VDDIO_3 power domain can support input voltages up to 3.6V (when VDDIO_3 is 3.6V).

Refer to the data sheet for the specific pins that have Fail-Safe or High-Speed pads and the maximum input voltage for the Fail-Safe pins when the VDDIO supply is unpowered.

12.2.1 Block diagram



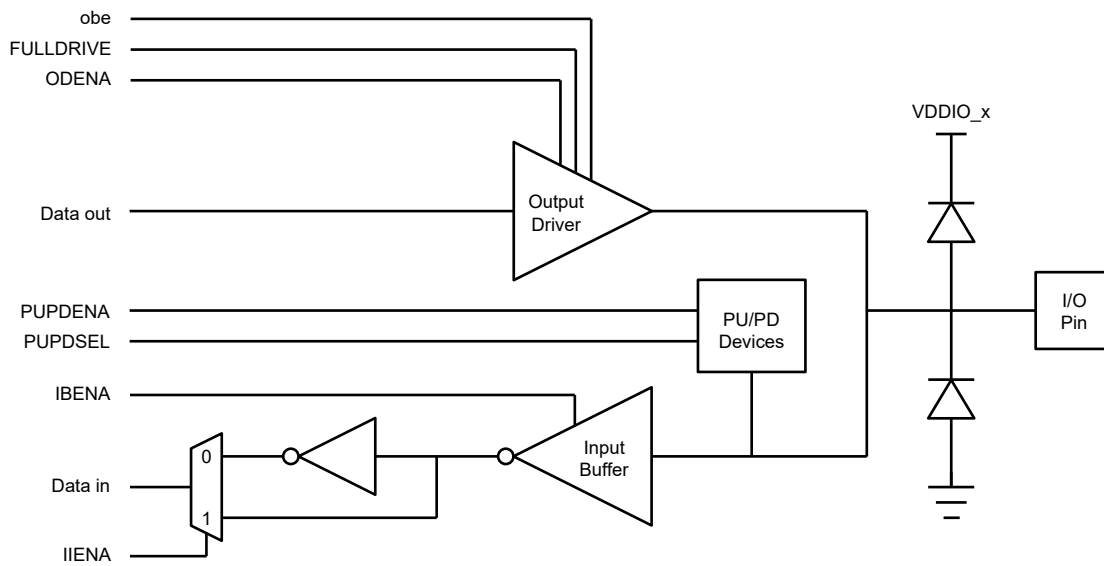


Figure 24. Simplified High Speed Pin Diagram

12.2.2 Features

The following properties may be configured for each standard port pin:

- Pull-up resistor, pull-down resistor, or neither.
- Input buffer enable/disable.
- Drive strength.
- Pseudo open-drain mode.

Additional properties may be configured for all Fail Safe port pins:

- Slew rate.
- Analog multiplexor enable for pins that include an analog function.

12.3 Signals

For details on the available port pins and functions supported, please refer RT500_pinouts spreadsheet attachment and the RT500 Datasheet.

12.4 Memory map and register definition

This section includes the IOPCTL module memory map and detailed descriptions of all registers.

The IOPCTL registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For

instance, pins that have an I²C function can be configured for different I²C-bus modes, while pins that have an analog alternate function have an analog mode can be selected.

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore care should be taken to avoid this situation.

12.4.1 IOPCTL register descriptions

12.4.1.1 IO PAD CONTROLLER Registers memory map

IOPCTL base address: 4000_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 36Ch	IOPCTL configuration (PIO0_0 - PIO6_27)	32	RW	0000_0000h
400h	Flexcomm 15 SCL (FC15_I2C_SCL)	32	RW	0000_0000h
404h	Flexcomm 15 SDA (FC15_I2C_SDA)	32	RW	0000_0000h

12.4.1.2 IOPCTL configuration (PIO0_0 - PIO6_27)

Offset

Registers in this array exist only for the following combinations of index values.

Index <i>a</i>	Index <i>b</i>
0–5	0–31
6	0–27

Register	Offset
PIOa_b	0h + (a × 80h) + (b × 4h)

Function

Each port pin PIO has one IOPCTL register assigned to control the characteristics of the pin.

NOTE

The reset value for PIO2_25 and PIO2_26 is 0x41, selecting the Serial Wire Debug functions and enabling the input buffers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				IIENA	ODEN A	AMEN A	FULLD RI...	SLEW RATE	IBENA	PUPD SEL	PUPD ENA	FSEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11 IIENA	Input Invert Enable This bit avoids the need to include an external inverter on an input that is meant to be the opposite polarity of the external signal. 0b - Disabled, Input function is not inverted 1b - Enabled, Input is function inverted
10 ODENA	Open-drain mode enable 0b - Normal push-pull output 1b - Pseudo open-drain output (high drive disabled)
9 AMENA	Analog Mux Enable 0b - Analog multiplexor disabled, required for digital pin function 1b - Analog multiplexor enabled, required for analog pin function
8 FULLDRIVE	Drive Selector The “Full output drive” setting has twice the high and low drive capability of the “Normal output drive” setting. The drive strength depends on the I/O supply voltage for the specific pin, which can be found in the device datasheet. See device datasheet for details. 0b - Normal output drive 1b - Full output drive, twice the drive of normal mode.
7 SLEWRATE	Slew Rate Control Enabling output slew rate control limits the speed at which the pin can toggle. This effect is voltage and load dependent, see the device datasheet for details. 0b - Standard mode, output slew rate is not controlled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Slow mode, output slew rate control is enabled, limiting the output rate change and maximum toggle frequency. See device datasheet for details.
6 IBENA	Input Buffer Enable This bit controls the input buffer for digital functions. It can be disabled when not needed and must be disabled when the pin is used for an analog function. Input buffer is not needed for a GPIO that is used as an output and is never read back by software. Peripheral functions that are generally outputs do need the input buffer enabled because the peripheral uses it to sense the situation on the pin as part of its function. If there is any doubt, it is best to leave the input buffer enabled 0b - Input buffer disabled 1b - Input buffer enabled
5 PUPDSEL	Pullup or Pulldown Selector These bits determine pull-up, pull-down state of port pins. 0b - Pull-down 1b - Pull-up
4 PUPDENA	Pullup / Pulldown Enable These bits determine pull-up or pull-down state of resistors that may optionally be enabled on any port pin. 0b - Disable 1b - Enable
3-0 FSEL	Function Selector (Digital Function) Selects pin function. 0000b - Function 0 0001b - Function 1 0010b - Function 2 0011b - Function 3 0100b - Function 4 0101b - Function 5 0110b - Function 6 0111b - Function 7 1000b - Function 8 1001b - Function 9 1010b - Function 10 1011b - Function 11

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1100b - Function 12
	1101b - Function 13
	1110b - Function 14
	1111b - Function 15

12.4.1.3 Flexcomm 15 SCL (FC15_I2C_SCL)

Offset

Register	Offset
FC15_I2C_SCL	400h

Function

Pin control for PMIC_I2C_SCL

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				IIENA	ODEN	AMEN	FULLD	SLEW	IBENA	PUPD	PUPD	FSEL			
W					A	A	RI...	RATE			SEL	ENA				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11 IIENA	Input Invert Enable This bit avoids the need to include an external inverter on an input that is meant to be the opposite polarity of the external signal. 0b - Disabled, Input function is not inverted 1b - Enabled, Input is function inverted

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 ODENA	Open-drain mode enable 0b - Normal push-pull output 1b - Pseudo open-drain output (high drive disabled)
9 AMENA	Analog Mux Enable 0b - Analog multiplexor disabled, required for digital pin function 1b - Analog multiplexor enabled, required for analog pin function
8 FULLDRIVE	Drive Selector The "Full output drive" setting has twice the high and low drive capability of the "Normal output drive" setting. The drive strength depends on the I/O supply voltage for the specific pin, which can be found in the device datasheet. See device datasheet for details. 0b - Normal output drive 1b - Full output drive, twice the drive of normal mode.
7 SLEWRATE	Slew Rate Control Enabling output slew rate control limits the speed at which the pin can toggle. This effect is voltage and load dependent, see the device datasheet for details. 0b - Standard mode, output slew rate is not controlled. 1b - Slow mode, output slew rate control is enabled, limiting the output rate change and maximum toggle frequency. See device datasheet for details.
6 IBENA	Input Buffer Enable This bit controls the input buffer for digital functions. It can be disabled when not needed and must be disabled when the pin is used for an analog function. Input buffer is not needed for a GPIO that is used as an output and is never read back by software. Peripheral functions that are generally outputs do need the input buffer enabled because the peripheral uses it to sense the situation on the pin as part of its function. If there is any doubt, it is best to leave the input buffer enabled 0b - Input buffer disabled 1b - Input buffer enabled
5 PUPDSEL	Pullup or Pulldown Selector These bits determine pull-up, pull-down state of port pins. 0b - Pull-down 1b - Pull-up
4 PUPDENA	Pullup / Pulldown Enable These bits determine pull-up or pull-down state of resistors that may optionally be enabled on any port pin. 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
3-0 FSEL	Function Selector (Digital Function) Selects pin function. 0000b - Function 0 0001b - Function 1 0010b - Function 2 0011b - Function 3 0100b - Function 4 0101b - Function 5 0110b - Function 6 0111b - Function 7 1000b - Function 8 1001b - Function 9 1010b - Function 10 1011b - Function 11 1100b - Function 12 1101b - Function 13 1110b - Function 14 1111b - Function 15

12.4.1.4 Flexcomm 15 SDA (FC15_I2C_SDA)

Offset

Register	Offset
FC15_I2C_SDA	404h

Function

Pin control for PMIC_I2C_SDA

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				IIENA	ODEN A	AMEN A	FULLD RI...	SLEW RATE	IBENA	PUPD SEL	PUPD ENA	FSEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved
11 IIENA	Input Invert Enable This bit avoids the need to include an external inverter on an input that is meant to be the opposite polarity of the external signal. 0b - Disabled, Input function is not inverted 1b - Enabled, Input is function inverted
10 ODENA	Open-drain mode enable 0b - Normal push-pull output 1b - Pseudo open-drain output (high drive disabled)
9 AMENA	Analog Mux Enable 0b - Analog multiplexor disabled, required for digital pin function 1b - Analog multiplexor enabled, required for analog pin function
8 FULLDRIVE	Drive Selector The “Full output drive” setting has twice the high and low drive capability of the “Normal output drive” setting. The drive strength depends on the I/O supply voltage for the specific pin, which can be found in the device datasheet. See device datasheet for details. 0b - Normal output drive 1b - Full output drive, twice the drive of normal mode.
7 SLEWRATE	Slew Rate Control Enabling output slew rate control limits the speed at which the pin can toggle. This effect is voltage and load dependent, see the device datasheet for details. 0b - Standard mode, output slew rate is not controlled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Slow mode, output slew rate control is enabled, limiting the output rate change and maximum toggle frequency. See device datasheet for details.
6 IBENA	Input Buffer Enable This bit controls the input buffer for digital functions. It can be disabled when not needed and must be disabled when the pin is used for an analog function. Input buffer is not needed for a GPIO that is used as an output and is never read back by software. Peripheral functions that are generally outputs do need the input buffer enabled because the peripheral uses it to sense the situation on the pin as part of its function. If there is any doubt, it is best to leave the input buffer enabled 0b - Input buffer disabled 1b - Input buffer enabled
5 PUPDSEL	Pullup or Pulldown Selector These bits determine pull-up, pull-down state of port pins. 0b - Pull-down 1b - Pull-up
4 PUPDENA	Pullup / Pulldown Enable These bits determine pull-up or pull-down state of resistors that may optionally be enabled on any port pin. 0b - Disable 1b - Enable
3-0 FSEL	Function Selector (Digital Function) Selects pin function. 0000b - Function 0 0001b - Function 1 0010b - Function 2 0011b - Function 3 0100b - Function 4 0101b - Function 5 0110b - Function 6 0111b - Function 7 1000b - Function 8 1001b - Function 9 1010b - Function 10 1011b - Function 11

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1100b - Function 12
	1101b - Function 13
	1110b - Function 14
	1111b - Function 15

12.5 RT500 pin multiplexing table (FOWLP)

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
F14	PIO0_0	PIO0_0	PIO0_0	FC0_SCK			CTIMER0_MAT0	I2S_BRIDGE_CLK_IN	GPIO_INT_BMAT		SEC_PIO0_0	
G16	PIO0_1	PIO0_1	PIO0_1	FC0_TXD_SCL_MI SO_WS			CTIMER0_MAT1	I2S_BRIDGE_WS_IN			SEC_PIO0_1	
H16	PIO0_2	PIO0_2	PIO0_2	FC0_RXD_SDA_MO SI_DATA			CTIMER0_MAT2	I2S_BRIDGE_DATA_IN			SEC_PIO0_2	
H15	PIO0_3	PIO0_3	PIO0_3	FC0_CTS_SDA_SS EL0			CTIMER0_MAT3	FC1_SSE_L2			SEC_PIO0_3	
H14	PIO0_4	PIO0_4	PIO0_4	FC0_RTS_SCL_SS EL1			CTIMER1_NP0	FC1_SSE_L3		CMP0_OUT	SEC_PIO0_4	
F16	PIO0_5 / CH0A	PIO0_5	PIO0_5	FC0_SSE_L2	SCT0_GPI0	SCT0_OUT0_T0	CTIMER1_NP1				SEC_PIO0_5	
F17	PIO0_6 / CH0B	PIO0_6	PIO0_6	FC0_SSE_L3	SCT0_GPI1	SCT0_OUT0_T1	CTIMER0_MAT0				SEC_PIO0_6	
J15	PIO0_7 / TRST	PIO0_7	PIO0_7	FC1_SCK	SCT0_GPI4	SCT0_OUT0_T4	CTIMER1_MAT0	I2S_BRIDGE_CLK_OUT			SEC_PIO0_7	
H12	PIO0_8 / TCK	PIO0_8	PIO0_8	FC1_TXD_SCL_MI SO_WS	SCT0_GPI5	SCT0_OUT0_T5	CTIMER1_MAT1	I2S_BRIDGE_WS_OUT			SEC_PIO0_8	
H17	PIO0_9 / TMS	PIO0_9	PIO0_9	FC1_RXD_SDA_MO SI_DATA	SCT0_GPI6	SCT0_OUT0_T6	CTIMER1_MAT2	I2S_BRIDGE_DATA_OUT			SEC_PIO0_9	
K16	PIO0_10 / TDI	PIO0_10	PIO0_10	FC1_CTS_SDA_SS EL0	SCT0_GPI7	SCT0_OUT0_T7	CTIMER1_MAT3	FC0_SSE_L2			SEC_PIO0_10	

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
K15	PIO0_11 / TDO	PIO0_11	PIO0_11	FC1_RTS _SCL_SS EL1	SCT0_G PI0	SCT0_OU T8	CTIMER_I NP2	FC0_SSE L3			SEC_PIO 0_11	
E14	PIO0_12 / CH1A	PIO0_12	PIO0_12	FC1_SSE L2	SCT0_G PI2	SCT0_OU T2	CTIMER_I NP3				SEC_PIO 0_12	
F15	PIO0_13 / CH1B	IO0_13	PIO0_13	FC1_SSE L3	SCT0_G PI3	SCT0_OU T3	CTIMER0 _MAT1				SEC_PIO 0_13	
B12	PIO0_14	PIO0_14	PIO0_14	FC2_SCK	SCT0_G PI0	SCT0_OU T0	CTIMER2 _MAT0	I2S_BRID GE_CLK_I N			SEC_PIO 0_14	
B15	PIO0_15	PIO0_15	PIO0_15	FC2_TXD _SCL_MI SO_WS	SCT0_G PI1	SCT0_OU T1	CTIMER2 _MAT1	I2S_BRID GE_WS_I N			SEC_PIO 0_15	
A16	PIO0_16	PIO0_16	PIO0_16	FC2_RXD _SDA_MO SI_DATA	SCT0_G PI2	SCT0_OU T2	CTIMER2 _MAT2	I2S_BRID GE_DATA _IN			SEC_PIO 0_16	
B17	PIO0_17	PIO0_17	PIO0_17	FC2_CTS _SDA_SS EL0	SCT0_G PI3	SCT0_OU T3	CTIMER2 _MAT3	FC5_SSE L2			SEC_PIO 0_17	
B16	PIO0_18	PIO0_18	PIO0_18	FC2_RTS _SCL_SS EL1	SCT0_G PI6	SCT0_OU T6	CTIMER_I NP4	FC5_SSE L3			SEC_PIO 0_18	
F13	PIO0_19 / CH2_A	PIO0_19	PIO0_19	FC2_SSE L2	SCT0_G PI4	SCT0_OU T4	CTIMER_I NP5	UTICK_C AP0			SEC_PIO 0_19	
A14	PIO0_21	PIO0_21	PIO0_21	FC3_SCK	SCT0_G PI5	SCT0_OU T5	CTIMER3 _MAT0	CTIMER_I NP11	TRACECL K		SEC_PIO 0_21	
B14	PIO0_22	PIO0_22	PIO0_22	FC3_TXD _SCL_MI SO_WS	SCT0_G PI6	SCT0_OU T6	CTIMER3 _MAT1	CTIMER_I NP7	TRACEDA TA[0]		SEC_PIO 0_22	

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLPL)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
C13	PIO0_23	PIO0_23	PIO0_23	FC3_RXD_SDA_MO SI_DATA	SCT0_G PI7	SCT0_OU T8	CTIMER3 _MAT2	CTIMER0 _MAT3	TRACEDA TA[1]		SEC_PIO 0_23	
D13	PIO0_24	PIO0_24	PIO0_24	FC3_CTS _SDA_SS EL0	SCT0_G PI2	SCT0_OU T9	CTIMER3 _MAT3	FC2_SSE L2	TRACEDA TA[2]	CLKOUT	SEC_PIO 0_24	
C12	PIO0_25	PIO0_25	PIO0_25	FC3_RTS _SCL_SS EL1		FREQME_ GPIO_CLK K	CTIMER_I NP6	FC2_SSE L3	TRACEDA TA[3]	CLKIN	SEC_PIO 0_25	
A12	PIO0_28	PIO0_28	PIO0_28	FC4_SCK			CTIMER4 _MAT0	I2S_BRID GE_CLK_ OUT			SEC_PIO 0_28	
B11	PIO0_29	PIO0_29	PIO0_29	FC4_TXD _SCL_MI SO_WS			CTIMER4 _MAT1	I2S_BRID GE_WS_ OUT			SEC_PIO 0_29	
D14	PIO0_30	PIO0_30	PIO0_30	FC4_RXD _SDA_MO SI_DATA			CTIMER4 _MAT2	I2S_BRID GE_DATA _OUT			SEC_PIO 0_30	
D12	PIO0_31	PIO0_31	PIO0_31	FC4_CTS _SDA_SS EL0	SCT0_G PI0	SCT0_OU T6	CTIMER4 _MAT3	FC3_SSE L2			SEC_PIO 0_31	
A10	PIO1_0	PIO1_0	PIO1_0	FC4_RTS _SCL_SS EL1	SCT0_G PI1	SCT0_OU T7	CTIMER_I NP8	FC3_SSE L3				
K2	PIO1_3	PIO1_3	PIO1_3	FC5_SCK					HS_SPI1_S CK			
K1	PIO1_4	PIO1_4	PIO1_4	FC5_TXD _SCL_MI SO_WS					HS_SPI1_ MISO			

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
L2	PIO1_5	PIO1_5	PIO1_5	FC5_RXD _SDA_MO SI_DATA					HS_SPI1_ MOSI			
N4	PIO1_6	PIO1_6	PIO1_6	FC5_CTS _SDA_SS EL0	SCT0_G PI4	SCT0_OU T4		FC4_SSE L2	HS_SPI1_S SELN0			
M1	PIO1_7	PIO1_7	PIO1_7	FC5_RTS _SCL_SS EL1	SCT0_G PI5	SCT0_OU T5	CTIMER_I NP9	FC4_SSE L3	HS_SPI1_S SELN1			
M5	PIO1_10	PIO1_10	PIO1_10	MCLK		FREQUE_ GPIO_CL K	CTIMER_I NP10			CLKOUT		
K13	PIO1_11	PIO1_11	PIO1_11	HS_SPI0_ SCK			CTIMER2 _MAT0					
K14	PIO1_12	PIO1_12	PIO1_12	HS_SPI0_ MISO			CTIMER2 _MAT1					
K17	PIO1_13	PIO1_13	PIO1_13	HS_SPI0_ MOSI			CTIMER2 _MAT2					
L16	PIO1_14	PIO1_14	PIO1_14	HS_SPI0_ SSELN0			CTIMER2 _MAT3					
M16	PIO1_15 / ISP0	PIO1_15	PIO1_15	HS_SPI0_ SSELN1			CTIMER3 _MAT0					
T17	PIO1_18	PIO1_18	PIO1_18	FLEXSPI0 _SCLK	SCT0_G PI0		CTIMER3 _MAT3					
U16	PIO1_19	PIO1_19	PIO1_19	FLEXSPI0 _SS0_N	SCT0_O UT0		CTIMER4 _MAT0			CLKOUT		
T15	PIO1_20	PIO1_20	PIO1_20	FLEXSPI0 _DATA0	SCT0_G PI1		CTIMER4 _MAT1					

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
T14	PIO1_21	PIO1_21	PIO1_21	FLEXSPI0_DATA1	SCT0_O UT1		CTIMER4_MAT2					
R13	PIO1_22	PIO1_22	PIO1_22	FLEXSPI0_DATA2	SCT0_G PI2		CTIMER4_MAT3					
R12	PIO1_23	PIO1_23	PIO1_23	FLEXSPI0_DATA3	SCT0_O UT2		CTIMER_I NP8					
N12	PIO1_24	PIO1_24	PIO1_24	FLEXSPI0_DATA4	SCT0_G PI3							
R14	PIO1_25	PIO1_25	PIO1_25	FLEXSPI0_DATA5	SCT0_O UT3							
P14	PIO1_26	PIO1_26	PIO1_26	FLEXSPI0_DATA6	SCT0_G PI4							
P13	PIO1_27	PIO1_27	PIO1_27	FLEXSPI0_DATA7	SCT0_O UT4							
U14	PIO1_28	PIO1_28	PIO1_28	FLEXSPI0_DQS	SCT0_G PI5							
U12	PIO1_29	PIO1_29	PIO1_29	FLEXSPI0_SS1_N	SCT0_O UT5	UTICK_C AP2	CTIMER_I NP13	FLEXSPI0_SCLK_N				
R5	PIO1_30	PIO1_30	PIO1_30	SD0_CLK	SCT0_G PI0							
R6	PIO1_31	PIO1_31	PIO1_31	SD0_CMD	SCT0_G PI1							
U4	PIO2_0	PIO2_0	PIO2_0	SD0_D[0]	SCT0_G PI2							SmartDM A_PIO0
T4	PIO2_1	PIO2_1	PIO2_1	SD0_D[1]	SCT0_G PI3							SmartDM A_PIO1
T7	PIO2_2	PIO2_2	PIO2_2	SD0_D[2]	SCT0_O UT0							SmartDM A_PIO2

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWL P)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
U6	PIO2_3	PIO2_3	PIO2_3	SD0_D[3]	SCT0_O UT1							SmartDM A_PIO3
P6	PIO2_4	PIO2_4	PIO2_4	SD0_WR_ PRT	SCT0_O UT2			SD0_DS				SmartDM A_PIO4
P5	PIO2_5	PIO2_5	PIO2_5	SD0_D[4]	SCT0_O UT3			FC8_SCK				SmartDM A_PIO5
R4	PIO2_6	PIO2_6	PIO2_6	SD0_D[5]	SCT0_G PI4		CTIMER1 _MAT0	FC8_TXD _SCL_MI SO_WS				SmartDM A_PIO6
P4	PIO2_7	PIO2_7	PIO2_7	SD0_D[6]	SCT0_G PI5		CTIMER1 _MAT1	FC8_RXD _SDA_MO SI_DATA				SmartDM A_PIO7
T6	PIO2_8	PIO2_8	PIO2_8	SD0_D[7]	SCT0_O UT4		CTIMER1 _MAT2	FC8_CTS _SDA_SS EL0				SmartDM A_PIO8
T3	PIO2_9	PIO2_9	PIO2_9	SD0_CAR D_DET_N	SCT0_O UT5		CTIMER1 _MAT3	FC8_CTS _SDA_SS EL1				SmartDM A_PIO9
N5	PIO2_10	PIO2_10	PIO2_10	SD0_RES ET_N	SCT0_G PI6		CTIMER2 _MAT0	FC8_SSE L2				SmartDM A_PIO10
R2	PIO2_11	PIO2_11	PIO2_11	SD0_VOL T	SCT0_G PI7		CTIMER2 _MAT1	FC8_SSE L3				SmartDM A_PIO11
E15	PIO2_14 / CMP0_A	PIO2_14	PIO2_14		SCT0_O UT8		CTIMER_I NP1			32KHZ_C LKOUT		SmartDM A_PIO14
D17	PIO2_15 / CMP0_D	PIO2_15	PIO2_15		SCT0_O UT9					CLKIN		SmartDM A_PIO15
N3	PIO2_24	PIO2_24	PIO2_24	SWO								SmartDM A_PIO24

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
M2	PIO2_25	PIO2_25	PIO2_25	SWCLK								SmartDM A_PIO25
M4	PIO2_26	PIO2_26	PIO2_26	SWDIO								SmartDM A_PIO26
M3	PIO2_27	PIO2_27	PIO2_27	USB1_OV ERCURR ENTN								SmartDM A_PIO27
P1	PIO2_28	PIO2_28	PIO2_28	USB1_PO RTPWRN								SmartDM A_PIO28
B10	PIO2_29	PIO2_29	PIO2_29		SCT0_O UT0			CLKOUT				SmartDM A_PIO29
D10	PIO2_30	PIO2_30	PIO2_30	I3C0_SCL	SCT0_O UT3			CLKIN		CMP0_OU T		SmartDM A_PIO30
C14	PIO2_31 / CMP0_B	PIO2_31	PIO2_31	I3C0_PU R	SCT0_O UT7	UTICK_C AP3	CTIMER_I NP15	SWO				SmartDM A_PIO31
T2	USB1_VB US	USB1_VB US										
M6	VDD1V8	VDD1V8	VDD1V8									
K5	USB1_VD D3V3	USB1_VD D3V3										
T1	USB1_DM	USB1_DM										
U2	USB1_DP	USB1_DP										
E4	PMIC_MO DE1	PMIC_MO DE1										
D3	PMIC_MO DE0	PMIC_MO DE0										

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
K6	PMIC_I2C_SDA	PMIC_I2C_SDA										
K4	PMIC_I2C_SCL	PMIC_I2C_SCL										
D5	PMIC_IR_Q_N	PMIC_IR_Q_N										
C5	LDO_ENA_BLE	LDO_ENA_BLE										
B4	XTALIN	XTALIN										
A4	XTALOUT	XTALOUT										
A2	RTCXIN	RTCXIN										
B3	RTCXOUT	RTCXOUT										
C4	RESETN	RESETN										
F12	VREFP	VREFP										
G12	VREFN	VREFN										
D16	PIO3_1	PIO3_1	PIO3_1	PDM_CLK ₂₃	PDM_D ATA23			FC0_TXD _SCL_MI SO_WS	I3C1_SCL			
C16	PIO3_2	PIO3_2	PIO3_2	PDM_CLK ₄₅	PDM_D ATA45			FC0_RXD _SDA_MO SI_DATA	I3C1_SDA			
D15	PIO3_3	PIO3_3	PIO3_3	PDM_CLK ₆₇	PDM_D ATA67	LCD_D23		FC0_CTS _SDA_SS ELO	I3C1_PUR	CMP0_OUT		
A8	PIO3_8	PIO3_8	PIO3_8	SD1_CLK	LCD_D9		CTIMER0_MAT0		FC10_SCK			

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
B8	PIO3_9	PIO3_9	PIO3_9	SD1_CM D	LCD_D1 0		CTIMER0 _MAT1		FC10_TXD _SCL_MIS O			
C8	PIO3_10	PIO3_10	PIO3_10	SD1_D[0]	LCD_D1 1		CTIMER0 _MAT2		FC10_RXD _SDA_MO SI			
C10	PIO3_11	PIO3_11	PIO3_11	SD1_D[1]	LCD_D1 2		CTIMER0 _MAT3		FC10_CTS _SDA_SSE LN0			
A6	PIO3_12	PIO3_12	PIO3_12	SD1_D[2]	LCD_D1 3		CTIMER_I NP0		FC10_RTS _SCL_SSE LN1			
B7	PIO3_13	PIO3_13	PIO3_13	SD1_D[3]	LCD_D1 4		CTIMER_I NP1		FC10_SSE LN2			
D9	PIO3_14	PIO3_14	PIO3_14	SD1_WR_ PRT	LCD_D1 5		CTIMER3 _MAT0	SD1_DS	FC10_SSE LN3			
E10	PIO3_15	PIO3_15	PIO3_15	SD1_D[4]	LCD_D1 6		CTIMER3 _MAT1	FC5_SCK				
C9	PIO3_16	PIO3_16	PIO3_16	SD1_D[5]	LCD_D1 7		CTIMER3 _MAT2	FC5_TXD _SCL_MI SO_WS				
D8	PIO3_17	PIO3_17	PIO3_17	SD1_D[6]	LCD_D1 8		CTIMER3 _MAT3	FC5_RXD _SDA_MO SI_DATA				
B6	PIO3_18	PIO3_18	PIO3_18	SD1_D[7]	LCD_D1 9		CTIMER4 _MAT0	FC5_CTS _SDA_SS ELO				
C6	PIO3_19	PIO3_19	PIO3_19	SD1_CAR D_DET_N	LCD_D2 0		CTIMER4 _MAT1	MCLK				

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
D6	PIO3_20	PIO3_20	PIO3_20	SD1_RES ET_N	LCD_D2 1		CTIMER4 _MAT2					
E5	PIO3_21	PIO3_21	PIO3_21	SD1_VOL T	LCD_D2 2		CTIMER4 _MAT3		GPIO_INT_ BMAT			
R16	PIO3_25	PIO3_25	PIO3_25	FC6_SCK								
T16	PIO3_26	PIO3_26	PIO3_26	FC6_TXD _SCL_MI SO_WS								
N14	PIO3_27	PIO3_27	PIO3_27	FC6_RXD _SDA_MO SI_DATA								
N13	PIO3_28 / ISP1	PIO3_28	PIO3_28	FC6_CTS _SDA_SS EL0								
M13	PIO3_29 / ISP2	PIO3_29	PIO3_29	FC6_RTS _SCL_SS EL1								
N15	PIO4_0	PIO4_0	PIO4_0	FC7_SCK			FREQME _GPIO_C LK			CLKOUT		
M15	PIO4_1	PIO4_1	PIO4_1	FC7_TXD _SCL_MI SO_WS						CLKIN		
M17	PIO4_2	PIO4_2	PIO4_2	FC7_RXD _SDA_MO SI_DATA								
M14	PIO4_3	PIO4_3	PIO4_3	FC7_CTS _SDA_SS EL0								

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
P17	PIO4_4	PIO4_4	PIO4_4	FC7_RTS _SCL_SS EL1				FC1_SCK				
P16	PIO4_5	PIO4_5	PIO4_5	FC7_SSE L2				FC1_TXD _SCL_MI SO_WS				
P15	PIO4_6	PIO4_6	PIO4_6	FC7_SSE L3				FC1_RXD _SDA_MO SI_DATA				
D2	MIPI_DSI _CLKP	MIPI_DSI _CLKP										
D1	MIPI_DSI _CLKN	MIPI_DSI _CLKN										
B1	MIPI_DSI _D0P	MIPI_DSI _D0P										
C2	MIPI_DSI _D0N	MIPI_DSI _D0N										
E3	MIPI_DSI _D1P	MIPI_DSI _D1P										
H2	PIO4_11	PIO4_11	PIO4_11	FC2_SCK	FLEXSP I1_SCLK		SD1_CLK					
H1	PIO4_12	PIO4_12	PIO4_12	FC2_TXD _SCL_MI SO_WS	FLEXSP I1_DATA 0		SD1_CM D					
G2	PIO4_13	PIO4_13	PIO4_13	FC2_RXD _SDA_MO SI_DATA	FLEXSP I1_DATA 1		SD1_D[0]					

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
F1	PIO4_14	PIO4_14	PIO4_14	FC2_CTS _SDA_SS EL0	FLEXSP I1_DATA 2		SD1_D[1]					
K3	PIO4_15	PIO4_15	PIO4_15	FC2_RTS _SCL_SS EL1	FLEXSP I1_DATA 3		SD1_D[2]					
H3	PIO4_16	PIO4_16	PIO4_16	FC2_SSE L2	FLEXSP I1_DQS		SD1_D[3]					
F2	PIO4_17	PIO4_17	PIO4_17	FC2_SSE L3	FLEXSP I1_SS1_ N	FLEXSPI1 _SCLK_N	SD1_WR_ PRT					
E13	PIO4_18 / CH6_A	PIO4_18	PIO4_18		FLEXSP I1_SS0_ N		SD1_D[4]					
R8	PIO4_20	PIO4_20	PIO4_20	DBI_CSX			SD1_D[6]		FC11_SCK		FLEXIO_D 0	
P10	PIO4_21	PIO4_21	PIO4_21	DBI_DSX			SD1_D[7]		FC11_TXD _SCL_MIS O		FLEXIO_D 1	
U10	PIO4_22	PIO4_22	PIO4_22				SD1_CAR D_DET_N		FC11_RXD _SDA_MO SI		FLEXIO_D 2	
T8	PIO4_23	PIO4_23	PIO4_23	DBI_RWD X	LCD_EN ABLE		SD1_RES ET_N		FC11_CTS _SDA_SSE LN0	TRACECL K	FLEXIO_D 3	
T10	PIO4_24	PIO4_24	PIO4_24	DBI_WRX	LCD_DT CLK		SD1_VOL T		FC11_RTS _SCL_SSE LN1	TRACEDA TA[0]	FLEXIO_D 4	

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
T11	PIO4_25	PIO4_25	PIO4_25	DBI_E	LCD_HS YNC				FC11_SSE LN2	TRACEDA TA[1]	FLEXIO_D 5	
T12	PIO4_26	PIO4_26	PIO4_26	LCD_VSY NC					FC11_SSE LN3	TRACEDA TA[2]	FLEXIO_D 6	
P9	PIO4_27	PIO4_27	PIO4_27	LCD_D0	DBI_D0					TRACEDA TA[3]	FLEXIO_D 7	
U8	PIO4_28	PIO4_28	PIO4_28	LCD_D1	DBI_D1						FLEXIO_D 8	
P8	PIO4_29	PIO4_29	PIO4_29	LCD_D2	DBI_D2						FLEXIO_D 9	
N8	PIO4_30	PIO4_30	PIO4_30	LCD_D3	DBI_D3				FC12_TXD _SCL_MIS O		FLEXIO_D 10	
N10	PIO4_31	PIO4_31	PIO4_31	LCD_D4	DBI_D4				FC12_RXD _SDA_MO SI		FLEXIO_D 11	
P12	PIO5_0	PIO5_0	PIO5_0	LCD_D5	DBI_D5				FC12_CTS _SDA_SSE LN0		FLEXIO_D 12	
M9	PIO5_1	PIO5_1	PIO5_1	LCD_D6	DBI_D6				FC12_RTS _SCL_SSE LN1		FLEXIO_D 13	
R9	PIO5_2	PIO5_2	PIO5_2	LCD_D7	DBI_D7				FC12_SSE LN2	LOW_FRE Q_CLKOU T	FLEXIO_D 14	
R10	PIO5_3	PIO5_3	PIO5_3	LCD_D8	DBI_D8				FC12_SSE LN3	LOW_FRE Q_CLKOU T_N	FLEXIO_D 15	

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWL P)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
P2	PIO5_4	PIO5_4	PIO5_4	LCD_D9	DBI_D9		PDM_CLK 01					
P3	PIO5_8	PIO5_8	PIO5_8	LCD_D13	DBI_D13		PDM_DA TA01					
H5	PIO5_15	PIO5_15	PIO5_15	LCD_D20	FLEXSP I1_DATA 4		FC4_CTS _SDA_SS _EL0					
H4	PIO5_16	PIO5_16	PIO5_16	LCD_D21	FLEXSP I1_DATA 5		FC4_RTS _SCL_SS _EL1					
J3	PIO5_17	PIO5_17	PIO5_17	LCD_D22	FLEXSP I1_DATA 6		FC4_SSE L2					
J4	PIO5_18	PIO5_18	PIO5_18	LCD_D23	FLEXSP I1_DATA 7		FC4_SSE L3					
J12, J13, K12, M10, M12	VDDIO_0											
E9, F10, F11, F9, J5 J6	VDDIO_1											
N6, P7	VDDIO_2											
M8 N9	VDDIO_3											
F6, F7	VDDIO_4											
G9, H10, H8, H9, J10, J11, J9, K10, K8, K9, L9	VDDCOR E											
J14	VDD1V8											
H6	VDD1V8											

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWLP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
D4 B2	VDD_AO1 V8											
H13	VDDA_AD C1V8											
E12	VDDA_BI AS											
D11, D7	VSSA											
J8	MIPI_DSI _VDD11											
F8	MIPI_DSI _VDD18											
E6	VDD_PLL 18											
E8	VDD1V8_ 1											
F5	MIPI_DSI _VDDA_C AP											
F4	MIPI_DSI _VSS											
A1, A17, C3, C7, C11, C15, E7, E11, G3, G4, G5, G7, G8, G10, G11, G13, G14, G15, H11, K7, K11, L3, L4, L5, L6, L7, L8, L10, L11, L12, L13,	VSS											

Table continues on the next page...

Table continued from the previous page...

Part Num (249FOWL P)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
L14, L15, M7, M11, N7, N11, P11, R3, R7, R11, R15, U1, U17												
G6, H7, J7	VDD1V8											

12.6 RT500 pin multiplexing table (WLCSP)

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
J1	PIO0_0	PIO0_0	PIO0_0	FC0_SCK			CTIMER0_ MAT0	I2S_BRID GE_CLK_I N	GPIO_INT_ BMAT			
G3	PIO0_1	PIO0_1	PIO0_1	FC0_TXD_ SCL_MISO_ WS			CTIMER0_ MAT1	I2S_BRID GE_WS_I N				
F1	PIO0_10 / TDI	PIO0_10	PIO0_10	FC1_CTS_ SDA_SSE L0	SCT0_G PI7	SCT0_OU T7	CTIMER1_ MAT3	FC0_SSEL 2				
E3	PIO0_11 / TDO	PIO0_11	PIO0_11	FC1_RTS_ SCL_SSEL 1	SCT0_G PI0	SCT0_OU T8	CTIMER_I NP2	FC0_SSEL 3				
G5	PIO0_12 / ADC0_1	PIO0_12	PIO0_12	FC1_SSEL 2	SCT0_G PI2	SCT0_OU T2	CTIMER_I NP3					
J2	PIO0_13 / ADC0_9	PIO0_13	PIO0_13	FC1_SSEL 3	SCT0_G PI3	SCT0_OU T3	CTIMER0_ MAT1					
K6	PIO0_14	PIO0_14	PIO0_14	FC2_SCK	SCT0_G PI0	SCT0_OU T0	CTIMER2_ MAT0	I2S_BRID GE_CLK_I N				
K4	PIO0_15	PIO0_15	PIO0_15	FC2_TXD_ SCL_MISO_ WS	SCT0_G PI1	SCT0_OU T1	CTIMER2_ MAT1	I2S_BRID GE_WS_I N				
M3	PIO0_16	PIO0_16	PIO0_16	FC2_RXD_ SDA_MOS I_DATA	SCT0_G PI2	SCT0_OU T2	CTIMER2_ MAT2	I2S_BRID GE_DATA_ IN			USB0	USB1
M2	PIO0_17	PIO0_17	PIO0_17	FC2_CTS_ SDA_SSE L0	SCT0_G PI3	SCT0_OU T3	CTIMER2_ MAT3	FC5_SSEL 2				
J4	PIO0_18	PIO0_18	PIO0_18	FC2_RTS_ SCL_SSEL 1	SCT0_G PI6	SCT0_OU T6	CTIMER_I NP4	FC5_SSEL 3				

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
H4	PIO0_19 / ADC0_2	PIO0_19	PIO0_19	FC2_SSEL 2	SCT0_G PI4	SCT0_OU T4	CTIMER_I NP5	UTICK_CA P0				
H2	PIO0_2	PIO0_2	PIO0_2	FC0_RXD_ SDA_MOS I_DATA			CTIMER0_ MAT2	I2S_BRID GE_DATA _IN				
M4	PIO0_21	PIO0_21	PIO0_21	FC3_SCK	SCT0_G PI5	SCT0_OU T5	CTIMER3_ MAT0	CTIMER_I NP11	TRACECLK			
J5	PIO0_22	PIO0_22	PIO0_22	FC3_TXD_ SCL_MISO _WS	SCT0_G PI6	SCT0_OU T6	CTIMER3_ MAT1	CTIMER_I NP7	TRACEDAT A[0]			
L5	PIO0_23	PIO0_23	PIO0_23	FC3_RXD_ SDA_MOS I_DATA	SCT0_G PI7	SCT0_OU T8	CTIMER3_ MAT2	CTIMER0_ MAT3	TRACEDAT A[1]			
L4	PIO0_24	PIO0_24	PIO0_24	FC3_CTS_ SDA_SSE L0	SCT0_G PI2	SCT0_OU T9	CTIMER3_ MAT3	FC2_SSEL 2	TRACEDAT A[2]	CLKOUT		
H6	PIO0_25	PIO0_25	PIO0_25	FC3_RTS_ SCL_SSEL 1		FREQME _GPIO_C LK	CTIMER_I NP6	FC2_SSEL 3	TRACEDAT A[3]	CLKIN		
M6	PIO0_28	PIO0_28	PIO0_28	FC4_SCK			CTIMER4_ MAT0	I2S_BRID GE_CLK_ OUT				
J7	PIO0_29	PIO0_29	PIO0_29	FC4_TXD_ SCL_MISO _WS			CTIMER4_ MAT1	I2S_BRID GE_WS_O UT				
G2	PIO0_3	PIO0_3	PIO0_3	FC0_CTS_ SDA_SSE L0			CTIMER0_ MAT3	FC1_SSEL 2				

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
L3	PIO0_30	PIO0_30	PIO0_30	FC4_RXD_ SDA_MOS I_DATA			CTIMER4_ MAT2	I2S_BRID GE_DATA _OUT				
J6	PIO0_31	PIO0_31	PIO0_31	FC4_CTS_ SDA_SSE L0	SCT0_G PI0	SCT0_OU T6	CTIMER4_ MAT3	FC3_SSEL 2				
F4	PIO0_4	PIO0_4	PIO0_4	FC0_RTS_ SCL_SSEL 1			CTIMER_I NP0	FC1_SSEL 3		CMP0_OUT		
H3	PIO0_5/ ADC0_0	PIO0_5	PIO0_5	FC0_SSEL 2	SCT0_G PI0	SCT0_OU T0	CTIMER_I NP1					
K1	PIO0_6/ ADC0_8	PIO0_6	PIO0_6	FC0_SSEL 3	SCT0_G PI1	SCT0_OU T1	CTIMER0_ MAT0					
E4	PIO0_7/ TRST	PIO0_7	PIO0_7	FC1_SCK	SCT0_G PI4	SCT0_OU T4	CTIMER1_ MAT0	I2S_BRID GE_CLK_ OUT				
G1	PIO0_8/ TCK	PIO0_8	PIO0_8	FC1_TXD_ SCL_MISO _WS	SCT0_G PI5	SCT0_OU T5	CTIMER1_ MAT1	I2S_BRID GE_WS_O UT				
F3	PIO0_9/ TMS	PIO0_9	PIO0_9	FC1_RXD_ SDA_MOS I_DATA	SCT0_G PI6	SCT0_OU T6	CTIMER1_ MAT2	I2S_BRID GE_DATA _OUT				
L7	PIO1_0	PIO1_0	PIO1_0	FC4_RTS_ SCL_SSEL 1	SCT0_G PI1	SCT0_OU T7	CTIMER_I NP8	FC3_SSEL 3				
D3	PIO1_11	PIO1_11	PIO1_11	HS_SPI0_ SCK			CTIMER2_ MAT0					
E2	PIO1_12	PIO1_12	PIO1_12	HS_SPI0_ MISO			CTIMER2_ MAT1					

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
D2	PIO1_13	PIO1_13	PIO1_13	HS_SPI0_MOSI			CTIMER2_MAT2					
C2	PIO1_14	PIO1_14	PIO1_14	HS_SPI0_SSELN0			CTIMER2_MAT3					
D1	PIO1_15 / ISP0	PIO1_15	PIO1_15	HS_SPI0_SSELN1			CTIMER3_MAT0					
A5	PIO1_18	PIO1_18	PIO1_18	FLEXSPI0_SCLK	SCT0_GPI0		CTIMER3_MAT3					
D4	PIO1_19	PIO1_19	PIO1_19	FLEXSPI0_SS0_N	SCT0_0UT0		CTIMER4_MAT0			CLKOUT		
B5	PIO1_20	PIO1_20	PIO1_20	FLEXSPI0_DATA0	SCT0_GPI1		CTIMER4_MAT1					
C5	PIO1_21	PIO1_21	PIO1_21	FLEXSPI0_DATA1	SCT0_0UT1		CTIMER4_MAT2					
D5	PIO1_22	PIO1_22	PIO1_22	FLEXSPI0_DATA2	SCT0_GPI2		CTIMER4_MAT3					
C6	PIO1_23	PIO1_23	PIO1_23	FLEXSPI0_DATA3	SCT0_0UT2		CTIMER_I NP8					
A6	PIO1_28	PIO1_28	PIO1_28	FLEXSPI0_DQS	SCT0_GPI5							
F11	PIO1_3	PIO1_3	PIO1_3	FC5_SCK					HS_SPI1_SCK			
F12	PIO1_4	PIO1_4	PIO1_4	FC5_TXD_SCL_MISO_WS					HS_SPI1_MISO			
E12	PIO1_5	PIO1_5	PIO1_5	FC5_RXD_SDA_MOSI_DATA					HS_SPI1_MOSI			

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
E9	PIO1_6	PIO1_6	PIO1_6	FC5_CTS_ SDA_SSE L0	SCT0_G PI4	SCT0_OU T4		FC4_SSEL 2	HS_SPI1_S SELN0			
F9	PIO1_7	PIO1_7	PIO1_7	FC5_RTS_ SCL_SSEL 1	SCT0_G PI5	SCT0_OU T5	CTIMER_I NP9	FC4_SSEL 3	HS_SPI1_S SELN1			
L2	PIO1_9 / ADC0_12 / CMP1_B	PIO1_9	PIO1_9	FC5_SSEL 3	SCT0_G PI7	UTICK_C AP1	CTIMER1_ MAT3		HS_SPI1_S SELN3			
K2	PIO2_14 / CMP0_A	PIO2_14	PIO2_14		SCT0_O UT8		CTIMER_I NP1			32KHZ_CL KOUT		SmartD MA_PIO 14
L1	PIO2_15 / CMP0_D	PIO2_15	PIO2_15		SCT0_O UT9					CLKIN		SmartD MA_PIO 15
E8	PIO2_24	PIO2_24	PIO2_24	SWO					GPIO_INT_ BMAT			SmartD MA_PIO 24
F10	PIO2_25	PIO2_25	PIO2_25	SWCLK								SmartD MA_PIO 25
E11	PIO2_26	PIO2_26	PIO2_26	SWDIO								SmartD MA_PIO 26
E10	PIO2_27	PIO2_27	PIO2_27	USB1_OV ERCURRE NTN								SmartD MA_PIO 27
M7	PIO2_29	PIO2_29	PIO2_29	I3C0_SCL	SCT0_O UT0			CLKOUT				SmartD MA_PIO 29

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
K7	PIO2_30	PIO2_30	PIO2_30	I3C0_SDA	SCT0_O UT3			CLKIN		CMP0_OUT		SmartD MA_PIO 30
K5	PIO2_31 / CMP0_B	PIO2_31	PIO2_31	I3C0_PUR	SCT0_O UT7	UTICK_C AP3	CTIMER_I NP15	SWO				SmartD MA_PIO 31
A4	PIO3_28 / ISP1	PIO3_28	PIO3_28	FC6_CTS_ SDA_SSEL L0								
B4	PIO3_29 / ISP2	PIO3_29	PIO3_29	FC6_RTS_ SCL_SSEL 1								
B8	PIO4_20	PIO4_20	PIO4_20	DBI_CSX			SD1_D[6]		FC11_SCK		FLEXIO_ D0	
B7	PIO4_21	PIO4_21	PIO4_21	DBI_DCX			SD1_D[7]		FC11_TXD_ SCL_MISO		FLEXIO_ D1	
A8	PIO4_22	PIO4_22	PIO4_22				SD1_CAR D_DET_N		FC11_RXD_ SDA_MOSI		FLEXIO_ D2	
E7	PIO4_23	PIO4_23	PIO4_23	DBI_RWD X	LCD_E NABLE		SD1_RES ET_N		FC11_CTS_ SDA_SSEL N0	TRACECLK	FLEXIO_ D3	
C7	PIO4_24	PIO4_24	PIO4_24	DBI_WRX	LCD_DT CLK		SD1_VOL T		FC11_RTS_ SCL_SSEL N1	TRACEDAT A[0]	FLEXIO_ D4	
D7	PIO4_25	PIO4_25	PIO4_25	DBI_E	LCD_H SYNC				FC11_SSEL N2	TRACEDAT A[1]	FLEXIO_ D5	
D6	PIO4_26	PIO4_26	PIO4_26	LCD_VSY NC					FC11_SSEL N3	TRACEDAT A[2]	FLEXIO_ D6	

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
C8	PIO4_27	PIO4_27	PIO4_27	LCD_D0	DBI_D0					TRACEDAT A[3]	FLEXIO_ D7	
B10	PIO4_28	PIO4_28	PIO4_28	LCD_D1	DBI_D1						FLEXIO_ D8	
D8	PIO4_29	PIO4_29	PIO4_29	LCD_D2	DBI_D2						FLEXIO_ D9	
C9	PIO4_30	PIO4_30	PIO4_30	LCD_D3	DBI_D3				FC12_TXD_ SCL_MISO		FLEXIO_ D10	
B9	PIO4_31	PIO4_31	PIO4_31	LCD_D4	DBI_D4				FC12_RXD_ SDA_MOSI		FLEXIO_ D11	
D9	PIO5_4	PIO5_4	PIO5_4	LCD_D9	DBI_D9		PDM_CLK 01					
C11	PIO5_8	PIO5_8	PIO5_8	LCD_D13	DBI_D1 3		PDM_DAT A01					
C4	PIO6_27	PIO6_27	PIO6_27	MCLK								
J12	MIPI_DSI _CLKN	MIPI_DSI _CLKN										
H11	MIPI_DSI _CLKP	MIPI_DSI _CLKP										
H9	MIPI_DSI _D0N	MIPI_DSI _D0N										
H8	MIPI_DSI _D0P	MIPI_DSI _D0P										
H12	MIPI_DSI _VDD11											
G10	MIPI_DSI _VDD18											

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
K12	MIPI_DSI_VDDA_CAP											
J11	MIPI_DSI_VSS											
K9	PMIC_IR_Q_N	PMIC_IR_Q_N										
K10	PMIC_M_ODE0	PMIC_M_ODE0										
J9	PMIC_M_ODE1	PMIC_M_ODE1										
L11	RESETN	RESETN										
J10	RTCXIN	RTCXIN										
K11	RTCXOUT	RTCXOUT										
B11	USB1_DM	USB1_DM										
B12	USB1_DP	USB1_DP										
D10	USB1_VDD3V3	USB1_VDD3V3										
K3	VREFN/VSSA	VREFN										
J3	VREFP/VDDA_A DC1V8/ VDDA_BIAS	VREFP										

Table continues on the next page...

Table continued from the previous page...

Part Num (141WLCSP)	Pin Name	DEFAULT	Func 0	Func 1	Func 2	Func 3	Func 4	Func 5	Func 6	Func 7	Func 8	Func 15
L9	XTALIN	XTALIN										
K8	XTALOU	XTALOU										
A3, B1, B6, E1, F2, G4	VDDIO_0											
D11, G6, H5, L6, M5	VDDIO_1											
NC	VDDIO_2											
E5	VDDIO_3											
NC	VDDIO_4											
A10, B2, C1, C3, C10, C12, E6, F6, G7, G12, M8	VDDCOR											
A11, L12, M11, F5, L8	VDD1V8											
L10	VDD_AO											
M10	VDD1V8_1											
J8	VSSA											
A1, A2, A12, B3, F7, F8, G8, G11, H1, H7, G9, H10, M1, M9, M12	VSS											

Chapter 13

Peripheral Input Mux Controller (INPUTMUX)

13.1 Chip-specific INPUTMUX information

Table 53. Reference links to related information

Topic	Related module	Reference
Full description	INPUTMUX	INPUTMUX
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

13.1.1 Module instances

This device has one instance of the INPUTMUX module, INPUTMUX0.

13.2 Overview

Multiple sources are multiplexed to a selection of peripheral inputs. The sources can be external pins, interrupts, output signals of other peripherals, or other internal signals.

Input multiplexers for most peripherals consist of one or more multiplexers that choose one of several inputs to be routed to a specific input of that peripheral. For example, each CTimer has four capture inputs, each of which has an input multiplexer that can select from among a number of pin functions (if they are connected via IOPCTL) and some internal signals.

The figure below shows a generic input multiplexer arrangement with $n+1$ inputs. For example, in the case of CTimer, there are four muxes, one for each capture input.

Some peripherals have a more complex arrangement and have detailed figures. See [Pin interrupt input multiplexing](#) for Pin interrupt (PINT) multiplexing and [DMA trigger input multiplexing](#) for DMA trigger multiplexing.

13.2.1 Block diagram

The figure below shows the generic input multiplexing.

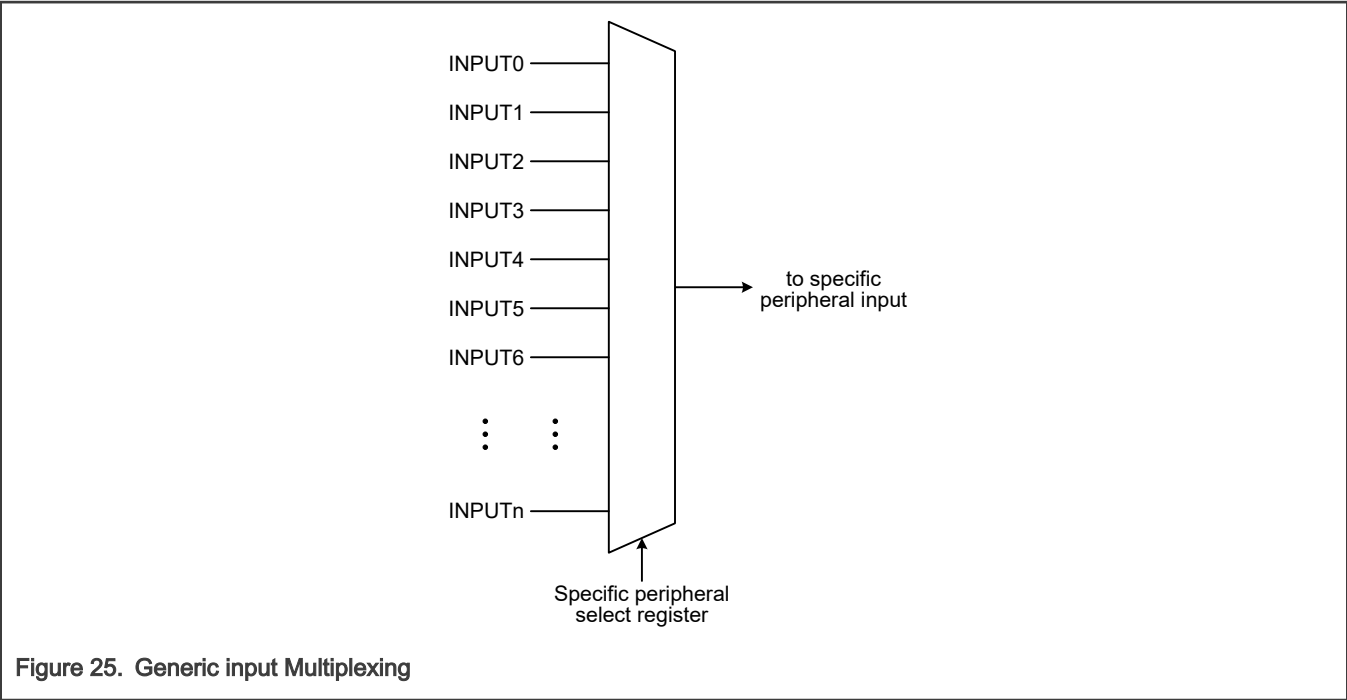


Figure 25. Generic input Multiplexing

13.2.2 Features

The Peripheral input mux controller -

- Configures the inputs to the SCT.
- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures interrupt requests to the DSP.
- Configures the inputs to the DMA0 and DMA1 triggers.
- Configures the DMA0 and DMA1 output triggers to be reused as input triggers.
- Configures the inputs to the asynchronous CTimers.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in the Frequency Measure peripheral.
- Selects DMA requests to go to either DMA0 or DMA1.
- Selects input triggers to go to either DMA0 or DMA1.

13.3 Signals

The INPUTMUX module is not associated with any device pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 54. INPUTMUX pin description

Pins	Peripheral
Any pin on port 0 or 1	Pin interrupts 0 to 7
Any pin with the FREQME_GPIO_CLK function selected via IOPCTL	Frequency measure block

13.3.1 Pin interrupt input multiplexing

The input mux for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

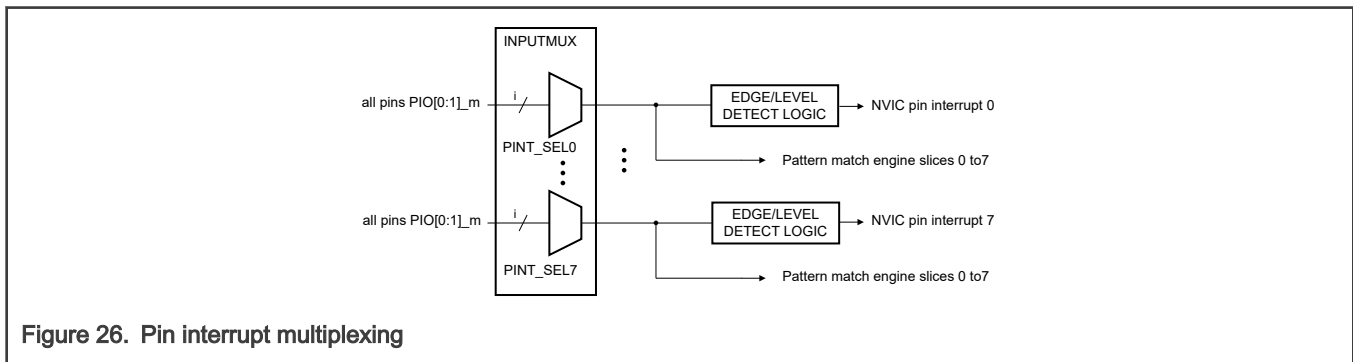


Figure 26. Pin interrupt multiplexing

13.3.2 DMA request configuration

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request.

DMA requests are individually enabled to each DMA controller as shown in the figure below. See [DMAC0 request enable 0 \(DMA0_REQ_ENA0\)](#) and [DMAC1 request enable 0 \(DMA1_REQ_ENA0\)](#).

NOTE

DMA Request inputs do not employ Peripheral Input Muxing. However, DMA trigger inputs do employ Peripheral Input Muxing.

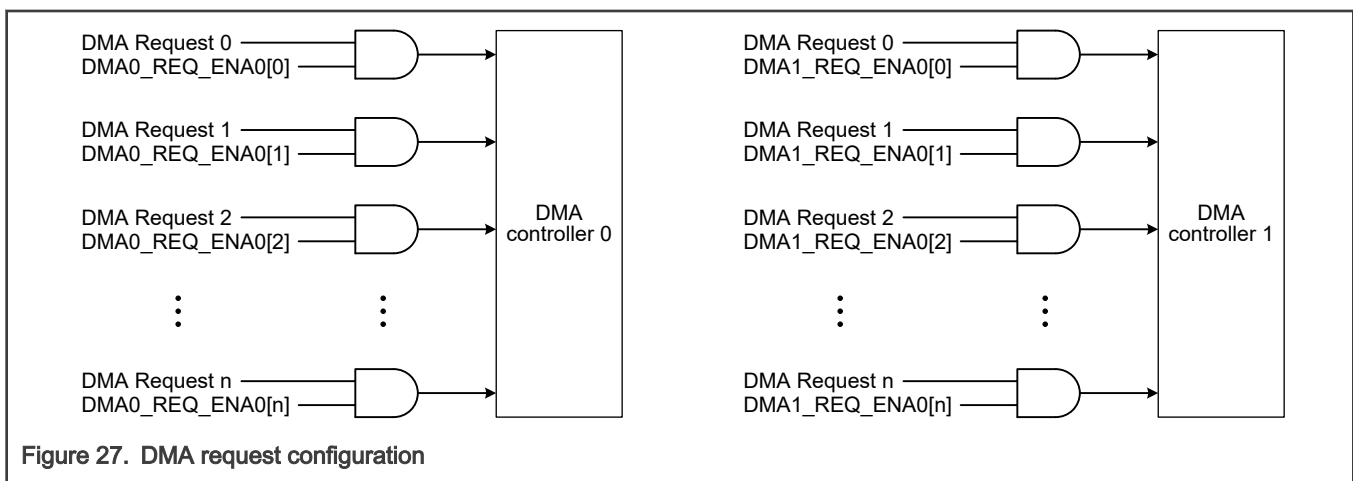


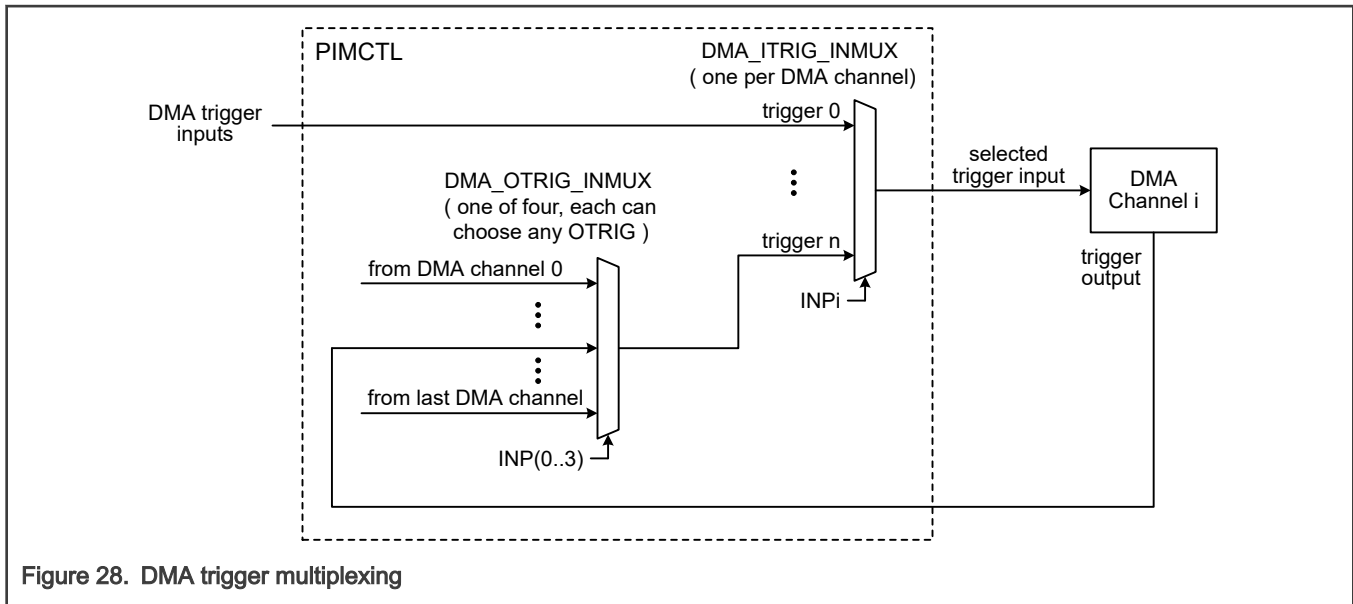
Figure 27. DMA request configuration

13.3.3 DMA trigger input multiplexing

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

The trigger input multiplexing for each DMA controller is configured as shown in the figure below. In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. please refer [DMAC0 Output Trigger Select \(DMAC0_OTRIG_SEL0 - DMAC0_OTRIG_SEL3\)](#) and [DMAC1 Output Trigger Select \(DMAC1_OTRIG_SEL0 - DMAC1_OTRIG_SEL3\)](#).

The potential trigger selections for DMAC0 are shown in [DMAC0 Input Trigger Select \(DMAC0_ITRIG_SEL0 - DMAC0_ITRIG_SEL36\)](#). The potential trigger selections for DMAC1 are shown in [DMAC1 Input Trigger Select \(DMAC1_ITRIG_SEL0 - DMAC1_ITRIG_SEL36\)](#).



13.4 Configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUTMUX registers. Once the input multiplexer is configured, disable the clock to the INPUTMUX block in the CLKCTL1_PSCCTL2 register.

13.5 Memory map and register definition

This section includes the INPUTMUX memory map and detailed descriptions of all registers.

All input mux registers reside on word address boundaries. Details of the registers appear in the description of each function.

13.5.1 INPUTMUX register descriptions

13.5.1.1 INPUTMUX memory map

INPUTMUX base address: 4002_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 18h	SCT Peripheral Input multiplexer a (SCT0_IN0_SEL - SCT0_IN6_SEL)	32	RW	0000_001Fh
100h - 11Ch	GPIO Pin Input Multiplexer a (PINT_SEL0 - PINT_SEL7)	32	RW	0000_001Fh
140h - 1A8h	Fusion DSP Interrupt Input Multiplexer (DSP_INT0_SEL - DSP_INT26_SEL)	32	RW	0000_003Fh
200h - 290h	DMAC0 Input Trigger Select (DMAC0_ITRIG_SEL0 - DMAC0_ITRIG_SEL36)	32	RW	0000_001Fh
300h - 30Ch	DMAC0 Output Trigger Select (DMAC0_OTRIG_SEL0 - DMAC0_OTRIG_SEL3)	32	RW	0000_001Fh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
320h	DMAC0 Channel mux select 0 (DMAC0_CHMUX_SEL0)	32	RW	0000_0000h
324h	DMAC0 Channel mux select 1 (DMAC0_CHMUX_SEL1)	32	RW	0000_0000h
328h	DMAC0 Channel mux select 2 (DMAC0_CHMUX_SEL2)	32	RW	0000_0000h
32Ch	DMAC0 Channel mux select 3 (DMAC0_CHMUX_SEL3)	32	RW	0000_0000h
330h	DMAC0 Channel mux select 4 (DMAC0_CHMUX_SEL4)	32	RW	0000_0000h
334h	DMAC0 Channel mux select 5 (DMAC0_CHMUX_SEL5)	32	RW	0000_0000h
338h	DMAC0 Channel mux select 6 (DMAC0_CHMUX_SEL6)	32	RW	0000_0000h
33Ch	DMAC0 Channel mux select 7 (DMAC0_CHMUX_SEL7)	32	RW	0000_0000h
340h	DMAC0 Channel mux select 8 (DMAC0_CHMUX_SEL8)	32	RW	0000_0000h
344h	DMAC0 Channel mux select 9 (DMAC0_CHMUX_SEL9)	32	RW	0000_0000h
348h	DMAC0 Channel mux select 10 (DMAC0_CHMUX_SEL10)	32	RW	0000_0000h
34Ch	DMAC0 Channel mux select 11 (DMAC0_CHMUX_SEL11)	32	RW	0000_0000h
350h	DMAC0 Channel mux select 12 (DMAC0_CHMUX_SEL12)	32	RW	0000_0000h
354h	DMAC0 Channel mux select 13 (DMAC0_CHMUX_SEL13)	32	RW	0000_0000h
358h	DMAC0 Channel mux select 14 (DMAC0_CHMUX_SEL14)	32	RW	0000_0000h
35Ch	DMAC0 Channel mux select 15 (DMAC0_CHMUX_SEL15)	32	RW	0000_0000h
400h - 490h	DMAC1 Input Trigger Select (DMAC1_ITRIG_SEL0 - DMAC1_ITRIG_SEL36)	32	RW	0000_001Fh
500h - 50Ch	DMAC1 Output Trigger Select (DMAC1_OTRIG_SEL0 - DMAC1_OTRIG_SEL3)	32	RW	0000_003Fh
520h	DMAC1 Channel mux select 0 (DMAC1_CHMUX_SEL0)	32	RW	0000_0000h
524h	DMAC1 Channel mux select 1 (DMAC1_CHMUX_SEL1)	32	RW	0000_0000h
528h	DMAC1 Channel mux select 2 (DMAC1_CHMUX_SEL2)	32	RW	0000_0000h
52Ch	DMAC1 Channel mux select 3 (DMAC1_CHMUX_SEL3)	32	RW	0000_0000h
530h	DMAC1 Channel mux select 4 (DMAC1_CHMUX_SEL4)	32	RW	0000_0000h
534h	DMAC1 Channel mux select 5 (DMAC1_CHMUX_SEL5)	32	RW	0000_0000h
538h	DMAC1 Channel mux select 6 (DMAC1_CHMUX_SEL6)	32	RW	0000_0000h
53Ch	DMAC1 Channel mux select 7 (DMAC1_CHMUX_SEL7)	32	RW	0000_0000h
540h	DMAC1 Channel mux select 8 (DMAC1_CHMUX_SEL8)	32	RW	0000_0000h
544h	DMAC1 Channel mux select 9 (DMAC1_CHMUX_SEL9)	32	RW	0000_0000h
548h	DMAC1 Channel mux select 10 (DMAC1_CHMUX_SEL10)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
54Ch	DMAC1 Channel mux select 11 (DMAC1_CHMUX_SEL11)	32	RW	0000_0000h
550h	DMAC1 Channel mux select 12 (DMAC1_CHMUX_SEL12)	32	RW	0000_0000h
554h	DMAC1 Channel mux select 13 (DMAC1_CHMUX_SEL13)	32	RW	0000_0000h
558h	DMAC1 Channel mux select 14 (DMAC1_CHMUX_SEL14)	32	RW	0000_0000h
55Ch	DMAC1 Channel mux select 15 (DMAC1_CHMUX_SEL15)	32	RW	0000_0000h
600h - 64Ch	CT32BIT Timer Capture Multiplexers (CT32BIT0_CAP0_SEL - CT32BIT4_CAP3_SEL)	32	RW	0000_001Fh
700h - 704h	Frequency Measurement Input Channel Multiplexers (FMEASURE_CH0_SEL - FMEASURE_CH1_SEL)	32	RW	0000_001Fh
720h - 73Ch	SMART_DMA trigger channel select (SMART_DMA_TRIG_CH0_SEL - SMART_DMA_TRIG_CH7_SEL)	32	RW	0000_007Fh
740h	DMAC0 request enable 0 (DMAC0_REQ_ENA0)	32	RW	FFFF_FFFFh
744h	DMAC0 request enable 1 (DMAC0_REQ_ENA1)	32	RW	FFFF_FFFFh
748h	DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET)	32	RW	See section
74Ch	DMAC0 request enable set 1 (DMAC0_REQ_ENA1_SET)	32	RW	See section
750h	DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR)	32	RW	See section
754h	DMAC0 request enable 1 clear (DMAC0_REQ_ENA1_CLR)	32	RW	See section
760h	DMAC1 request enable 0 (DMAC1_REQ_ENA0)	32	RW	FFFF_FFFFh
764h	DMAC1 request enable 1 (DMAC1_REQ_ENA1)	32	RW	FFFF_FFFFh
768h	DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET)	32	RW	See section
76Ch	DMAC1 request enable set 1 (DMAC1_REQ_ENA1_SET)	32	RW	See section
770h	DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR)	32	RW	See section
774h	DMAC1 request enable 1 clear (DMAC1_REQ_ENA1_CLR)	32	RW	See section
780h	DMAC0 Input Trigger Enable 0 (DMAC0_ITRIG_ENA0)	32	RW	FFFF_FFFFh
788h	DMAC0 Input Trigger Enable 0 Set (DMAC0_ITRIG_ENA0_SET)	32	W	See section
790h	DMAC0 Input Trigger Enable 0 Clear (DMAC0_ITRIG_ENA0_CLR)	32	W	See section
7A0h	DMAC1 Input Trigger Enable 0 (DMAC1_ITRIG_ENA0)	32	RW	FFFF_FFFFh
7A8h	DMAC1 Input Trigger Enable 0 set (DMAC1_ITRIG_ENA0_SET)	32	W	See section
7B0h	DMAC1 Input Trigger Enable 0 clear (DMAC1_ITRIG_ENA0_CLR)	32	W	See section

13.5.1.2 SCT Peripheral Input multiplexer a (SCT0_IN0_SEL - SCT0_IN6_SEL)

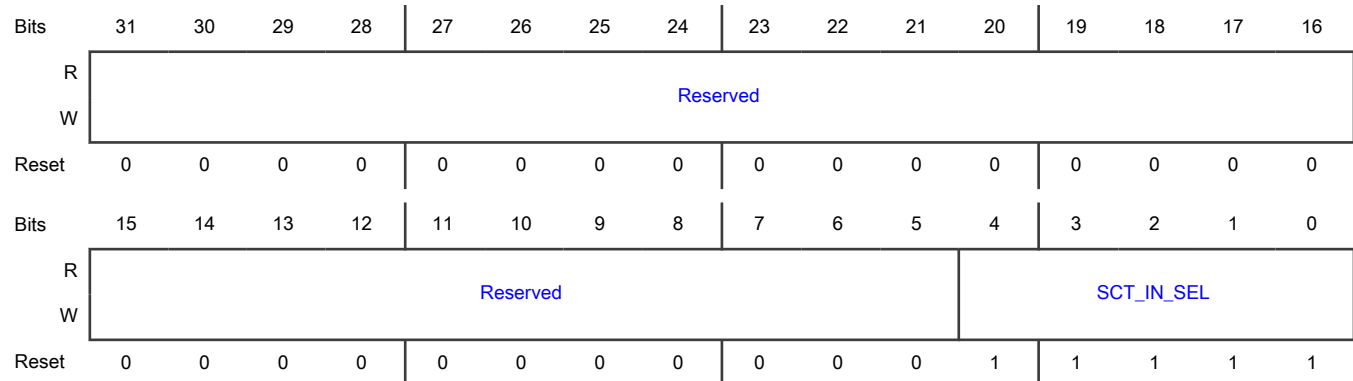
Offset

Register	Offset
SCT0_IN0_SEL	0h
SCT0_IN1_SEL	4h
SCT0_IN2_SEL	8h
SCT0_IN3_SEL	Ch
SCT0_IN4_SEL	10h
SCT0_IN5_SEL	14h
SCT0_IN6_SEL	18h

Function

The SCT (Counter Timer) has a provision for 7 channel inputs. The source of each channel input can be selected as defined below.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 SCT_IN_SEL	SCT0 Input Selection. 0_0000b - SCT0_PIN_INP0 0_0001b - SCT0_PIN_INP1 0_0010b - SCT0_PIN_INP2 0_0011b - SCT0_PIN_INP3

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0100b - SCT0_PIN_INP4
	0_0101b - SCT0_PIN_INP5
	0_0110b - SCT0_PIN_INP6
	0_0111b - SCT0_PIN_INP7
	0_1000b - CT32BIT0_MAT0
	0_1001b - CT32BIT1_MAT0
	0_1010b - CT32BIT2_MAT0
	0_1011b - CT32BIT3_MAT0
	0_1100b - CT32BIT4_MAT0
	0_1101b - ADCIRQ
	0_1110b - GPIOINT_BMATCH
	0_1111b - USB0_FRAME_TOGGLE
	1_0000b - CMP0_OUT
	1_0001b - SHARED I2S0_SCLK
	1_0010b - SHARED I2S1_SCLK
	1_0011b - SHARED I2S0_WS
	1_0100b - SHARED I2S1_WS
	1_0101b - MCLK
	1_0110b - ARM_TXEV
	1_0111b - DEBUG_HALTED
	1_1000b - Reserved
	1_1001b - Reserved
	1_1010b - Reserved
	1_1011b - Reserved
	1_1100b - Reserved
	1_1101b - Reserved
	1_1110b - Reserved
	1_1111b - Reserved

13.5.1.3 GPIO Pin Input Multiplexer a (PINT_SEL0 - PINT_SEL7)

Offset

For a = 0 to 7:

Register	Offset
PINT_SEL _a	100h + (a × 4h)

Function

The pin interrupt and pattern match engine have the provision for 8 channel inputs. The source of each channel input is selected as defined below. Each of these 8 registers selects one pin as the source of a pin interrupt or as the input to the pattern match engine. Each of the pin interrupts must be enabled in the NVIC before it becomes active.

Diagram

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

Fields

Field	Function
31-8 —	Reserved
7-0 PINT_SEL	Interrupt select Pin number select for pin interrupt or pattern match engine input. PIO0_0 to PIO1_31 correspond to numbers 0 to 63. For PIO _x _y : INTPIN = (x * 32) + y).

13.5.1.4 Fusion DSP Interrupt Input Multiplexer (DSP_INT0_SEL - DSP_INT26_SEL)

Offset

For a = 0 to 26:

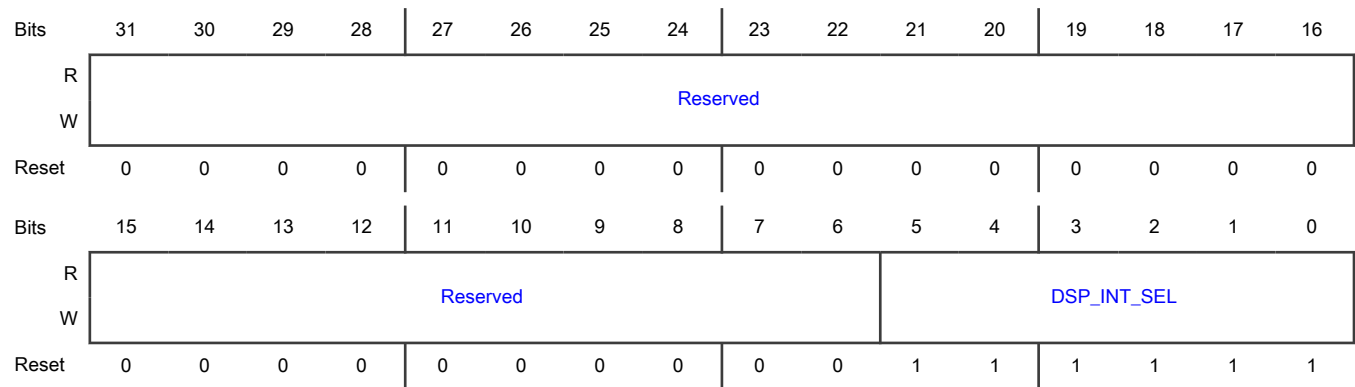
Register	Offset
DSP_INT _a _SEL	140h + (a × 4h)

Function

The source of each channel input is selected as defined below.

NOTE

Fusion DSP inputs have a fixed priority level based on interrupt slot number so the selection of interrupt sources are routed to specific Interrupts is important.

Diagram**Fields**

Field	Function
31-6 —	Reserved
5-0 DSP_INT_SEL	Fusion DSP Input(n) Selection. 00_0000b - FLEXCOMM0_IRQ 00_0001b - FLEXCOMM1_IRQ 00_0010b - FLEXCOMM2_IRQ 00_0011b - FLEXCOMM3_IRQ 00_0100b - FLEXCOMM4_IRQ 00_0101b - FLEXCOMM5_IRQ 00_0110b - FLEXCOMM6_IRQ 00_0111b - FLEXCOMM7_IRQ 00_1000b - FLEXCOMM14_IRQ 00_1001b - FLEXCOMM16_IRQ 00_1010b - GPIO_INT0_IRQ0 00_1011b - GPIO_INT0_IRQ1 00_1100b - GPIO_INT0_IRQ2 00_1101b - GPIO_INT0_IRQ3 00_1110b - GPIO_INT0_IRQ4 00_1111b - GPIO_INT0_IRQ5 01_0000b - GPIO_INT0_IRQ6 01_0001b - GPIO_INT0_IRQ7 01_0010b - NSHSGPIO_INT0_IRQ0 01_0011b - NSHSGPIO_INT1_IRQ1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_0100b - WDT1
	01_0101b - DMAC0_IRQ
	01_0110b - DMAC1_IRQ
	01_0111b - MU_B_IRQ
	01_1000b - UTICK0_IRQ
	01_1001b - MRT0_IRQ
	01_1010b - OS_EVENT_TIMER or OS_EVENT_WAKEUP
	01_1011b - CTIMER0
	01_1100b - CTIMER1
	01_1101b - CTIMER2
	01_1110b - CTIMER3
	01_1111b - CTIMER4
	10_0000b - RTC_LITE0_ALARM or RTC_LITE0_WAKEUP
	10_0001b - I3C0
	10_0010b - I3C1
	10_0011b - DMIC0
	10_0100b - HWVAD
	10_0101b - LCDIF_IRQ
	10_0110b - GPU_IRQ
	10_0111b - SMARTDMA_IRQ
	10_1000b - FLEXIO_IRQ

13.5.1.5 DMAC0 Input Trigger Select (DMAC0_ITRIG_SEL0 - DMAC0_ITRIG_SEL36)

Offset

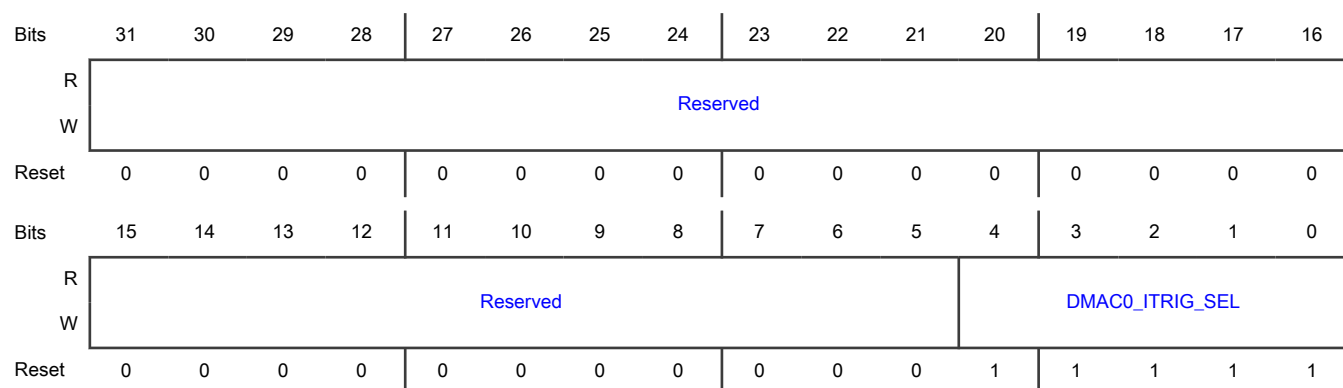
For a = 0 to 36:

Register	Offset
DMAC0_ITRIG_SEL _a	200h + (a × 4h)

Function

DMA trigger input mux registers allow selecting one trigger input each of DMA channel from among the potential internal sources. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 DMAC0_ITRIG_SEL	DMA Input Trigger Selection. 0_0000b - GPIO_INT0 0_0001b - GPIO_INT1 0_0010b - GPIO_INT2 0_0011b - GPIO_INT3 0_0100b - T0_DMAREQ_M0 0_0101b - T0_DMAREQ_M1 0_0110b - T1_DMAREQ_M0 0_0111b - T1_DMAREQ_M1 0_1000b - T2_DMAREQ_M0 0_1001b - T2_DMAREQ_M1 0_1010b - T3_DMAREQ_M0 0_1011b - T3_DMAREQ_M1 0_1100b - T4_DMAREQ_M0 0_1101b - T4_DMAREQ_M1 0_1110b - DMA0_TRIGOUT_A 0_1111b - DMA0_TRIGOUT_B 1_0000b - DMA0_TRIGOUT_C 1_0001b - DMA0_TRIGOUT_D 1_0010b - SCT_DMAC0_REQ0 1_0011b - SCT_DMAC1_REQ1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1_0100b - HASHCRYPT_OUT_DMA
	1_0101b - ACMP_DMA
	1_0110b - FlexSPI0_RX_DMA
	1_0111b - FlexSPI0_TX_DMA
	1_1000b - ADC_DMA
	1_1001b - FlexSPI1_RX_DMA
	1_1010b - FlexSPI1_TX_DMA
	1_1011b - Reserved
	1_1100b - Reserved
	1_1101b - Reserved
	1_1111b - Reserved

13.5.1.6 DMAC0 Output Trigger Select (DMAC0_OTRIG_SEL0 - DMAC0_OTRIG_SEL3)

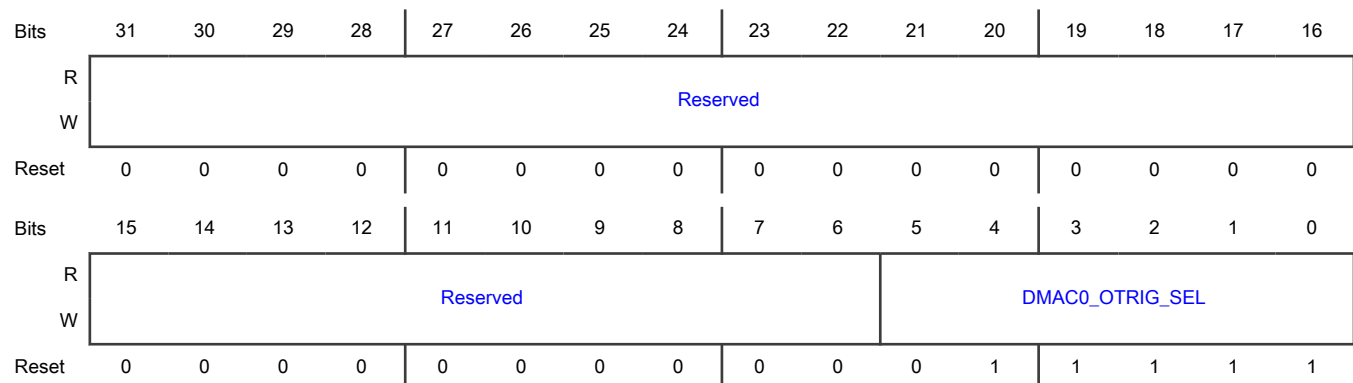
Offset

Register	Offset
DMAC0_OTRIG_SEL0	300h
DMAC0_OTRIG_SEL1	304h
DMAC0_OTRIG_SEL2	308h
DMAC0_OTRIG_SEL3	30Ch

Function

This register provides a multiplexer for the four inputs (DMAC0_TRIGOUT_A through DMAC0_TRIGOUT_D) of each DMA trigger input mux. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 DMAC0_OTRIG_SEL	DMAC0 Output Triggers Select for A, B, C, D IE., DMAC0_OTRIG_A, DMAC0_OTRIG_B, DMAC0_OTRIG_C, DMAC0_OTRIG_D DMA0 Output Triggers 00_0000b - DMAC0_OTRIG_CH0 00_0001b - DMAC0_OTRIG_CH1 00_0010b - DMAC0_OTRIG_CH2 00_0011b - DMAC0_OTRIG_CH3 00_0100b - DMAC0_OTRIG_CH4 00_0101b - DMAC0_OTRIG_CH5 00_0110b - DMAC0_OTRIG_CH6 00_0111b - DMAC0_OTRIG_CH7 00_1000b - DMAC0_OTRIG_CH8 00_1001b - DMAC0_OTRIG_CH9 00_1010b - DMAC0_OTRIG_CH10 00_1011b - DMAC0_OTRIG_CH11 00_1100b - DMAC0_OTRIG_CH12 00_1101b - DMAC0_OTRIG_CH13 00_1110b - DMAC0_OTRIG_CH14 00_1111b - DMAC0_OTRIG_CH15 01_0000b - DMAC0_OTRIG_CH16 01_0001b - DMAC0_OTRIG_CH17 01_0010b - DMAC0_OTRIG_CH18 01_0011b - DMAC0_OTRIG_CH19 01_0100b - DMAC0_OTRIG_CH20 01_0101b - DMAC0_OTRIG_CH21 01_0110b - DMAC0_OTRIG_CH22 01_0111b - DMAC0_OTRIG_CH23 01_1000b - DMAC0_OTRIG_CH24 01_1001b - DMAC0_OTRIG_CH25 01_1010b - DMAC0_OTRIG_CH26 01_1011b - DMAC0_OTRIG_CH27 01_1100b - DMAC0_OTRIG_CH28

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1101b - DMAC0_OTRIG_CH29
	01_1110b - DMAC0_OTRIG_CH30
	01_1111b - DMAC0_OTRIG_CH31
	10_0000b - DMAC0_OTRIG_CH32
	10_0001b - DMAC0_OTRIG_CH33
	10_0010b - DMAC0_OTRIG_CH34
	10_0011b - DMAC0_OTRIG_CH35
	10_0100b - DMAC0_OTRIG_CH36

13.5.1.7 DMAC0 Channel mux select 0 (DMAC0_CHMUX_SELO)

Offset

Register	Offset
DMAC0_CHMUX_SELO	320h

Function

DMAC0 Channel mux select 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0 0b - DMIC_CH0_DMA 1b - FLEXCOM8_RX_DMA

13.5.1.8 DMAC0 Channel mux select 1 (DMAC0_CHMUX_SEL1)

Offset

Register	Offset
DMAC0_CHMUX_SEL1	324h

Function

DMAC0 Channel mux select 1

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 1 0b - DMIC_CH1_DMA 1b - FLEXCOM8_TX_DMA

13.5.1.9 DMAC0 Channel mux select 2 (DMAC0_CHMUX_SEL2)

Offset

Register	Offset
DMAC0_CHMUX_SEL2	328h

Function

DMAC0 Channel mux select 2

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMUX_SEL	DMAC0 Channel mux select 0b - DMIC_CH2_DMA 1b - FLEXCOM9_RX_DMA

13.5.1.10 DMAC0 Channel mux select 3 (DMAC0_CHMUX_SEL3)

Offset

Register	Offset
DMAC0_CHMUX_SEL3	32Ch

Function

DMAC0 Channel mux select 3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - DMIC_CH3_DMA 1b - FLEXCOM9_TX_DMA

13.5.1.11 DMAC0 Channel mux select 4 (DMAC0_CHMUX_SEL4)**Offset**

Register	Offset
DMAC0_CHMUX_SEL4	330h

Function

DMAC0 Channel mux select 4

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - DMIC_CH4_DMA 1b - FLEXCOM10_RX_DMA

13.5.1.12 DMAC0 Channel mux select 5 (DMAC0_CHMUX_SEL5)

Offset

Register	Offset
DMAC0_CHMUX_SEL5	334h

Function

DMAC0 Channel mux select 5

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - DMIC_CH5_DMA 1b - FLEXCOM10_TX_DMA

13.5.1.13 DMAC0 Channel mux select 6 (DMAC0_CHMUX_SEL6)

Offset

Register	Offset
DMAC0_CHMUX_SEL6	338h

Function

DMAC0 Channel mux select 6

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMUX_SEL	DMAC0 Channel mux select 0b - DMIC_CH6_DMA 1b - FLEXCOM13_RX_DMA

13.5.1.14 DMAC0 Channel mux select 7 (DMAC0_CHMUX_SEL7)

Offset

Register	Offset
DMAC0_CHMUX_SEL7	33Ch

Function

DMAC0 Channel mux select 7

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - DMIC_CH7_DMA 1b - FLEXCOM13_TX_DMA

13.5.1.15 DMAC0 Channel mux select 8 (DMAC0_CHMUX_SEL8)**Offset**

Register	Offset
DMAC0_CHMUX_SEL8	340h

Function

DMAC0 Channel mux select 8

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - FLEXCOM16_RX_DMA 1b - Reserved

13.5.1.16 DMAC0 Channel mux select 9 (DMAC0_CHMUX_SEL9)

Offset

Register	Offset
DMAC0_CHMUX_SEL9	344h

Function

DMAC0 Channel mux select 9

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - FLEXCOM16_TX_DMA 1b - Reserved

13.5.1.17 DMAC0 Channel mux select 10 (DMAC0_CHMUX_SEL10)

Offset

Register	Offset
DMAC0_CHMUX_SEL10	348h

Function

DMAC0 Channel mux select 10

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMUX_SEL	DMAC0 Channel mux select 0b - I3C1_RX_DMA 1b - Reserved

13.5.1.18 DMAC0 Channel mux select 11 (DMAC0_CHMUX_SEL11)

Offset

Register	Offset
DMAC0_CHMUX_SEL11	34Ch

Function

DMAC0 Channel mux select 11

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - I3C1_TX_DMA 1b - Reserved

13.5.1.19 DMAC0 Channel mux select 12 (DMAC0_CHMUX_SEL12)**Offset**

Register	Offset
DMAC0_CHMUX_SEL12	350h

Function

DMAC0 Channel mux select 12

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - Reserved 1b - FLEXCOM11_RX_DMA

13.5.1.20 DMAC0 Channel mux select 13 (DMAC0_CHMUX_SEL13)

Offset

Register	Offset
DMAC0_CHMUX_SEL13	354h

Function

DMAC0 Channel mux select 13

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMAC0 Channel mux select 0b - Reserved 1b - FLEXCOM11_TX_DMA

13.5.1.21 DMAC0 Channel mux select 14 (DMAC0_CHMUX_SEL14)

Offset

Register	Offset
DMAC0_CHMUX_SEL14	358h

Function

DMAC0 Channel mux select 14

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																0_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMUX_SEL	DMAC0 Channel mux select 0b - Reserved 1b - FLEXCOM12_RX_DMA

13.5.1.22 DMAC0 Channel mux select 15 (DMAC0_CHMUX_SEL15)

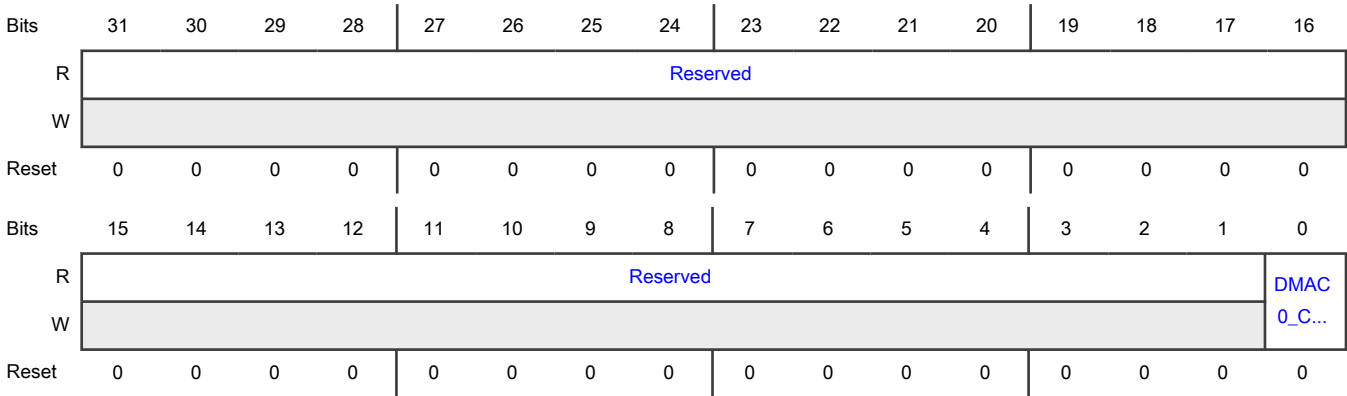
Offset

Register	Offset
DMAC0_CHMUX_SEL15	35Ch

Function

DMAC0 Channel mux select 15

Diagram



Fields

Field	Function
31-1 —	Reserved
0 DMAC0_CHMU X_SEL	DMA Channel mux select 0b - Reserved 1b - FLEXCOM12_TX_DMA

13.5.1.23 DMAC1 Input Trigger Select (DMAC1_ITRIG_SEL0 - DMAC1_ITRIG_SEL36)

Offset

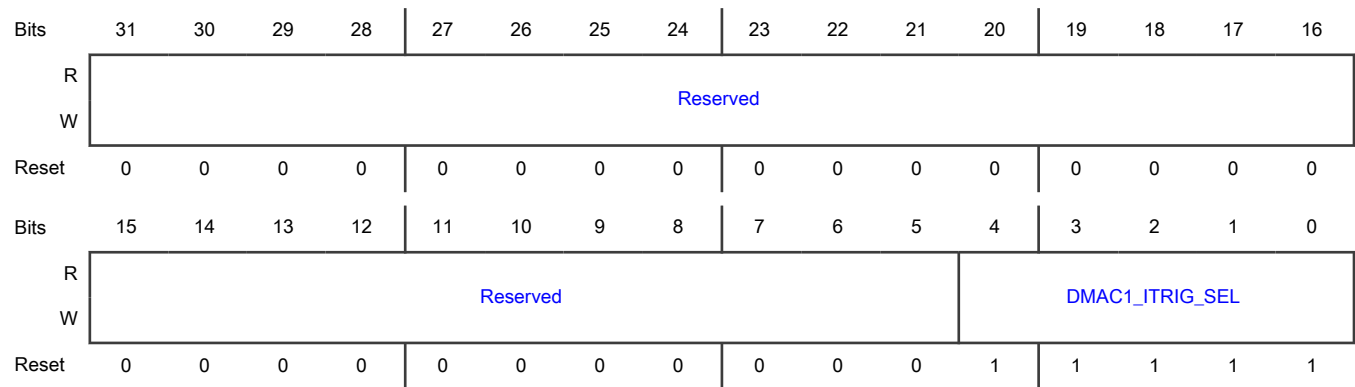
For a = 0 to 36:

Register	Offset
DMAC1_ITRIG_SEL _a	400h + (a × 4h)

Function

DMA trigger input mux registers allow selecting one trigger input each of DMA channel from among the potential internal sources. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 DMAC1_ITRIG_SEL	DMA Input Trigger Selection. 0_0000b - GPIO_INT0 0_0001b - GPIO_INT1 0_0010b - GPIO_INT2 0_0011b - GPIO_INT3 0_0100b - T0_DMAREQ_M0 0_0101b - T0_DMAREQ_M1 0_0110b - T1_DMAREQ_M0 0_0111b - T1_DMAREQ_M1 0_1000b - T2_DMAREQ_M0 0_1001b - T2_DMAREQ_M1 0_1010b - T3_DMAREQ_M0 0_1011b - T3_DMAREQ_M1 0_1100b - T4_DMAREQ_M0 0_1101b - T4_DMAREQ_M1 0_1110b - DMA0_TRIGOUT_A 0_1111b - DMA0_TRIGOUT_B 1_0000b - DMA0_TRIGOUT_C 1_0001b - DMA0_TRIGOUT_D 1_0010b - SCT_DMAC0_REQ0 1_0011b - SCT_DMAC1_REQ1

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1_0100b - HASHCRYPT_OUT_DMA
	1_0101b - ACMP_DMA
	1_0110b - FlexSPI0_RX_DMA
	1_0111b - FlexSPI0_TX_DMA
	1_1000b - ADC_DMA
	1_1001b - FlexSPI1_RX_DMA
	1_1010b - FlexSPI1_TX_DMA
	1_1011b - Reserved
	1_1100b - Reserved
	1_1101b - Reserved
	1_1111b - Reserved

13.5.1.24 DMAC1 Output Trigger Select (DMAC1_OTRIG_SEL0 - DMAC1_OTRIG_SEL3)

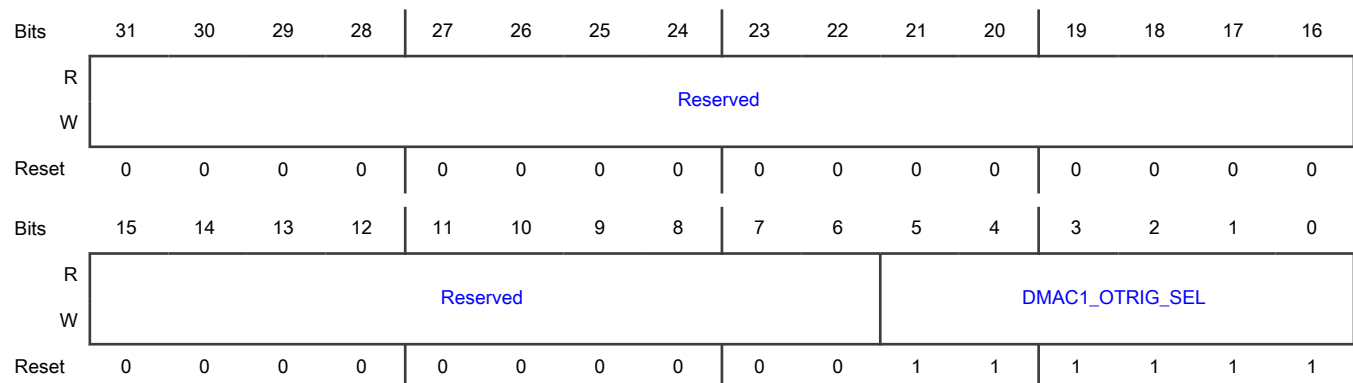
Offset

Register	Offset
DMAC1_OTRIG_SEL0	500h
DMAC1_OTRIG_SEL1	504h
DMAC1_OTRIG_SEL2	508h
DMAC1_OTRIG_SEL3	50Ch

Function

This register provides a multiplexer for the four inputs (DMAC1_TRIGOUT_A through DMAC1_TRIGOUT_D) of each DMA trigger input mux. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 DMAC1_OTRIG_SEL	DMA Output Trigger Selection. DMAC1 Output Triggers Select for A, B, C, D IE., DMAC1_OTRIG_A, DMAC1_OTRIG_B, DMAC1_OTRIG_C, DMAC1_OTRIG_D DMA0 Output Triggers 00_0000b - DMAC1_OTRIG_CH0 00_0001b - DMAC1_OTRIG_CH1 00_0010b - DMAC1_OTRIG_CH2 00_0011b - DMAC1_OTRIG_CH3 00_0100b - DMAC1_OTRIG_CH4 00_0101b - DMAC1_OTRIG_CH5 00_0110b - DMAC1_OTRIG_CH6 00_0111b - DMAC1_OTRIG_CH7 00_1000b - DMAC1_OTRIG_CH8 00_1001b - DMAC1_OTRIG_CH9 00_1010b - DMAC1_OTRIG_CH10 00_1011b - DMAC1_OTRIG_CH11 00_1100b - DMAC1_OTRIG_CH12 00_1101b - DMAC1_OTRIG_CH13 00_1110b - DMAC1_OTRIG_CH14 00_1111b - DMAC1_OTRIG_CH15 01_0000b - DMAC1_OTRIG_CH16 01_0001b - DMAC1_OTRIG_CH17 01_0010b - DMAC1_OTRIG_CH18 01_0011b - DMAC1_OTRIG_CH19 01_0100b - DMAC1_OTRIG_CH20 01_0101b - DMAC1_OTRIG_CH21 01_0110b - DMAC1_OTRIG_CH22 01_0111b - DMAC1_OTRIG_CH23 01_1000b - DMAC1_OTRIG_CH24 01_1001b - DMAC1_OTRIG_CH25 01_1010b - DMAC1_OTRIG_CH26 01_1011b - DMAC1_OTRIG_CH27

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01_1100b - DMAC1_OTRIG_CH28
	01_1101b - DMAC1_OTRIG_CH29
	01_1110b - DMAC1_OTRIG_CH30
	01_1111b - DMAC1_OTRIG_CH31
	10_0000b - DMAC1_OTRIG_CH32
	10_0001b - DMAC1_OTRIG_CH33
	10_0010b - DMAC1_OTRIG_CH34
	10_0011b - DMAC1_OTRIG_CH35
	10_0100b - DMAC1_OTRIG_CH36

13.5.1.25 DMAC1 Channel mux select 0 (DMAC1_CHMUX_SELO)

Offset

Register	Offset
DMAC1_CHMUX_SELO	520h

Function

DMAC1 Channel mux select 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 DMAC1_CHMUX_X_SEL	DMAC1 Channel mux select 0b - DMIC_CH0_DMA 1b - FLEXCOM8_RX_DMA

13.5.1.26 DMAC1 Channel mux select 1 (DMAC1_CHMUX_SEL1)

Offset

Register	Offset
DMAC1_CHMUX_SEL1	524h

Function

DMAC1 Channel mux select 1

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC 1_C...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMUX_X_SEL	DMAC1 Channel mux select 0b - DMIC_CH1_DMA 1b - FLEXCOM8_TX_DMA

13.5.1.27 DMAC1 Channel mux select 2 (DMAC1_CHMUX_SEL2)

Offset

Register	Offset
DMAC1_CHMUX_SEL2	528h

Function

DMAC1 Channel mux select 2

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC 1_C...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMUX_SEL	DMAC1 Channel mux select 0b - DMIC_CH2_DMA 1b - FLEXCOM9_RX_DMA

13.5.1.28 DMAC1 Channel mux select 3 (DMAC1_CHMUX_SEL3)

Offset

Register	Offset
DMAC1_CHMUX_SEL3	52Ch

Function

DMAC1 Channel mux select 3

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - DMIC_CH3_DMA 1b - FLEXCOM9_TX_DMA

13.5.1.29 DMAC1 Channel mux select 4 (DMAC1_CHMUX_SEL4)**Offset**

Register	Offset
DMAC1_CHMUX_SEL4	530h

Function

DMAC1 Channel mux select 4

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - DMIC_CH4_DMA 1b - FLEXCOM10_RX_DMA

13.5.1.30 DMAC1 Channel mux select 5 (DMAC1_CHMUX_SEL5)

Offset

Register	Offset
DMAC1_CHMUX_SEL5	534h

Function

DMAC1 Channel mux select 5

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select DMAC1 Channel mux select 0b - DMIC_CH5_DMA 1b - FLEXCOM10_TX_DMA

13.5.1.31 DMAC1 Channel mux select 6 (DMAC1_CHMUX_SEL6)

Offset

Register	Offset
DMAC1_CHMUX_SEL6	538h

Function

DMAC1 Channel mux select 6

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMUX_SEL	DMAC1 Channel mux select 0b - DMIC_CH6_DMA 1b - FLEXCOM13_RX_DMA

13.5.1.32 DMAC1 Channel mux select 7 (DMAC1_CHMUX_SEL7)

Offset

Register	Offset
DMAC1_CHMUX_SEL7	53Ch

Function

DMAC1 Channel mux select 7

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - DMIC_CH7_DMA 1b - FLEXCOM13_TX_DMA

13.5.1.33 DMAC1 Channel mux select 8 (DMAC1_CHMUX_SEL8)**Offset**

Register	Offset
DMAC1_CHMUX_SEL8	540h

Function

DMAC1 Channel mux select 8

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - FLEXCOM16_RX_DMA 1b - Reserved

13.5.1.34 DMAC1 Channel mux select 9 (DMAC1_CHMUX_SEL9)

Offset

Register	Offset
DMAC1_CHMUX_SEL9	544h

Function

DMAC1 Channel mux select 9

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - FLEXCOM16_TX_DMA 1b - Reserved

13.5.1.35 DMAC1 Channel mux select 10 (DMAC1_CHMUX_SEL10)

Offset

Register	Offset
DMAC1_CHMUX_SEL10	548h

Function

DMAC1 Channel mux select 10

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMUX_SEL	DMAC1 Channel mux select 0b - I3C1_RX_DMA 1b - Reserved

13.5.1.36 DMAC1 Channel mux select 11 (DMAC1_CHMUX_SEL11)

Offset

Register	Offset
DMAC1_CHMUX_SEL11	54Ch

Function

DMAC1 Channel mux select 11

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - I3C1_TX_DMA 1b - Reserved

13.5.1.37 DMAC1 Channel mux select 12 (DMAC1_CHMUX_SEL12)**Offset**

Register	Offset
DMAC1_CHMUX_SEL12	550h

Function

DMAC1 Channel mux select 12

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - Reserved 1b - FLEXCOM11_RX_DMA

13.5.1.38 DMAC1 Channel mux select 13 (DMAC1_CHMUX_SEL13)**Offset**

Register	Offset
DMAC1_CHMUX_SEL13	554h

Function

DMAC1 Channel mux select 13

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC
W																1_C...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - Reserved 1b - FLEXCOM11_TX_DMA

13.5.1.39 DMAC1 Channel mux select 14 (DMAC1_CHMUX_SEL14)

Offset

Register	Offset
DMAC1_CHMUX_SEL14	558h

Function

DMAC1 Channel mux select 14

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DMAC 1_C...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DMAC1_CHMUX_SEL	DMA channel mux select 0b - Reserved 1b - FLEXCOM12_RX_DMA

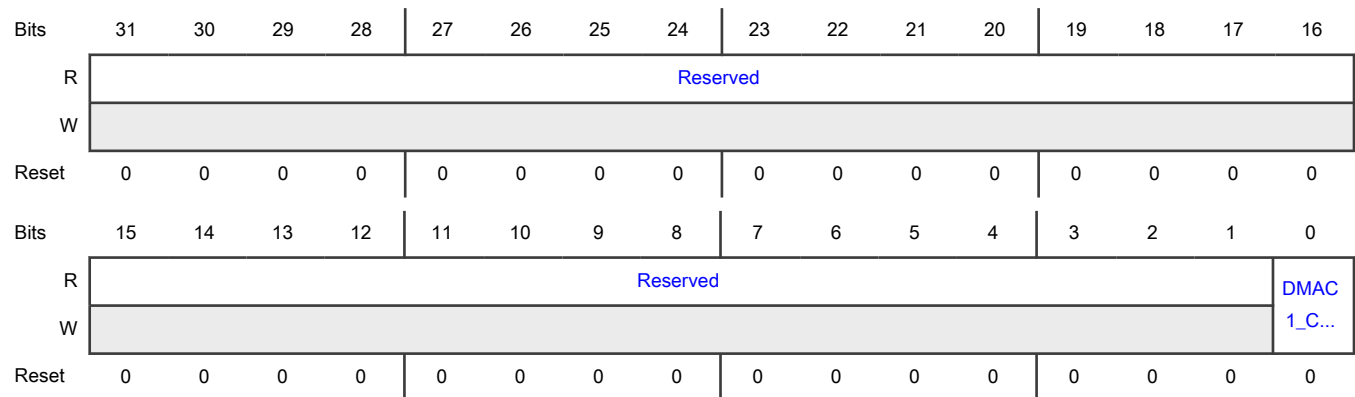
13.5.1.40 DMAC1 Channel mux select 15 (DMAC1_CHMUX_SEL15)

Offset

Register	Offset
DMAC1_CHMUX_SEL15	55Ch

Function

DMAC1 Channel mux select 15

Diagram**Fields**

Field	Function
31-1 —	Reserved
0 DMAC1_CHMU X_SEL	DMAC1 Channel mux select 0b - Reserved 1b - FLEXCOM12_TX_DMA

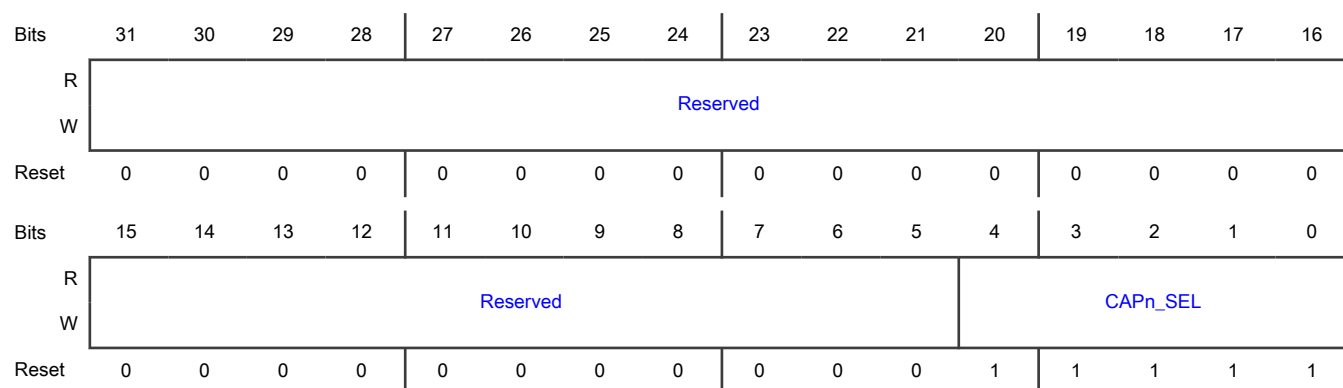
13.5.1.41 CT32BIT Timer Capture Multiplexers (CT32BIT0_CAP0_SEL - CT32BIT4_CAP3_SEL)**Offset**

For a = 0 to 4; b = 0 to 3:

Register	Offset
CT32BITa_CAPb_SEL	600h + (a × 10h) + (b × 4h)

Function

The 32-Bit Counter Timers have the provision each for 4 capture inputs. The source of each capture input is selected as defined below.

Diagram**Fields**

Field	Function
31-5 —	Reserved
4-0 CAPn_SEL	Counter Timer n, Capture Input m NOTE All other values are Reserved. 0_0000b - CT_INP0 (function must be selected in IOPCTL) 0_0001b - CT_INP1 (function must be selected in IOPCTL) 0_0010b - CT_INP2 (function must be selected in IOPCTL) 0_0011b - CT_INP3 (function must be selected in IOPCTL) 0_0100b - CT_INP4 (function must be selected in IOPCTL) 0_0101b - CT_INP5 (function must be selected in IOPCTL) 0_0110b - CT_INP6 (function must be selected in IOPCTL) 0_0111b - CT_INP7 (function must be selected in IOPCTL) 0_1000b - CT_INP8 (function must be selected in IOPCTL) 0_1001b - CT_INP9 (function must be selected in IOPCTL) 0_1010b - CT_INP10 (function must be selected in IOPCTL) 0_1011b - CT_INP11 (function must be selected in IOPCTL) 0_1100b - CT_INP12 (function must be selected in IOPCTL) 0_1101b - CT_INP13 (function must be selected in IOPCTL) 0_1110b - CT_INP14 (function must be selected in IOPCTL) 0_1111b - CT_INP15 (function must be selected in IOPCTL) 1_0000b - SHARED I2S0_WS 1_0001b - SHARED I2S1_WS 1_0010b - USB1_FRAME_TOGGLE (see USB Controller Chapter)

13.5.1.42 Frequency Measurement Input Channel Multiplexers (FMEASURE_CH0_SEL - FMEASURE_CH1_SEL)

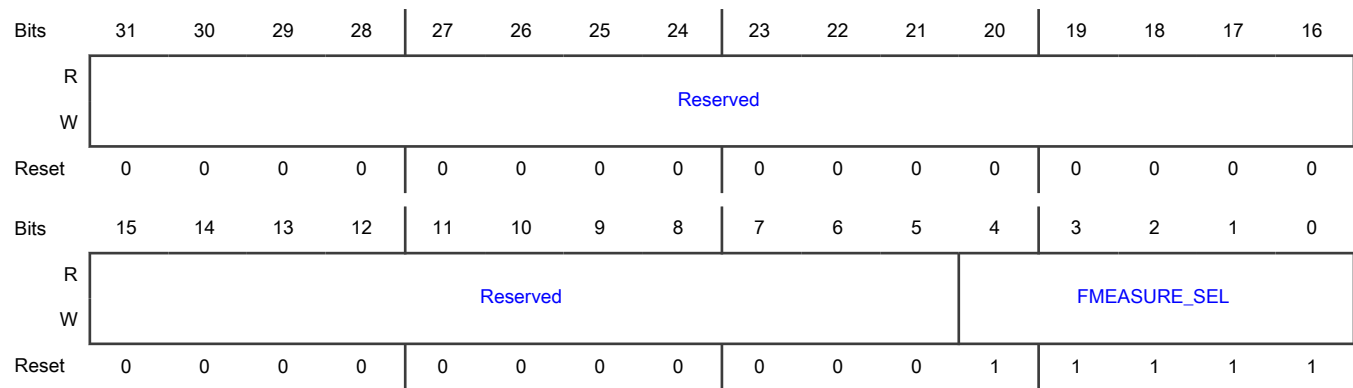
Offset

Register	Offset
FMEASURE_CH0_SEL	700h
FMEASURE_CH1_SEL	704h

Function

The frequency measurement block can source two clock inputs. The input channels to the frequency measurement block are selected as defined below. This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 FMEASURE_SEL	Frequency Measure Channel Selection 0_0000b - OSC_CLK 0_0001b - FRO_DIV16 0_0010b - FRO_DIV1 0_0011b - Low Power Oscillator Clock (LPOSC) 0_0100b - RTC 32 kHz OSC 0_0101b - Main SYCLK 0_0110b - FREQME_GPIO_CLK 0_0111b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_1000b - Reserved
	0_1001b - Reserved
	0_1010b - Reserved
	0_1011b - Clock Out
	0_1100b - Reserved
	0_1101b - Reserved
	0_1110b - Reserved
	0_1111b - Reserved
	1_0000b - Reserved
	1_0001b - Reserved

13.5.1.43 SMART_DMA trigger channel select (SMART_DMA_TRIG_CH0_SEL - SMART_DMA_TRIG_CH7_SEL)

Offset

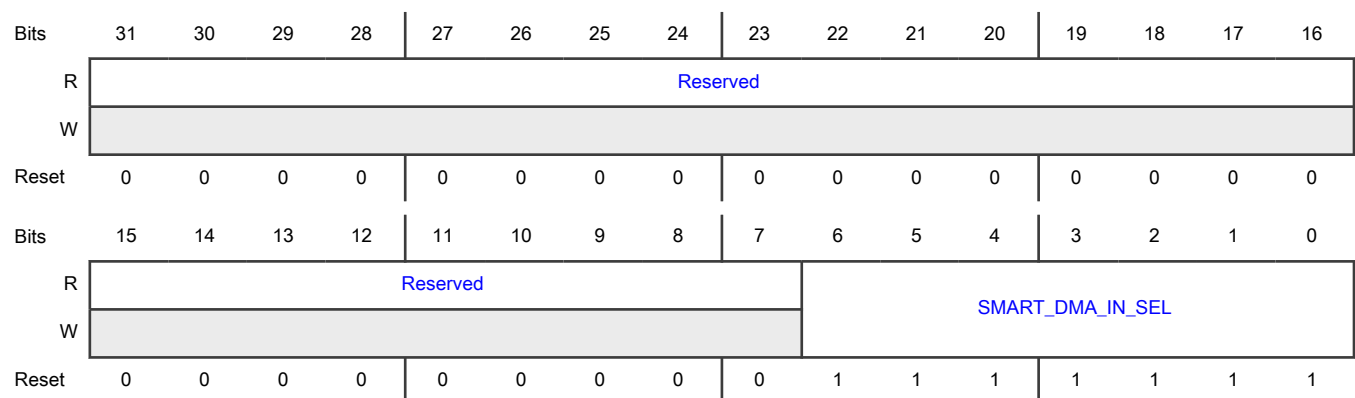
For a = 0 to 7:

Register	Offset
SMART_DMA_TRIG_CHa_SEL	720h + (a × 4h)

Function

The SMART_DMA has a provision for 8 trigger channels. The source of each input can be either 1 of 76 sources indicated and selected as defined below.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 SMART_DMA_I N_SEL	SMART_DMA Input(n) Selection 00 GPIO_P0_0 01 GPIO_P0_1 02 GPIO_P0_2 03 GPIO_P0_3 04 GPIO_P0_4 05 GPIO_P0_5 06 GPIO_P0_6 07 GPIO_P0_7 08 GPIO_P1_0 09 GPIO_P1_1 10 GPIO_P1_2 11 GPIO_P1_3 12 GPIO_P1_4 13 GPIO_P1_5 14 GPIO_P1_6 15 GPIO_P1_7 16 FLEXCOMM0_IRQ 17 FLEXCOMM1_IRQ 18 FLEXCOMM2_IRQ 19 FLEXCOMM3_IRQ 20 FLEXCOMM4_IRQ 21 FLEXCOMM5_IRQ 22 FLEXCOMM6_IRQ 23 FLEXCOMM7_IRQ 24 FLEXCOMM14_IRQ 25 FLEXCOMM16_IRQ 26 I3C0_IRQ 27 I3C1_IRQ 28 FLEXIO_IRQ 29 GPIO_INT0_IRQ0

Table continues on the next page...

Table continued from the previous page...

Field	Function
	30 GPIO_INT0_IRQ1
	31 GPIO_INT0_IRQ2
	32 GPIO_INT0_IRQ3
	33 GPIO_INT0_IRQ4
	34 GPIO_INT0_IRQ5
	35 GPIO_INT0_IRQ6
	36 GPIO_INT0_IRQ7
	37 NS_GPIO_HS_IRQ0
	38 NS_GPIO_HS_IRQ1
	39 SCT0_IRQ
	40 CTIMER0_IRQ
	41 CTIMER1_IRQ
	42 CTIMER2_IRQ
	43 CTIMER3_IRQ
	44 CTIMER4_IRQ
	45 UTICK0_IRQ
	46 MRT0_IRQ
	47 RTC_LITE0_IRQ
	48 OS_EVENT_IRQ
	49 WDT0_IRQ
	50 WDT1_IRQ
	51 ADC0_IRQ
	52 ACMP_IRQ
	53 D_MIC0
	54 HWVAD
	55 SDIO0_IRQ
	56 SDIO1_IRQ
	57 USB0_IRQ
	58 USB0_NEEDCLK
	59 LCDIF_IRQ
	60 GPU_IRQ
	61 SDMA0_IRQ
	62 SDMA1_IRQ
	63 POWERQUAD_IRQ

Table continues on the next page...

Table continued from the previous page...

Field	Function
	64 FLEXSPI_IRQ
	65 DSP_TIE_EXPSTATE1
	66 SCT_OUT8
	67 SCT_OUT9
	68 T4_MAT3
	69 T4_MAT2
	70 T3_MAT3
	71 T3_MAT2
	72 ARM_TXEV
	73 GPIOINT_BMATCH
	74 MIPI_IRQ
	75 USB_FS_TOGGLE

13.5.1.44 DMAC0 request enable 0 (DMAC0_REQ_ENA0)

Offset

Register	Offset
DMAC0_REQ_ENA0	740h

Function

DMAC0 request enable 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_	I3C1_	FLEX	FLEX	FLEX	FLEX	I3C0_	I3C0_	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0
W	TX	RX	COM...	COM...	COM...	COM...	TX	RX	_C...	_C...	_C...	_C...	_C...	_C...	_C...	_C...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX
W	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - Disable 1b - Enable
30 I3C1_RX	I3C1_RX enable 0b - Disable 1b - Enable
29 FLEXCOMM16_TX	FLEXCOMM16 TX enable 0b - Disable 1b - Enable
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - Disable 1b - Enable
27 FLEXCOMM14_TX	FLEXCOMM14 TX enable 0b - Disable 1b - Enable
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable 0b - Disable 1b - Enable
25 I3C0_TX	I3C TX enable 0b - Disable 1b - Enable
24 I3C0_RX	I3C RX enable 0b - Disable 1b - Enable
23 DMIC0_CH7_FLEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - Disable 1b - Enable
22 DMIC0_CH6_FLEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM13 RX DMA enable 0b - Disable 1b - Enable
21	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMIC0_CH5_F LEXCOMM10_ TX_DMA	0b - Disable 1b - Enable
20 DMIC0_CH4_F LEXCOMM10_ RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - Disable 1b - Enable
19 DMIC0_CH3_F LEXCOMM9_T X_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - Disable 1b - Enable
18 DMIC0_CH2_F LEXCOMM9_R X_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - Disable 1b - Enable
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - Disable 1b - Enable
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8 RX DMA enable 0b - Disable 1b - Enable
15 FLEXCOMM7_ TX	FLEXCOMM7 TX enable 0b - Disable 1b - Enable
14 FLEXCOMM7_ RX	FLEXCOMM7 RX enable 0b - Disable 1b - Enable
13 FLEXCOMM6_ TX	FLEXCOMM6 TX enable 0b - Disable 1b - Enable
12 FLEXCOMM6_ RX	FLEXCOMM6 RX enable 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 FLEXCOMM5_ TX	FLEXCOMM5 TX enable 0b - Disable 1b - Enable
10 FLEXCOMM5_ RX	FLEXCOMM5 RX enable 0b - Disable 1b - Enable
9 FLEXCOMM4_ TX	FLEXCOMM4 TX enable 0b - Disable 1b - Enable
8 FLEXCOMM4_ RX	FLEXCOMM4 RX enable 0b - Disable 1b - Enable
7 FLEXCOMM3_ TX	FLEXCOMM3 TX enable 0b - Disable 1b - Enable
6 FLEXCOMM3_ RX	FLEXCOMM3 RX enable 0b - Disable 1b - Enable
5 FLEXCOMM2_ TX	FLEXCOMM2 TX enable 0b - Disable 1b - Enable
4 FLEXCOMM2_ RX	FLEXCOMM2 RX enable 0b - Disable 1b - Enable
3 FLEXCOMM1_ TX	FLEXCOMM1 TX enable 0b - Disable 1b - Enable
2 FLEXCOMM1_ RX	FLEXCOMM1 RX enable 0b - Disable 1b - Enable
1	FLEXCOMM0 TX enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM0_TX	0b - Disable 1b - Enable
0 FLEXCOMM0_RX	FLEXCOMM0 RX enable 0b - Disable 1b - Enable

13.5.1.45 DMAC0 request enable 1 (DMAC0_REQ_ENA1)

Offset

Register	Offset
DMAC0_REQ_ENA1	744h

Function

DMAC0 request enable 1

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												HASH CRYPT...	FLEXC OM...	FLEXC OM...	FLEXC OM...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-5 —	Reserved
4 HASHCRYPT_I N	HASHCRYPT_IN 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 FLEXCOMM12_TX	FLEXCOMM12_TX 0b - Disable 1b - Enable
2 FLEXCOMM12_RX	FLEXCOMM12_RX 0b - Disable 1b - Enable
1 FLEXCOMM11_TX	FLEXCOMM11_TX 0b - Disable 1b - Enable
0 FLEXCOMM11_RX	FLEXCOMM11_RX 0b - Disable 1b - Enable

13.5.1.46 DMAC0 request enable set 0 (DMAC0_REQ_ENA0_SET)

Offset

Register	Offset
DMAC0_REQ_ENA0_SET	748h

Function

DMAC0 request enable set 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_TX	I3C1_RX	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	I3C0_TX	I3C0_RX	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - No effect 1b - Sets the ENA0 bit
30 I3C1_RX	I3C1_RX enable 0b - No effect 1b - Sets the ENA0 bit
29 FLEXCOMM16_TX	FLEXCOMM16 RX enable 0b - No effect 1b - Sets the ENA0 bit
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - No effect 1b - Sets the ENA0 bit
27 FLEXCOMM14_TX	FLEXCOMM14 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
25 I3C0_TX	I3C TX enable set 0b - No effect 1b - Sets the ENA0 Bit
24 I3C0_RX	I3C RX enable set 0b - No effect 1b - Sets the ENA0 Bit
23 DMIC0_CH7_FLEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
22 DMIC0_CH6_FLEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM10 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
21	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMIC0_CH5_F LEXCOMM10_ TX_DMA	0b - No effect 1b - Sets the ENA0 Bit
20 DMIC0_CH4_F LEXCOMM10_ RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
19 DMIC0_CH3_F LEXCOMM9_T X_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
18 DMIC0_CH2_F LEXCOMM9_R X_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
15 FLEXCOMM7_ TX	FLEXCOMM7 TX enable 0b - No effect 1b - Sets the ENA0 Bit
14 FLEXCOMM7_ RX	FLEXCOMM7 RX enable 0b - No effect 1b - Sets the ENA0 Bit
13 FLEXCOMM6_ TX	FLEXCOMM6 TX enable 0b - No effect 1b - Sets the ENA0 Bit
12 FLEXCOMM6_ RX	FLEXCOMM6 RX enable 0b - No effect 1b - Sets the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 FLEXCOMM5_ TX	FLEXCOMM5 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
10 FLEXCOMM5_ RX	FLEXCOMM5 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
9 FLEXCOMM4_ TX	FLEXCOMM4 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
8 FLEXCOMM4_ RX	FLEXCOMM4 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
7 FLEXCOMM3_ TX	FLEXCOMM3 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
6 FLEXCOMM3_ RX	FLEXCOMM3 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
5 FLEXCOMM2_ TX	FLEXCOMM2 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
4 FLEXCOMM2_ RX	FLEXCOMM2 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
3 FLEXCOMM1_ TX	FLEXCOMM1 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
2 FLEXCOMM1_ RX	FLEXCOMM1 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
1	FLEXCOMM0 TX enable set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM0_TX	0b - No effect 1b - Sets the ENA0 Bit
0 FLEXCOMM0_RX	FLEXCOMM0 RX enable set 0b - No effect 1b - Sets the ENA0 Bit

13.5.1.47 DMAC0 request enable set 1 (DMAC0_REQ_ENA1_SET)

Offset

Register	Offset
DMAC0_REQ_ENA1_SET	74Ch

Function

DMAC0 request enable set 1

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												HASH CRYPT...	FLEXC OM...	FLEXC OM...	FLEXC OM...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-5 —	Reserved
4 HASHCRYPT_IN	HASHCRYPT_IN 0b - No Effect 1b - Sets the ENA1 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 FLEXCOMM12_TX	FLEXCOMM12_TX 0b - No Effect 1b - Sets the ENA1 Bit
2 FLEXCOMM12_RX	FLEXCOMM12_RX 0b - No Effect 1b - Sets the ENA1 Bit
1 FLEXCOMM11_TX	FLEXCOMM11_TX 0b - No Effect 1b - Sets the ENA1 Bit
0 FLEXCOMM11_RX	FLEXCOMM11_RX 0b - No Effect 1b - Sets the ENA1 Bit

13.5.1.48 DMAC0 request enable clear 0 (DMAC0_REQ_ENA0_CLR)

Offset

Register	Offset
DMAC0_REQ_ENA0_CLR	750h

Function

DMAC0 request enable clear 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_TX	I3C1_RX	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	I3C0_TX	I3C0_RX	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...	FLEX COM...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - No effect 1b - Clears the ENA0 bit
30 I3C1_RX	I3C1_RX enable 0b - No effect 1b - Clears the ENA0 bit
29 FLEXCOMM16_TX	FLEXCOMM16 TX enable 0b - No effect 1b - Clears the ENA0 bit
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - No effect 1b - Clears the ENA0 bit
27 FLEXCOMM14_TX	FLEXCOMM14 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
25 I3C0_TX	I3C TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
24 I3C0_RX	I3C RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
23 DMIC0_CH7_FLEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
22 DMIC0_CH6_FLEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM13 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
21	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMIC0_CH5_F LEXCOMM10_ TX_DMA	0b - No effect 1b - Clears the ENA0 Bit
20 DMIC0_CH4_F LEXCOMM10_ RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
19 DMIC0_CH3_F LEXCOMM9_T X_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
18 DMIC0_CH2_F LEXCOMM9_R X_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
15 FLEXCOMM7_ TX	FLEXCOMM7 TX enable 0b - No effect 1b - Clears the ENA0 Bit
14 FLEXCOMM7_ RX	FLEXCOMM7 RX enable 0b - No effect 1b - Clears the ENA0 Bit
13 FLEXCOMM6_ TX	FLEXCOMM6 TX enable 0b - No effect 1b - Clears the ENA0 Bit
12 FLEXCOMM6_ RX	FLEXCOMM6 RX enable 0b - No effect 1b - Clears the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 FLEXCOMM5_ TX	FLEXCOMM5 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
10 FLEXCOMM5_ RX	FLEXCOMM5 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
9 FLEXCOMM4_ TX	FLEXCOMM4 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
8 FLEXCOMM4_ RX	FLEXCOMM4 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
7 FLEXCOMM3_ TX	FLEXCOMM3 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
6 FLEXCOMM3_ RX	FLEXCOMM3 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
5 FLEXCOMM2_ TX	FLEXCOMM2 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
4 FLEXCOMM2_ RX	FLEXCOMM2 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
3 FLEXCOMM1_ TX	FLEXCOMM1 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
2 FLEXCOMM1_ RX	FLEXCOMM1 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
1	FLEXCOMM0 TX enable clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM0_TX	0b - No effect 1b - Clears the ENA0 Bit
0 FLEXCOMM0_RX	FLEXCOMM0 RX enable clear FLEXCOMM0 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit

13.5.1.49 DMAC0 request enable 1 clear (DMAC0_REQ_ENA1_CLR)

Offset

Register	Offset
DMAC0_REQ_ENA1_CLR	754h

Function

DMAC0 request enable 1 clear

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												HASH CRYPT...	FLEXC OM...	FLEXC OM...	FLEXC OM...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-5 —	Reserved
4	HASHCRYPT_IN

Table continues on the next page...

Table continued from the previous page...

Field	Function
HASHCRYPT_I N	0b - No effect 1b - Clears the ENA1 Bit
3 FLEXCOMM12 _TX	FLEXCOMM12_TX 0b - No effect 1b - Clears the ENA1 Bit
2 FLEXCOMM12 _RX	FLEXCOMM12_RX 0b - No effect 1b - Clears the ENA1 Bit
1 FLEXCOMM11 _TX	FLEXCOMM11_TX 0b - No effect 1b - Clears the ENA1 Bit
0 FLEXCOMM11 _RX	FLEXCOMM11_RX 0b - No effect 1b - Clears the ENA1 Bit

13.5.1.50 DMAC1 request enable 0 (DMAC1_REQ_ENA0)

Offset

Register	Offset
DMAC1_REQ_ENA0	760h

Function

DMAC1 request enable 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_	I3C1_	FLEX	FLEX	FLEX	FLEX	I3C0_	I3C0_	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0
W	TX	RX	COM...	COM...	COM...	COM...	TX	RX	_C...	_C...	_C...	_C...	_C...	_C...	_C...	_C...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX
W	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - Disable 1b - Enable
30 I3C1_RX	I3C1_RX enable 0b - Disable 1b - Enable
29 FLEXCOMM16_TX	FLEXCOMM16 RX enable 0b - Disable 1b - Enable
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - Disable 1b - Enable
27 FLEXCOMM14_TX	FLEXCOMM14 TX enable 0b - Disable 1b - Enable
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable 0b - Disable 1b - Enable
25 I3C0_TX	I3C TX enable 0b - Disable 1b - Enable
24 I3C0_RX	I3C RX enable 0b - Disable 1b - Enable
23 DMIC0_CH7_F LEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - Disable 1b - Enable
22 DMIC0_CH6_F LEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM13 RX DMA enable 0b - Disable 1b - Enable
21	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMIC0_CH5_F LEXCOMM10_ TX_DMA	0b - Disable 1b - Enable
20 DMIC0_CH4_F LEXCOMM10_ RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - Disable 1b - Enable
19 DMIC0_CH3_F LEXCOMM9_T X_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - Disable 1b - Enable
18 DMIC0_CH2_F LEXCOMM9_R X_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - Disable 1b - Enable
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - Disable 1b - Enable
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8 RX DMA enable 0b - Disable 1b - Enable
15 FLEXCOMM7_ TX	FLEXCOMM7_TX FLEXCOMM7 TX enable 0b - Disable 1b - Enable
14 FLEXCOMM7_ RX	FLEXCOMM7_RX FLEXCOMM7 RX enable 0b - Disable 1b - Enable
13 FLEXCOMM6_ TX	FLEXCOMM6_TX FLEXCOMM6 TX enable 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 FLEXCOMM6_ RX	FLEXCOMM6_RX FLEXCOMM6 RX enable 0b - Disable 1b - Enable
11 FLEXCOMM5_ TX	FLEXCOMM5_TX FLEXCOMM5 TX enable 0b - Disable 1b - Enable
10 FLEXCOMM5_ RX	FLEXCOMM5_RX FLEXCOMM5 RX enable 0b - Disable 1b - Enable
9 FLEXCOMM4_ TX	FLEXCOMM4_TX FLEXCOMM4 TX enable 0b - Disable 1b - Enable
8 FLEXCOMM4_ RX	FLEXCOMM4_RX FLEXCOMM4 RX enable 0b - Disable 1b - Enable
7 FLEXCOMM3_ TX	FLEXCOMM3_TX FLEXCOMM3 TX enable 0b - Disable 1b - Enable
6 FLEXCOMM3_ RX	FLEXCOMM3_RX FLEXCOMM3 RX enable 0b - Disable 1b - Enable
5 FLEXCOMM2_ TX	FLEXCOMM2_TX FLEXCOMM2 TX enable 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enable
4 FLEXCOMM2_ RX	FLEXCOMM2_RX FLEXCOMM2 RX enable 0b - Disable 1b - Enable
3 FLEXCOMM1_ TX	FLEXCOMM1_TX FLEXCOMM1 TX enable 0b - Disable 1b - Enable
2 FLEXCOMM1_ RX	FLEXCOMM1_RX FLEXCOMM1 RX enable 0b - Disable 1b - Enable
1 FLEXCOMM0_ TX	FLEXCOMM0_TX FLEXCOMM0 TX enable 0b - Disable 1b - Enable
0 FLEXCOMM0_ RX	FLEXCOMM0_RX FLEXCOMM0 RX enable 0b - Disable 1b - Enable

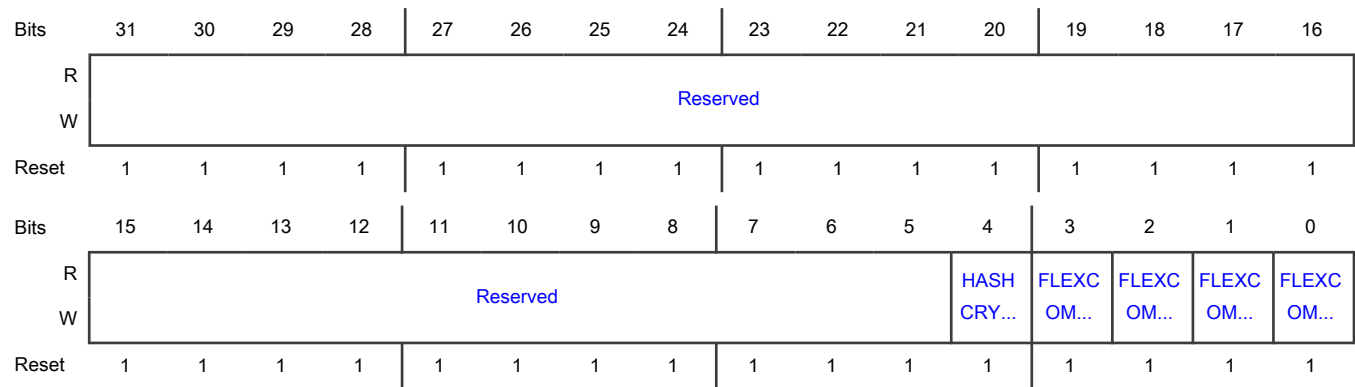
13.5.1.51 DMAC1 request enable 1 (DMAC1_REQ_ENA1)

Offset

Register	Offset
DMAC1_REQ_ENA1	764h

Function

DMAC1 request enable 1

Diagram**Fields**

Field	Function
31-5 —	Reserved
4 HASHCRYPT_I N	HASHCRYPT_IN 0b - Disable 1b - Enable
3 FLEXCOMM12 _TX	FLEXCOMM12_TX 0b - Disable 1b - Enable
2 FLEXCOMM12 _RX	FLEXCOMM12_RX 0b - Disable 1b - Enable
1 FLEXCOMM11 _TX	FLEXCOMM11_TX 0b - Disable 1b - Enable
0 FLEXCOMM11 _RX	FLEXCOMM11_RX 0b - Disable 1b - Enable

13.5.1.52 DMAC1 request enable set 0 (DMAC1_REQ_ENA0_SET)

Offset

Register	Offset
DMAC1_REQ_ENA0_SET	768h

Function

DMAC1 request enable set 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_TX	I3C1_RX	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	I3C0_TX	I3C0_RX	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...	DMIC0_C...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...	FLEX_COM...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - No effect 1b - Sets the ENA0 bit
30 I3C1_RX	I3C1_RX enable 0b - No effect 1b - Sets the ENA0 bit
29 FLEXCOMM16_TX	FLEXCOMM16 TX enable 0b - No effect 1b - Sets the ENA0 bit
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - No effect 1b - Sets the ENA0 bit
27	FLEXCOMM14 TX enable set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM14_TX	0b - No effect 1b - Sets the ENA0 Bit
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
25 I3C0_TX	I3C TX enable set 0b - No effect 1b - Sets the ENA0 Bit
24 I3C0_RX	I3C RX enable set 0b - No effect 1b - Sets the ENA0 Bit
23 DMIC0_CH7_F FLEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
22 DMIC0_CH6_F FLEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM13 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
21 DMIC0_CH5_F FLEXCOMM10_TX_DMA	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
20 DMIC0_CH4_F FLEXCOMM10_RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
19 DMIC0_CH3_F FLEXCOMM9_TX_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
18 DMIC0_CH2_F FLEXCOMM9_RX_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8 RX DMA enable 0b - No effect 1b - Sets the ENA0 Bit
15 FLEXCOMM7_ TX	FLEXCOMM7 TX enable 0b - No effect 1b - Sets the ENA0 Bit
14 FLEXCOMM7_ RX	FLEXCOMM7 RX enable 0b - No effect 1b - Sets the ENA0 Bit
13 FLEXCOMM6_ TX	FLEXCOMM6 TX enable 0b - No effect 1b - Sets the ENA0 Bit
12 FLEXCOMM6_ RX	FLEXCOMM6 RX enable 0b - No effect 1b - Sets the ENA0 Bit
11 FLEXCOMM5_ TX	FLEXCOMM5 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
10 FLEXCOMM5_ RX	FLEXCOMM5 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
9 FLEXCOMM4_ TX	FLEXCOMM4 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
8 FLEXCOMM4_ RX	FLEXCOMM4 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
7	FLEXCOMM3 TX enable set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM3_TX	0b - No effect 1b - Sets the ENA0 Bit
6 FLEXCOMM3_RX	FLEXCOMM3 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
5 FLEXCOMM2_TX	FLEXCOMM2 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
4 FLEXCOMM2_RX	FLEXCOMM2 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
3 FLEXCOMM1_TX	FLEXCOMM1 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
2 FLEXCOMM1_RX	FLEXCOMM1 RX enable set 0b - No effect 1b - Sets the ENA0 Bit
1 FLEXCOMM0_TX	FLEXCOMM0 TX enable set 0b - No effect 1b - Sets the ENA0 Bit
0 FLEXCOMM0_RX	FLEXCOMM0 RX enable set 0b - No effect 1b - Sets the ENA0 Bit

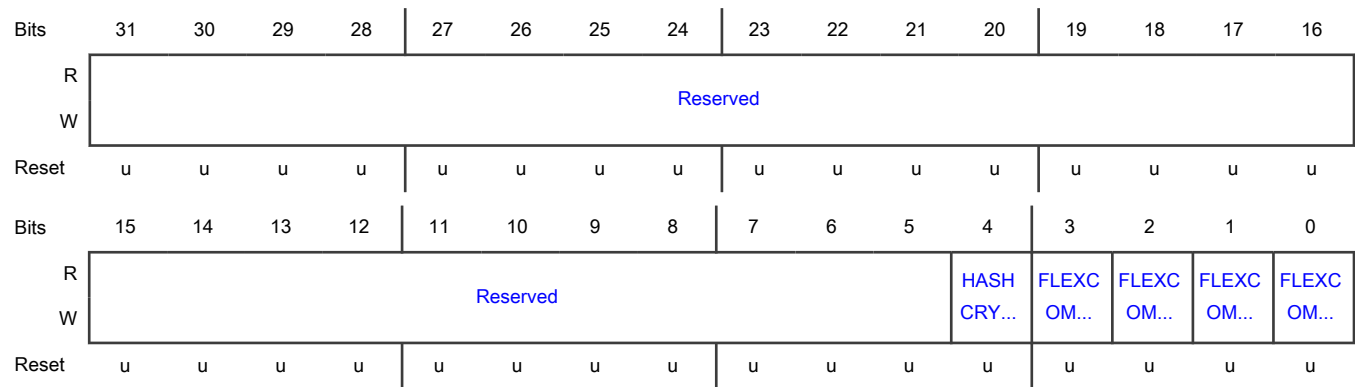
13.5.1.53 DMAC1 request enable set 1 (DMAC1_REQ_ENA1_SET)

Offset

Register	Offset
DMAC1_REQ_ENA1_SET	76Ch

Function

DMAC1 request enable set 1

Diagram**Fields**

Field	Function
31-5 —	Reserved
4 HASHCRYPT_I N	HASHCRYPT_IN 0b - No Effect 1b - Sets the ENA1 Bit
3 FLEXCOMM12 _TX	FLEXCOMM12_TX 0b - No Effect 1b - Sets the ENA1 Bit
2 FLEXCOMM12 _RX	FLEXCOMM12_RX 0b - No Effect 1b - Sets the ENA1 Bit
1 FLEXCOMM11 _TX	FLEXIO_SHFT_FLEXCOMM_TX 0b - No Effect 1b - Sets the ENA1 Bit
0 FLEXCOMM11 _RX	FLEXIO_SHFT_FLEXCOMM_RX 0b - No Effect 1b - Sets the ENA1 Bit

13.5.1.54 DMAC1 request enable clear 0 (DMAC1_REQ_ENA0_CLR)

Offset

Register	Offset
DMAC1_REQ_ENA0_CLR	770h

Function

DMAC1 request enable clear 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I3C1_	I3C1_	FLEX	FLEX	FLEX	FLEX	I3C0_	I3C0_	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0	DMIC0
W	TX	RX	COM...	COM...	COM...	COM...	TX	RX	_C...	_C...	_C...	_C...	_C...	_C...	_C...	_C...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX	FLEX
W	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...	COM...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 I3C1_TX	I3C1_TX enable 0b - No effect 1b - Clears the ENA0 bit
30 I3C1_RX	I3C1_RX enable 0b - No effect 1b - Clears the ENA0 bit
29 FLEXCOMM16_TX	FLEXCOMM16 TX enable 0b - No effect 1b - Clears the ENA0 bit
28 FLEXCOMM16_RX	FLEXCOMM16 RX enable 0b - No effect 1b - Clears the ENA0 bit
27	FLEXCOMM14 TX enable clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM14_TX	0b - No effect 1b - Clears the ENA0 Bit
26 FLEXCOMM14_RX	FLEXCOMM14 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
25 I3C0_TX	I3C TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
24 I3C0_RX	I3C RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
23 DMIC0_CH7_F FLEXCOMM13_TX_DMA	DMIC0 channel 7 / FLEXCOMM13 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
22 DMIC0_CH6_F FLEXCOMM13_RX_DMA	DMIC0 channel 6 / FLEXCOMM13 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
21 DMIC0_CH5_F FLEXCOMM10_TX_DMA	DMIC0 channel 5 / FLEXCOMM10 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
20 DMIC0_CH4_F FLEXCOMM10_RX_DMA	DMIC0 channel 4 / FLEXCOMM10 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
19 DMIC0_CH3_F FLEXCOMM9_TX_DMA	DMIC0 channel 3 / FLEXCOMM9 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
18 DMIC0_CH2_F FLEXCOMM9_RX_DMA	DMIC0 channel 2 / FLEXCOMM9 RX DMA enable 0b - No effect 1b - Clears the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 DMIC0_CH1_F LEXCOMM8_T X_DMA	DMIC0 channel 1 / FLEXCOMM8 TX DMA enable 0b - No effect 1b - Clears the ENA0 Bit
16 DMIC0_CH0_F LEXCOMM8_R X_DMA	DMIC0 channel 0 / FLEXCOMM8_RX_DMA enable 0b - No effect 1b - Clears the ENA0 Bit
15 FLEXCOMM7_ TX	FLEXCOMM7 TX enable 0b - No effect 1b - Clears the ENA0 Bit
14 FLEXCOMM7_ RX	FLEXCOMM7 RX enable 0b - No effect 1b - Clears the ENA0 Bit
13 FLEXCOMM6_ TX	FLEXCOMM6 TX enable 0b - No effect 1b - Clears the ENA0 Bit
12 FLEXCOMM6_ RX	FLEXCOMM6 RX enable 0b - No effect 1b - Clears the ENA0 Bit
11 FLEXCOMM5_ TX	FLEXCOMM5 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
10 FLEXCOMM5_ RX	FLEXCOMM5 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
9 FLEXCOMM4_ TX	FLEXCOMM4 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
8 FLEXCOMM4_ RX	FLEXCOMM4 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
7	FLEXCOMM3 TX enable clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM3_TX	0b - No effect 1b - Clears the ENA0 Bit
6 FLEXCOMM3_RX	FLEXCOMM3 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
5 FLEXCOMM2_TX	FLEXCOMM2 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
4 FLEXCOMM2_RX	FLEXCOMM2 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
3 FLEXCOMM1_TX	FLEXCOMM1 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
2 FLEXCOMM1_RX	FLEXCOMM1 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit
1 FLEXCOMM0_TX	FLEXCOMM0 TX enable clear 0b - No effect 1b - Clears the ENA0 Bit
0 FLEXCOMM0_RX	FLEXCOMM0 RX enable clear 0b - No effect 1b - Clears the ENA0 Bit

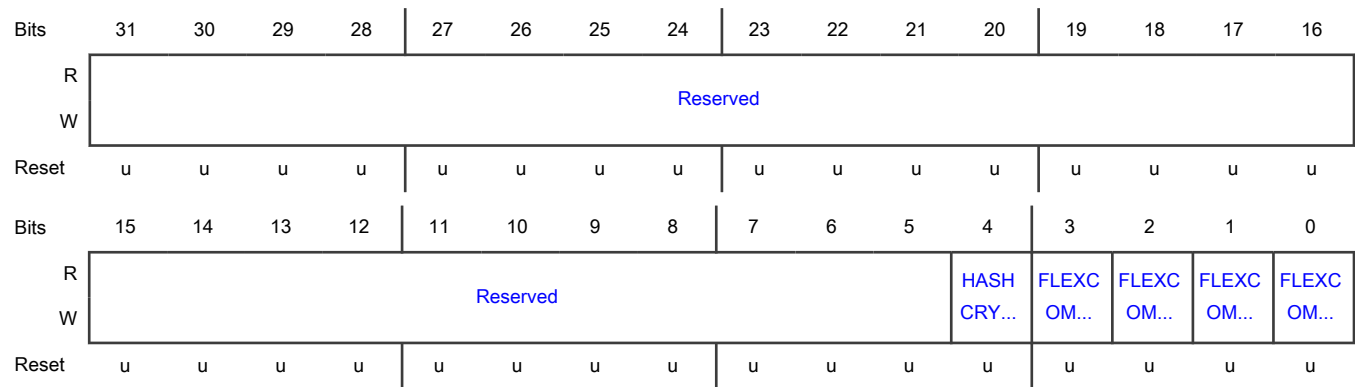
13.5.1.55 DMAC1 request enable 1 clear (DMAC1_REQ_ENA1_CLR)

Offset

Register	Offset
DMAC1_REQ_ENA1_CLR	774h

Function

DMAC1 request enable 1 clear

Diagram**Fields**

Field	Function
31-5 —	Reserved
4 HASHCRYPT_I N	HASHCRYPT_IN 0b - No effect 1b - Clears the ENA1 Bit
3 FLEXCOMM12 _TX	FLEXCOMM12_TX 0b - No effect 1b - Clears the ENA1 Bit
2 FLEXCOMM12 _RX	FLEXCOMM12_RX 0b - No effect 1b - Clears the ENA1 Bit
1 FLEXCOMM11 _TX	FLEXCOMM11_TX 0b - No effect 1b - Clears the ENA1 Bit
0 FLEXCOMM11 _RX	FLEXCOMM11_RX 0b - No effect 1b - Clears the ENA1 Bit

13.5.1.56 DMAC0 Input Trigger Enable 0 (DMAC0_ITRIG_ENA0)**Offset**

Register	Offset
DMAC0_ITRIG_ENA0	780h

Function

DMAC0 Input Trigger Enable 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	DMA_ TRI...	DMA_ TRI...	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMA_ TRI...	DMA_ TRI...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-27 —	Reserved
26 FLEXSPI1_TX	FlexSPI1_TX 0b - Disable 1b - Enable
25 FLEXSPI1_RX	FlexSPI1_RX 0b - Disable 1b - Enable
24 ADC	ADC 0b - Disable 1b - Enable
23 FLEXSPI0_TX	FlexSPI0_TX 0b - Disable 1b - Enable
22 FLEXSPI0_RX	FlexSPI0_RX 0b - Disable 1b - Enable
21 ACMP	ACMP 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
20 HASHCRYPT_ OUT	HASHCRYPT_OUT 0b - Disable 1b - Enable
19 SCT_DMA_RE Q1	SCT_DMA_REQ1 0b - Disable 1b - Enable
18 SCT_DMA_RE Q0	SCT_DMA_REQ0 0b - Disable 1b - Enable
17 DMA_TRIGOUT _D	DMA_TRIGOUT_D 0b - Disable 1b - Enable
16 DMA_TRIGOUT _C	DMA_TRIGOUT_C 0b - Disable 1b - Enable
15 DMA_TRIGOUT _B	DMA_TRIGOUT_B 0b - Disable 1b - Enable
14 DMA_TRIGOUT _A	DMA_TRIGOUT_A 0b - Disable 1b - Enable
13 T4_DMAREQ_ M1	T4_DMAREQ_M1 0b - Disable 1b - Enable
12 T4_DMAREQ_ M0	T4_DMAREQ_M0 0b - Disable 1b - Enable
11 T3_DMAREQ_ M1	T3_DMAREQ_M1 0b - Disable 1b - Enable
10	T3_DMAREQ_M0

Table continues on the next page...

Table continued from the previous page...

Field	Function
T3_DMAREQ_M0	0b - Disable 1b - Enable
9 T2_DMAREQ_M1	T2_DMAREQ_M1 0b - Disable 1b - Enable
8 T2_DMAREQ_M0	T2_DMAREQ_M0 0b - Disable 1b - Enable
7 T1_DMAREQ_M1	T1_DMAREQ_M1 0b - Disable 1b - Enable
6 T1_DMAREQ_M0	T1_DMAREQ_M0 0b - Disable 1b - Enable
5 T0_DMAREQ_M1	T0_DMAREQ_M1 0b - Disable 1b - Enable
4 T0_DMAREQ_M0	T0_DMAREQ_M0 0b - Disable 1b - Enable
3 GPIO_INT3	GPIO_INT3 0b - Disable 1b - Enable
2 GPIO_INT2	GPIO_INT2 0b - Disable 1b - Enable
1 GPIO_INT1	GPIO_INT1 0b - Disable 1b - Enable
0 GPIO_INT0	GPIO_INT0 0b - Disable 1b - Enable

Table continues on the next page...

Field	Function

13.5.1.57 DMAC0 Input Trigger Enable 0 Set (DMAC0_ITRIG_ENA0_SET)

Offset

Register	Offset
DMAC0_ITRIG_ENA0_SET	788h

Function

DMAC0 Input Trigger Enable 0 Set

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved					FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	SDMA 0_T...	SDMA 0_T...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SDMA 0_T...	SDMA 0_T...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-27 —	Reserved
26 FLEXSPI1_TX	FlexSPI1_TX set 0b - No Effect 1b - Sets the ENA0 Bit
25 FLEXSPI1_RX	FlexSPI1_RX set 0b - No Effect 1b - Sets the ENA0 Bit
24 ADC	ADC set 0b - No Effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the ENA0 Bit
23 FLEXSPI0_TX	FlexSPI0_TX set 0b - No Effect 1b - Sets the ENA0 Bit
22 FLEXSPI0_RX	FlexSPI0_RX set 0b - No Effect 1b - Sets the ENA0 Bit
21 ACMP	ACMP set 0b - No Effect 1b - Sets the ENA0 Bit
20 HASHCRYPT_OUT	HASHCRYPT_OUT set 0b - No Effect 1b - Sets the ENA0 Bit
19 SCT_DMA_REQ1_Q1	SCT_DMA_REQ1 set 0b - No Effect 1b - Sets the ENA0 Bit
18 SCT_DMA_REQ0_Q0	SCT_DMA_REQ0 set 0b - No Effect 1b - Sets the ENA0 Bit
17 SDMA0_TRIGOUT_UT_D	SDMA0_TRIGOUT_D set 0b - No Effect 1b - Sets the ENA0 Bit
16 SDMA0_TRIGOUT_UT_C	SDMA0_TRIGOUT_C set 0b - No Effect 1b - Sets the ENA0 Bit
15 SDMA0_TRIGOUT_UT_B	SDMA0_TRIGOUT_B set 0b - No Effect 1b - Sets the ENA0 Bit
14 SDMA0_TRIGOUT_UT_A	SDMA0_TRIGOUT_A set 0b - No Effect 1b - Sets the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 T4_DMAREQ_M1	T4_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
12 T4_DMAREQ_M0	T4_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
11 T3_DMAREQ_M1	T3_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
10 T3_DMAREQ_M0	T3_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
9 T2_DMAREQ_M1	T2_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
8 T2_DMAREQ_M0	T2_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
7 T1_DMAREQ_M1	T1_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
6 T1_DMAREQ_M0	T1_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
5 T0_DMAREQ_M1	T0_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
4 T0_DMAREQ_M0	T0_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
3 GPIO_INT3	GPIO_INT3 set

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Effect 1b - Sets the ENA0 Bit
2 GPIO_INT2	GPIO_INT2 set 0b - No Effect 1b - Sets the ENA0 Bit
1 GPIO_INT1	GPIO_INT1 set 0b - No Effect 1b - Sets the ENA0 Bit
0 GPIO_INT0	GPIO_INT0 set 0b - No Effect 1b - Sets the ENA0 Bit

13.5.1.58 DMAC0 Input Trigger Enable 0 Clear (DMAC0_ITRIG_ENA0_CLR)

Offset

Register	Offset
DMAC0_ITRIG_ENA0_CLR	790h

Function

DMAC0 Input Trigger Enable 0 Clear

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved					FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	SDMA 0_T...	SDMA 0_T...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SDMA 0_T...	SDMA 0_T...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-27 —	Reserved
26 FLEXSPI1_TX	FlexSPI1_TX clear 0b - No Effect 1b - Clears the ENA0 Bit
25 FLEXSPI1_RX	FlexSPI1_RX clear 0b - No Effect 1b - Clears the ENA0 Bit
24 ADC	ADC clear 0b - No Effect 1b - Clears the ENA0 Bit
23 FLEXSPI0_TX	FlexSPI0_TX clear 0b - No Effect 1b - Clears the ENA0 Bit
22 FLEXSPI0_RX	FlexSPI0_RX clear 0b - No Effect 1b - Clears the ENA0 Bit
21 ACMP	ACMP clear 0b - No Effect 1b - Clears the ENA0 Bit
20 HASHCRYPT_ OUT	HASHCRYPT_OUT clear 0b - No Effect 1b - Clears the ENA0 Bit
19 SCT_DMA_RE Q1	SCT_DMA_REQ1 clear 0b - No Effect 1b - Clears the ENA0 Bit
18 SCT_DMA_RE Q0	SCT_DMA_REQ0 clear 0b - No Effect 1b - Clears the ENA0 Bit
17 SDMA0_TRIGO UT_D	SDMA0_TRIGOUT_D clear 0b - No Effect 1b - Clears the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 SDMA0_TRIGO UT_C	SDMA0_TRIGOUT_C clear 0b - No Effect 1b - Clears the ENA0 Bit
15 SDMA0_TRIGO UT_B	SDMA0_TRIGOUT_B clear 0b - No Effect 1b - Clears the ENA0 Bit
14 SDMA0_TRIGO UT_A	SDMA0_TRIGOUT_A clear 0b - No Effect 1b - Clears the ENA0 Bit
13 T4_DMAREQ_ M1	T4_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
12 T4_DMAREQ_ M0	T4_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
11 T3_DMAREQ_ M1	T3_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
10 T3_DMAREQ_ M0	T3_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
9 T2_DMAREQ_ M1	T2_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
8 T2_DMAREQ_ M0	T2_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
7 T1_DMAREQ_ M1	T1_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
6	T1_DMAREQ_M0 clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
T1_DMAREQ_M0	0b - No Effect 1b - Clears the ENA0 Bit
5 T0_DMAREQ_M1	T0_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
4 T0_DMAREQ_M0	T0_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
3 GPIO_INT3	GPIO_INT3 clear 0b - No Effect 1b - Clears the ENA0 Bit
2 GPIO_INT2	GPIO_INT2 clear 0b - No Effect 1b - Clears the ENA0 Bit
1 GPIO_INT1	GPIO_INT1 clear 0b - No Effect 1b - Clears the ENA0 Bit
0 GPIO_INT0	GPIO_INT0 clear 0b - No Effect 1b - Clears the ENA0 Bit

13.5.1.59 DMAC1 Input Trigger Enable 0 (DMAC1_ITRIG_ENA0)

Offset

Register	Offset
DMAC1_ITRIG_ENA0	7A0h

Function

DMAC1 Input Trigger Enable 0

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	DMA_ TRI...	DMA_ TRI...	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMA_ TRI...	DMA_ TRI...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-27 —	Reserved
26 FLEXSPI1_TX	FlexSPI1_TX 0b - Disable 1b - Enable
25 FLEXSPI1_RX	FlexSPI1_RX 0b - Disable 1b - Enable
24 ADC	ADC 0b - Disable 1b - Enable
23 FLEXSPI0_TX	FlexSPI0_TX 0b - Disable 1b - Enable
22 FLEXSPI0_RX	FlexSPI0_RX 0b - Disable 1b - Enable
21 ACMP	ACMP 0b - Disable 1b - Enable
20	HASHCRYPT_OUT 0b - Disable

Table continues on the next page...

Table continued from the previous page...

Field	Function
HASHCRYPT_OUT	1b - Enable
19 SCT_DMA_REQ1	SCT_DMA_REQ1 0b - Disable 1b - Enable
18 SCT_DMA_REQ0	SCT_DMA_REQ0 0b - Disable 1b - Enable
17 DMA_TRIGOUT_D	DMA_TRIGOUT_D 0b - Disable 1b - Enable
16 DMA_TRIGOUT_C	DMA_TRIGOUT_C 0b - Disable 1b - Enable
15 DMA_TRIGOUT_B	DMA_TRIGOUT_B 0b - Disable 1b - Enable
14 DMA_TRIGOUT_A	DMA_TRIGOUT_A 0b - Disable 1b - Enable
13 T4_DMAREQ_M1	T4_DMAREQ_M1 0b - Disable 1b - Enable
12 T4_DMAREQ_M0	T4_DMAREQ_M0 0b - Disable 1b - Enable
11 T3_DMAREQ_M1	T3_DMAREQ_M1 0b - Disable 1b - Enable
10 T3_DMAREQ_M0	T3_DMAREQ_M0 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 T2_DMAREQ_M1	T2_DMAREQ_M1 0b - Disable 1b - Enable
8 T2_DMAREQ_M0	T2_DMAREQ_M0 0b - Disable 1b - Enable
7 T1_DMAREQ_M1	T1_DMAREQ_M1 0b - Disable 1b - Enable
6 T1_DMAREQ_M0	T1_DMAREQ_M0 0b - Disable 1b - Enable
5 T0_DMAREQ_M1	T0_DMAREQ_M1 0b - Disable 1b - Enable
4 T0_DMAREQ_M0	T0_DMAREQ_M0 0b - Disable 1b - Enable
3 GPIO_INT3	GPIO_INT3 0b - Disable 1b - Enable
2 GPIO_INT2	GPIO_INT2 0b - Disable 1b - Enable
1 GPIO_INT1	GPIO_INT1 0b - Disable 1b - Enable
0 GPIO_INT0	GPIO_INT0 0b - Disable 1b - Enable

13.5.1.60 DMAC1 Input Trigger Enable 0 set (DMAC1_ITRIG_ENA0_SET)

Offset

Register	Offset
DMAC1_ITRIG_ENA0_SET	7A8h

Function

DMAC1 Input Trigger Enable 0 set

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved					FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	SDMA O_T...	SDMA O_T...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SDMA O_T...	SDMA O_T...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-27 —	Reserved
26 FLEXSPI1_TX	FlexSPI1_TX set 0b - No Effect 1b - Sets the ENA0 Bit
25 FLEXSPI1_RX	FlexSPI1_RX set 0b - No Effect 1b - Sets the ENA0 Bit
24 ADC	ADC set 0b - No Effect 1b - Sets the ENA0 Bit
23	FlexSPI0_TX set

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXSPI0_TX	0b - No Effect 1b - Sets the ENA0 Bit
22 FLEXSPI0_RX	FlexSPI0_RX set 0b - No Effect 1b - Sets the ENA0 Bit
21 ACMP	ACMP set 0b - No Effect 1b - Sets the ENA0 Bit
20 HASHCRYPT_ OUT	HASHCRYPT_OUT set 0b - No Effect 1b - Sets the ENA0 Bit
19 SCT_DMA_RE Q1	SCT_DMA_REQ1 set 0b - No Effect 1b - Sets the ENA0 Bit
18 SCT_DMA_RE Q0	SCT_DMA_REQ0 set 0b - No Effect 1b - Sets the ENA0 Bit
17 SDMA0_TRIGO UT_D	SDMA0_TRIGOUT_D set 0b - No Effect 1b - Sets the ENA0 Bit
16 SDMA0_TRIGO UT_C	SDMA0_TRIGOUT_C set 0b - No Effect 1b - Sets the ENA0 Bit
15 SDMA0_TRIGO UT_B	SDMA0_TRIGOUT_B set 0b - No Effect 1b - Sets the ENA0 Bit
14 SDMA0_TRIGO UT_A	SDMA0_TRIGOUT_A set 0b - No Effect 1b - Sets the ENA0 Bit
13 T4_DMAREQ_ M1	T4_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 T4_DMAREQ_M0	T4_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
11 T3_DMAREQ_M1	T3_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
10 T3_DMAREQ_M0	T3_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
9 T2_DMAREQ_M1	T2_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
8 T2_DMAREQ_M0	T2_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
7 T1_DMAREQ_M1	T1_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
6 T1_DMAREQ_M0	T1_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
5 T0_DMAREQ_M1	T0_DMAREQ_M1 set 0b - No Effect 1b - Sets the ENA0 Bit
4 T0_DMAREQ_M0	T0_DMAREQ_M0 set 0b - No Effect 1b - Sets the ENA0 Bit
3 GPIO_INT3	GPIO_INT3 set 0b - No Effect 1b - Sets the ENA0 Bit
2 GPIO_INT2	GPIO_INT2 set

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No Effect 1b - Sets the ENA0 Bit
1 GPIO_INT1	GPIO_INT1 set 0b - No Effect 1b - Sets the ENA0 Bit
0 GPIO_INT0	GPIO_INT0 set 0b - No Effect 1b - Sets the ENA0 Bit

13.5.1.61 DMAC1 Input Trigger Enable 0 clear (DMAC1_ITRIG_ENA0_CLR)

Offset

Register	Offset
DMAC1_ITRIG_ENA0_CLR	7B0h

Function

DMAC1 Input Trigger Enable 0 clear

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved					FLEXS PI...	FLEXS PI...	ADC	FLEXS PI...	FLEXS PI...	ACMP	HASH CRY...	SCT_D MA...	SCT_D MA...	SDMA 0_T...	SDMA 0_T...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SDMA 0_T...	SDMA 0_T...	T4_D MAR...	T4_D MAR...	T3_D MAR...	T3_D MAR...	T2_D MAR...	T2_D MAR...	T1_D MAR...	T1_D MAR...	T0_D MAR...	T0_D MAR...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...	GPIO_ IN...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26 FLEXSPI1_TX	FlexSPI1_TX clear 0b - No Effect 1b - Clears the ENA0 Bit
25 FLEXSPI1_RX	FlexSPI1_RX clear 0b - No Effect 1b - Clears the ENA0 Bit
24 ADC	ADC clear 0b - No Effect 1b - Clears the ENA0 Bit
23 FLEXSPI0_TX	FlexSPI0_TX clear 0b - No Effect 1b - Clears the ENA0 Bit
22 FLEXSPI0_RX	FlexSPI0_RXclear 0b - No Effect 1b - Clears the ENA0 Bit
21 ACMP	ACMP clear 0b - No Effect 1b - Clears the ENA0 Bit
20 HASHCRYPT_ OUT	HASHCRYPT_OUTclear 0b - No Effect 1b - Clears the ENA0 Bit
19 SCT_DMA_RE Q1	SCT_DMA_REQ1 clear 0b - No Effect 1b - Clears the ENA0 Bit
18 SCT_DMA_RE Q0	SCT_DMA_REQ0 clear 0b - No Effect 1b - Clears the ENA0 Bit
17 SDMA0_TRIGO UT_D	SDMA0_TRIGOUT_D clear 0b - No Effect 1b - Clears the ENA0 Bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 SDMA0_TRIGO UT_C	SDMA0_TRIGOUT_C clear 0b - No Effect 1b - Clears the ENA0 Bit
15 SDMA0_TRIGO UT_B	SDMA0_TRIGOUT_B clear 0b - No Effect 1b - Clears the ENA0 Bit
14 SDMA0_TRIGO UT_A	SDMA0_TRIGOUT_A clear 0b - No Effect 1b - Clears the ENA0 Bit
13 T4_DMAREQ_ M1	T4_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
12 T4_DMAREQ_ M0	T4_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
11 T3_DMAREQ_ M1	T3_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
10 T3_DMAREQ_ M0	T3_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
9 T2_DMAREQ_ M1	T2_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
8 T2_DMAREQ_ M0	T2_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
7 T1_DMAREQ_ M1	T1_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
6	T1_DMAREQ_M0 clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
T1_DMAREQ_M0	0b - No Effect 1b - Clears the ENA0 Bit
5 T0_DMAREQ_M1	T0_DMAREQ_M1 clear 0b - No Effect 1b - Clears the ENA0 Bit
4 T0_DMAREQ_M0	T0_DMAREQ_M0 clear 0b - No Effect 1b - Clears the ENA0 Bit
3 GPIO_INT3	GPIO_INT3 clear 0b - No Effect 1b - Clears the ENA0 Bit
2 GPIO_INT2	GPIO_INT2 clear 0b - No Effect 1b - Clears the ENA0 Bit
1 GPIO_INT1	GPIO_INT1 clear 0b - No Effect 1b - Clears the ENA0 Bit
0 GPIO_INT0	GPIO_INT0 clear 0b - No Effect 1b - Clears the ENA0 Bit

Chapter 14

General Purpose Input/Output (GPIO)

14.1 Chip-specific GPIO information

Table 55. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	GPIO
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

This table describes the GPIO signals implemented on the chip.

Signal	I/O	Description
SEC_PIO0_0 to SEC_PIO0_31	IO	Secure HS GPIO port 0 (overlays non-secure Port 0)
PIO0_0 through PIO0_31	IO	HS GPIO port 0
PIO1_0 through PIO1_31	IO	HS GPIO port 1
PIO2_0 through PIO2_31	IO	HS GPIO port 2
PIO3_0 through PIO3_31	IO	HS GPIO port 3
PIO4_0 through PIO4_31	IO	HS GPIO port 4
PIO5_0 through PIO5_31	IO	HS GPIO port 5
PIO6_0 through PIO6_27	IO	HS GPIO port 6

14.1.1 Module instances

The chip includes two instances of GPIO module- Secure HS GPIO (Port 0) and HS GPIO (Ports 0-7). Secure Port 0 overlays non-secure Port 0 pins.

14.1.2 Initialization

Initial configuration of the GPIO can be accomplished as follows:

- For the non-secure GPIO ports, enable clocks in the PSCCTL1 register. For the secure GPIO port, enable the clock in the PSCCTL1 register (See CLKCTL). The secure GPIO is not secured by default; it must be secured in the same manner as other peripherals if security is desired.
- Clear the non-secure GPIO peripheral resets in the PRSTCTL1 register by writing to the PRSTCTL1_CLR register. Clear the secure GPIO peripheral reset in the PRSTCTL1 register by writing to the PRSTCTL1_CLR register (See RSTCTL).
- The non-secure GPIO can provide interrupts to the NVIC. These interrupts can alternatively be connected to the DSP. To allow non-secure GPIO interrupts to wake-up the device from deep-sleep mode, enable this in the STARTEN1 register (See SYSCTL).

NOTE

GPIO interrupts require a clock for edge detection, so only level mode will work for wake-up from deep-sleep.

- The secure GPIO can also provide interrupts to the NVIC. To allow secure GPIO interrupts to wake-up the device from deep-sleep mode, enable this in the STARTEN0 register (See SYSCCTL).

NOTE

GPIO interrupts require a clock for edge detection, so only level mode will work for wake-up from deep-sleep.

- Use the IOPCTL registers to connect the non-secure and/or secure (inputs/outputs) to external pins.

14.2 Overview

This section provides an introduction to the GPIO module. Each GPIO port supports up to 32 pins. The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output.

A single additional GPIO port, called Secure GPIO, can be optionally secured by software separately from the non-secure GPIO ports. The Secure GPIO has all of the features of non-secure GPIO and has its own instance of the GPIO interrupts.

14.2.1 Features

Each GPIO pin includes the following features:

- It can be configured as input or output by software.
 - Direction (input/output) can be set and cleared individually.
- It can be sensed and set individually by GPIO registers.
- It can optionally contribute to one of two GPIO interrupts, with selection of polarity, level or edge detection.
 - Enable for contributing to interrupt A (contributing pin states ORed together)
 - Enable for contributing to interrupt B (contributing pin states ORed together)
 - Status flag for each pin interrupt A
 - Status flag for each pin interrupt B
 - Ability to clear status flag for each pin interrupt A
 - Ability to clear status flag for each pin interrupt B
- Secure GPIO pins overlay non-secure port 0 pins and are intended to be secured in the application if used.

14.3 Application information

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PIN registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PIN register.

14.4 Functional description

The following sections describe functional details of the GPIO module.

14.4.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the I/O Configuration logic. A pin does not have to be selected for GPIO in I/O Configuration in order to read its state. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PIN register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPIN) register. Pins having a 1 in the port's Mask register will read as 0 from its MPIN register.

14.4.2 GPIO output

Each GPIO pin has an output bit in the GPIO. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin's output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOPCTL (this is the default), and
2. the pin must be selected for output by a 1 in its port's DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port's PIN register loads the output bits of all the pins written to.
- Writing to a port's MPIN register loads the output bits of pins identified by zeros in corresponding positions of the port's MASK register.
- Writing ones to a port's SET register sets output bits.
- Writing ones to a port's CLR register clears output bits.
- Writing ones to a port's NOT register toggles/complements/inverts output bits.

The state of a port's output bits can be read from its SET register. Reading any of the registers returns the state of pins, regardless of their direction or alternate functions.

14.4.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPIN register. Zeros in MASK enable the corresponding pins to be read from and written to MPIN. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPIN. When a port's MASK register contains all zeros, its PIN and MPIN registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPIN or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPIN or MASK registers.

14.4.4 GPIO interrupts

The GPIO function includes two interrupts, A and B, that can be used with any GPIO pin. Each interrupt has enables for each pin (INTENAn and INTENBn registers, one for each port), and a status for each pin (INTSTATAn and INTSTATBn registers, one for each port). Polarity and edge versus level trigger can also be set for each pin (INTPOLn and INTEDGn registers, one for each port).

Any GPIO pin whose configured condition is satisfied, and is enabled for one of the two GPIO interrupts, will send an interrupt to the CM33 and the Fusion DSP interrupt controllers. All contributing conditions need to be cleared to cause the interrupt to be deasserted.

The single secure GPIO port also supports its own GPIO interrupts.

14.4.5 Limitations in deep sleep mode

The GPIO_INT_A/B interrupt structure does not properly handle edge-triggered wakeup when the clocks are off. The IRQ to the CPU can still be configured as edge-triggered but the async wakeup signal will always behave as level-sensitive when clocks are not present. As soon as the chip enters deep_sleep mode (with the pin low) the level-sensitive wakeup signal is triggered to wake the part up. Since the IRQ itself is edge-triggered, there is no pending interrupt to the CPU so the part re-enters deep_sleep mode and then immediately wakes up again. The chip continues to toggle between active and deep_sleep until the pin is raised high.

The recommended work-around is to configure the interrupt as HIGH/LEVEL-SENSITIVE to wake the chip up, and then wait until the pin returns low before re-entering deep_sleep mode (and, possibly before performing whatever CPU operations are required). The part will still be asleep for approximately 97% of the time (21.6 mS out of every 22.22 mS). One possible means of determining when the pin has returned low would be to reconfigure the interrupt to FALLING-EDGE when the part wakes up and then switch it back to HIGH/LEVEL before going back to sleep. Polling of the pin or use of a timer are other alternatives.

14.5 Memory map and register definition

This section includes the GPIO memory map and detailed descriptions of all registers.

14.5.1 SECGPIO General Purpose I/O (GPIO) register descriptions

GPIO port addresses can be read and written as bytes, halfwords, or words. Registers within each kind of GPIO are functionally the same.

NOTE

Supported pins depends on the specific device and package.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

14.5.1.1 GPIO memory map

SECGPIO base address: 4020_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 1Fh	Byte pin registers for all port GPIO pins (B0_0 - B0_31)	8	RW	See section
1000h - 107Ch	Word pin registers for all port GPIO pins (W0_0 - W0_31)	32	RW	0000_0000h
2000h	Port direction (DIR0)	32	W	0000_0000h
2080h	Port mask (MASK0)	32	RW	0000_0000h
2100h	Port pin (PIN0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
2180h	Masked Port Pin (MPIN0)	32	RW	0000_0000h
2200h	Port set (SET0)	32	RW	0000_0000h
2280h	Port clear (CLR0)	32	RW	See section
2300h	Port toggle (NOT0)	32	W	See section
2380h	Port direction set (DIRSET0)	32	W	See section
2400h	Port direction clear (DIRCLR0)	32	RW	See section
2480h	Port direction toggle (DIRNOT0)	32	W	See section
2500h	Interrupt A enable control (INTENA0)	32	RW	0000_0000h
2580h	Interrupt B enable control (INTENB0)	32	RW	0000_0000h
2600h	Interrupt polarity control (INTPOL0)	32	RW	0000_0000h
2680h	Interrupt edge select (INTEDG0)	32	RW	0000_0000h
2700h	Interrupt status for interrupt A (INTSTATA0)	32	RW	0000_0000h
2780h	Interrupt status for interrupt B (INTSTATB0)	32	RW	0000_0000h

14.5.1.2 Byte pin registers for all port GPIO pins (B0_0 - B0_31)

Offset

For b = 0 to 31:

Register	Offset
B0_b	0h + (b × 1h)

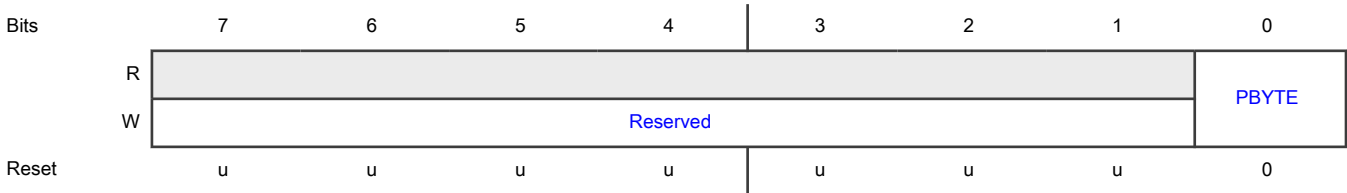
Function

Each GPIO pin has a byte register. Software reads and writes bytes to access individual pins. It can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram



Fields

Field	Function
7-1 —	Reserved (0 on read, ignored on write)
0 PBYTE	Port Byte Read- specifies the state of GPIO pins(PIOa_b), regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as 0. Write- loads the pin's output bit.

14.5.1.3 Word pin registers for all port GPIO pins (W0_0 - W0_31)

Offset

For b = 0 to 31:

Register	Offset
W0_b	1000h + (b × 4h)

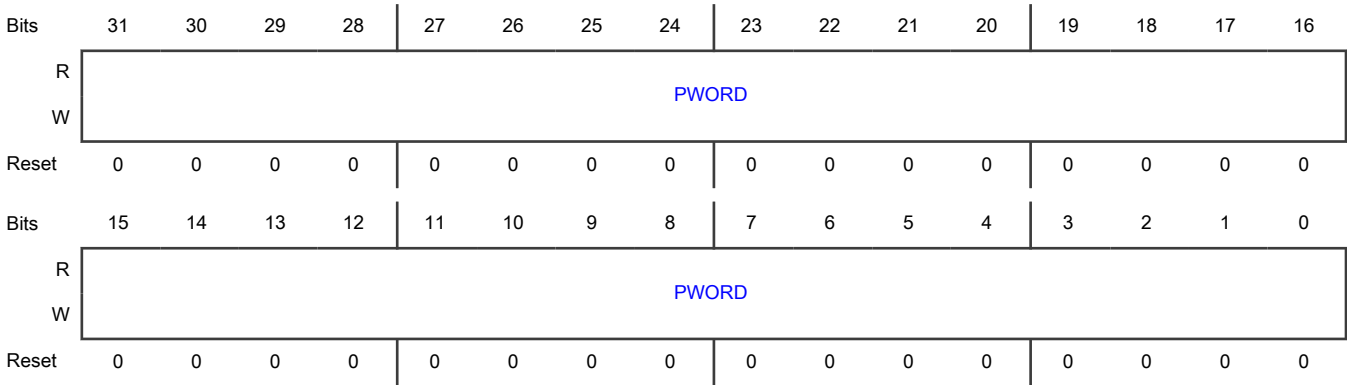
Function

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as zeros. Any write will clear the pin's output bit if the value written is all zeros, else it will set the pin's output bit.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram



Fields

Field	Function
31-0 PWORD	PWORD Read 0- pin PIOa_b is LOW. Write 0- clear output bit. Read 0xFFFF FFFF- pin PIOa_b is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF- sets output bit. Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit.

14.5.1.4 Port direction (DIR0)

Offset

Register	Offset
DIR0	2000h

Function

The DIR registers configure each pin in a GPIO port as input or output. Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRP3 1	DIRP3 0	DIRP2 9	DIRP2 8	DIRP2 7	DIRP2 6	DIRP2 5	DIRP2 4	DIRP2 3	DIRP2 2	DIRP2 1	DIRP2 0	DIRP1 9	DIRP1 8	DIRP1 7	DIRP1 6
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DIRP1 5	DIRP1 4	DIRP1 3	DIRP1 2	DIRP1 1	DIRP1 0	DIRP9	DIRP8	DIRP7	DIRP6	DIRP5	DIRP4	DIRP3	DIRP2	DIRP1	DIRP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 DIRPn	Selects pin direction for pin PIOa_b. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Input 1b - Output

14.5.1.5 Port mask (MASK0)**Offset**

Register	Offset
MASK0	2080h

Function

It defines which port pins are accessible in its MPIN register. Zeros in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	MASK P31	MASK P30	MASK P29	MASK P28	MASK P27	MASK P26	MASK P25	MASK P24	MASK P23	MASK P22	MASK P21	MASK P20	MASK P19	MASK P18	MASK P17	MASK P16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	MASK P15	MASK P14	MASK P13	MASK P12	MASK P11	MASK P10	MASK P9	MASK P8	MASK P7	MASK P6	MASK P5	MASK P4	MASK P3	MASK P2	MASK P1	MASK P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MASKPn	Port Mask Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Read MPIN: pin state; write MPIN: load output bit 1b - Read MPIN: 0; write MPIN: output bit not affected

14.5.1.6 Port pin (PIN0)

Offset

Register	Offset
PIN0	2100h

Function

Reading these registers returns the current state of the pins, regardless of direction, masking, or alternate digital functions. Pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram

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

Fields

Field	Function
31-0 PORTn	Port pins Reads pin states or loads output bits.

Table continues on the next page...

Field	Function
	Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Read- pin is low; Write- clear output bit 1b - Read- pin is high; Write- set output bit

14.5.1.7 Masked Port Pin (MPIN0)

Offset

Register	Offset
MPIN0	2180h

Function

These registers are similar to the PIN registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR
W	TP31	TP30	TP29	TP28	TP27	TP26	TP25	TP24	TP23	TP22	TP21	TP20	TP19	TP18	TP17	TP16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR
W	TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MPORTPn	Mask bits for port pins Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. NOTE The data read after reset depends on the state of the pin, which in turn may depend on an external source. 0b - Read- pin is LOW and/or the corresponding bit in the MASK register is 1; write- clear output bit if the corresponding bit in the MASK register is 0

Table continues on the next page...

Field	Function
	1b - Read- pin is HIGH and the corresponding bit in the MASK register is 0; write- set output bit if the corresponding bit in the MASK register is 0

14.5.1.8 Port set (SET0)

Offset

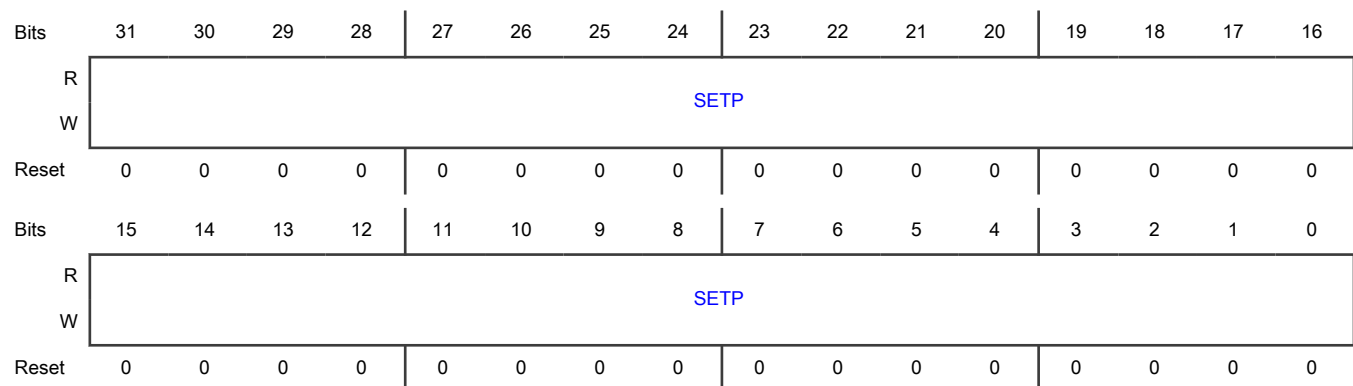
Register	Offset
SET0	2200h

Function

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram



Fields

Field	Function
31-0	Read or set output bits
SETP	Sets output bits for port pins 0000_0000_0000_0000_0000_0000_0000b - Read- output bit; write- no operation 0000_0000_0000_0000_0000_0000_0001b - Read- output bit; write- set output bit

14.5.1.9 Port clear (CLR0)

Offset

Register	Offset
CLR0	2280h

Function

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLRP 31	CLRP 30	CLRP 29	CLRP 28	CLRP 27	CLRP 26	CLRP 25	CLRP 24	CLRP 23	CLRP 22	CLRP 21	CLRP 20	CLRP 19	CLRP 18	CLRP 17	CLRP 16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLRP 15	CLRP 14	CLRP 13	CLRP 12	CLRP 11	CLRP 10	CLRP 9	CLRP 8	CLRP 7	CLRP 6	CLRP 5	CLRP 4	CLRP 3	CLRP 2	CLRP 1	CLRP 0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0	Clear output bits
CLRPN	Clear output bits for port pins 0b - No operation 1b - Clears output bit

14.5.1.10 Port toggle (NOT0)

Offset

Register	Offset
NOT0	2300h

Function

Output bits can be toggled by writing ones to these write-only registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

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

Fields

Field	Function
31-0 NOTPn	Toggle output bits Toggle direction bits for port pins. 0b - No operation 1b - Toggle output bit

14.5.1.11 Port direction set (DIRSET0)**Offset**

Register	Offset
DIRSET0	2380h

Function

Direction bits can be set by writing ones to these registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP...	DIRSE TP9	DIRSE TP8	DIRSE TP7	DIRSE TP6	DIRSE TP5	DIRSE TP4	DIRSE TP3	DIRSE TP2	DIRSE TP1	DIRSE TP0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 DIRSETPn	Direction set bits for Port pins Set pin direction for port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - No operation 1b - Sets direction bit

14.5.1.12 Port direction clear (DIRCLR0)

Offset

Register	Offset
DIRCLR0	2400h

Function

Direction bits can be cleared by writing ones to these write-only registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP...	DIRCL RP9	DIRCL RP8	DIRCL RP7	DIRCL RP6	DIRCL RP5	DIRCL RP4	DIRCL RP3	DIRCL RP2	DIRCL RP1	DIRCL RP0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 DIRCLRPn	Clear direction bits. Clear pin direction bits for Port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - No operation 1b - Clears direction bits

14.5.1.13 Port direction toggle (DIRNOT0)

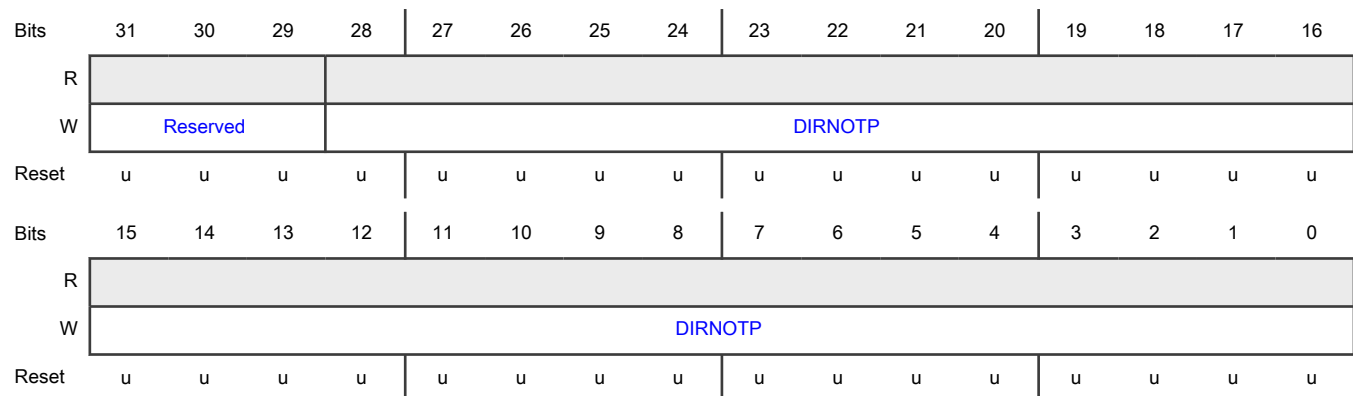
Offset

Register	Offset
DIRNOT0	2480h

Function

Direction bits can be toggled by writing ones to these write-only registers.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-0 DIRNOTP	Toggle direction bits. Toggle direction bits for each pin. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0_0000_0000_0000_0000_0000_0000_0000b - No operation 0_0000_0000_0000_0000_0000_0000_0001b - Toggles direction bit

14.5.1.14 Interrupt A enable control (INTENA0)

Offset

Register	Offset
INTENA0	2500h

Function

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt A.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0	Interrupt A enable bits.
INT_ENn	Interrupt enable control for each pin. 0b - Pin does not contribute to GPIO interrupt A 1b - Pin contributes to GPIO interrupt A

14.5.1.15 Interrupt B enable control (INTENB0)

Offset

Register	Offset
INTENB0	2580h

Function

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt B.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INT_ENn	Interrupt B enable bits. Interrupt enable control for each pin. 0b - Pin does not contribute to GPIO interrupt B 1b - Pin contributes to GPIO interrupt B

14.5.1.16 Interrupt polarity control (INTPOL0)**Offset**

Register	Offset
INTPOL0	2600h

Function

These registers select the polarity of GPIO interrupts.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_
W	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL9	CTL8	CTL7	CTL6	CTL5	CTL4	CTL3	CTL2	CTL1	CTL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 POL_CTLn	Polarity control for each pin Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - High level or rising edge triggered 1b - Low level or falling edge triggered

14.5.1.17 Interrupt edge select (INTEDG0)**Offset**

Register	Offset
INTEDG0	2680h

Function

These registers allow GPIO interrupts to be selected as level or edge triggered.

Diagram

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

Fields

Field	Function
31-0 EDGEN	Edge or level mode select bits. These bits allow GPIO interrupts to be selected as level or edge triggered. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Level mode 1b - Edge mode

14.5.1.18 Interrupt status for interrupt A (INTSTATA0)

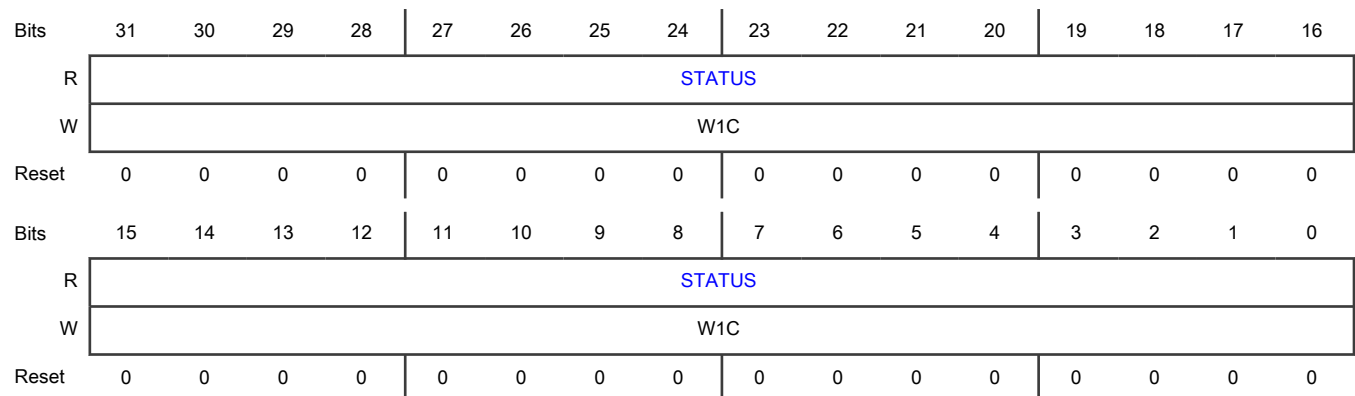
Offset

Register	Offset
INTSTATA0	2700h

Function

These registers report the status of interrupt group A for each GPIO port.

Diagram



Fields

Field	Function
31-0 STATUS	Interrupt status. Interrupt group A status bits Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read: 0 - not pending; Read: 1 - pending; Write: 0 - no action. Write: 1 - clear status bit.

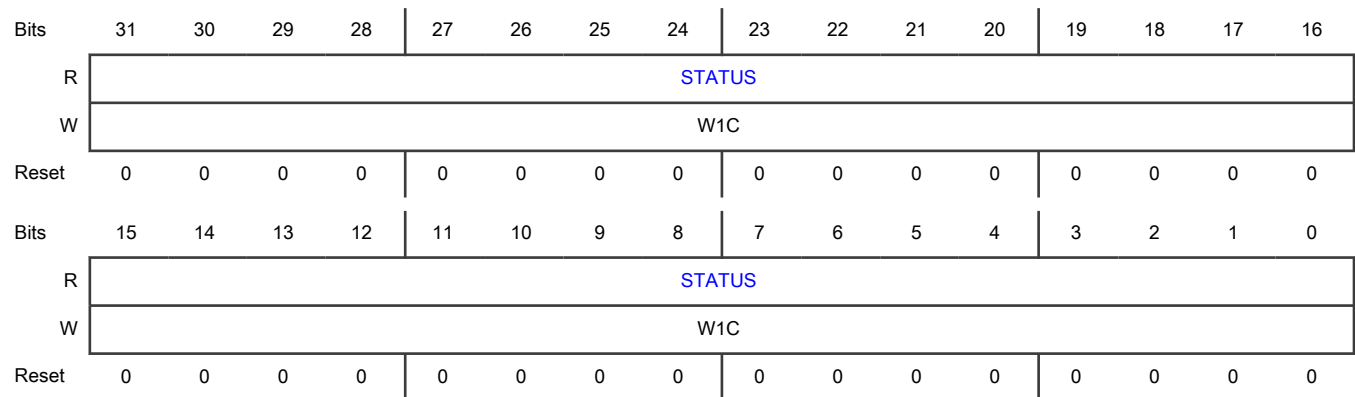
14.5.1.19 Interrupt status for interrupt B (INTSTATB0)

Offset

Register	Offset
INTSTATB0	2780h

Function

These registers report the status of interrupt group B for each GPIO port.

Diagram**Fields**

Field	Function
31-0 STATUS	Interrupt status Interrupt group B status bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read 0 - not pending; Read 1 - pending; Write 0 - no action. Write 1 - clear status bit.

14.5.2 GPIO General Purpose I/O (GPIO) register descriptions

GPIO port addresses can be read and written as bytes, halfwords, or words. Registers within each kind of GPIO are functionally the same.

NOTE

Supported pins depends on the specific device and package.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

14.5.2.1 GPIO memory map

GPIO base address: 4010_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h - DBh	Byte pin registers for all port GPIO pins (B0_0 - B6_27)	8	RW	See section
1000h - 136Ch	Word pin registers for all port GPIO pins (W0_0 - W6_27)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
2000h - 201Ch	Port direction (DIR0 - DIR7)	32	W	0000_0000h
2080h - 209Ch	Port mask (MASK0 - MASK7)	32	RW	0000_0000h
2100h - 211Ch	Port pin (PIN0 - PIN7)	32	RW	0000_0000h
2180h - 219Ch	Masked Port Pin (MPIN0 - MPIN7)	32	RW	0000_0000h
2200h - 221Ch	Port set (SET0 - SET7)	32	RW	0000_0000h
2280h - 229Ch	Port clear (CLR0 - CLR7)	32	RW	See section
2300h - 231Ch	Port toggle (NOT0 - NOT7)	32	W	See section
2380h - 239Ch	Port direction set (DIRSET0 - DIRSET7)	32	W	See section
2400h - 241Ch	Port direction clear (DIRCLR0 - DIRCLR7)	32	RW	See section
2480h - 249Ch	Port direction toggle (DIRNOT0 - DIRNOT7)	32	W	See section
2500h - 251Ch	Interrupt A enable control (INTENA0 - INTENA7)	32	RW	0000_0000h
2580h - 259Ch	Interrupt B enable control (INTENB0 - INTENB7)	32	RW	0000_0000h
2600h - 261Ch	Interrupt polarity control (INTPOL0 - INTPOL7)	32	RW	0000_0000h
2680h - 269Ch	Interrupt edge select (INTEDG0 - INTEDG7)	32	RW	0000_0000h
2700h - 271Ch	Interrupt status for interrupt A (INTSTATA0 - INTSTATA7)	32	RW	0000_0000h
2780h - 279Ch	Interrupt status for interrupt B (INTSTATB0 - INTSTATB7)	32	RW	0000_0000h

14.5.2.2 Byte pin registers for all port GPIO pins (B0_0 - B6_27)

Offset

Registers in this array exist only for the following combinations of index values.

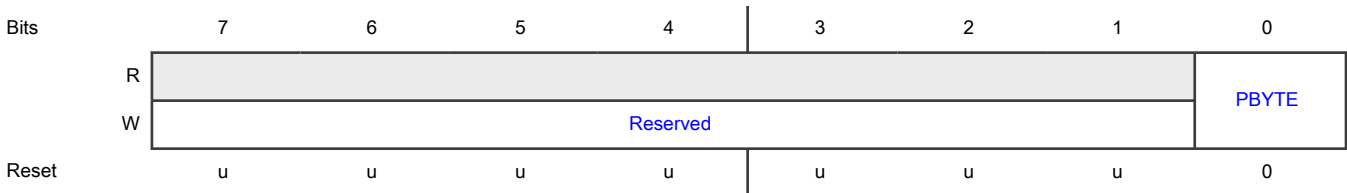
Index <i>a</i>	Index <i>b</i>
0–5	0–31
6	0–27

Register	Offset
Ba_b	0h + (a × 20h) + (b × 1h)

Function
Each GPIO pin has a byte register. Software reads and writes bytes to access individual pins. It can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

NOTE
The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram



Fields

Field	Function
7-1 —	Reserved (0 on read, ignored on write)
0 PBYTE	Port Byte Read- specifies the state of GPIO pins(PIOa_b), regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as 0. Write- loads the pin's output bit.

14.5.2.3 Word pin registers for all port GPIO pins (W0_0 - W6_27)

Offset

Registers in this array exist only for the following combinations of index values.

Index <i>a</i>	Index <i>b</i>
0–5	0–31
6	0–27

Register	Offset
Wa_b	1000h + (a × 80h) + (b × 4h)

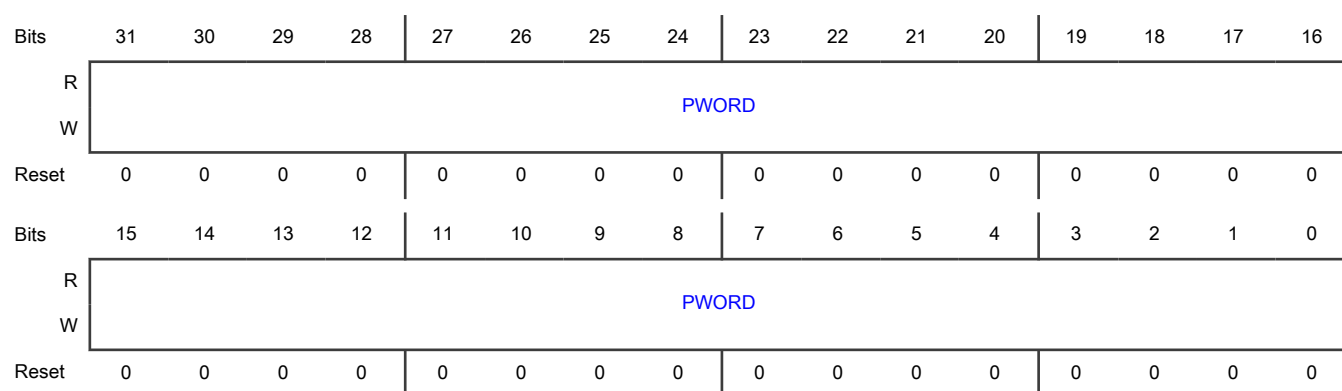
Function

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as zeros. Any write will clear the pin's output bit if the value written is all zeros, else it will set the pin's output bit.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram



Fields

Field	Function
31-0 PWORD	PWORD Read 0- pin PIOa_b is LOW. Write 0- clear output bit. Read 0xFFFF FFFF- pin PIOa_b is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF- sets output bit. Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit.

14.5.2.4 Port direction (DIR0 - DIR7)

Offset

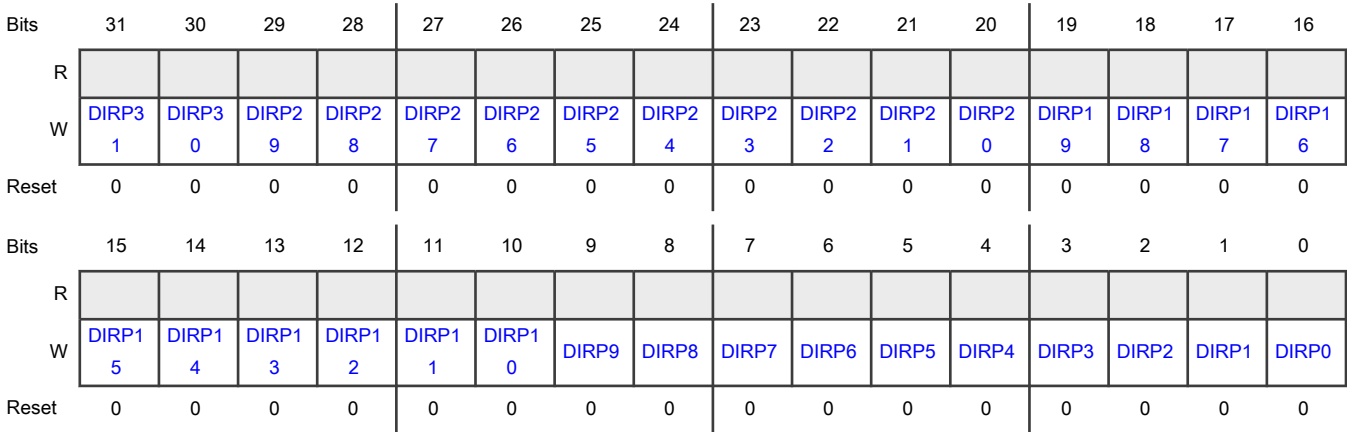
For a = 0 to 7:

Register	Offset
DIRa	2000h + (a × 4h)

Function

The DIR registers configure each pin in a GPIO port as input or output. Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Diagram



Fields

Field	Function
31-0	Selects pin direction for pin PIOa_b.
DIRPn	Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Input 1b - Output

14.5.2.5 Port mask (MASK0 - MASK7)

Offset

For a = 0 to 7:

Register	Offset
MASKa	2080h + (a × 4h)

Function

It defines which port pins are accessible in its MPIN register. Zeros in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK
W	P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK	MASK
W	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MASKPn	Port Mask Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Read MPIN: pin state; write MPIN: load output bit 1b - Read MPIN: 0; write MPIN: output bit not affected

14.5.2.6 Port pin (PIN0 - PIN7)**Offset**

For a = 0 to 7:

Register	Offset
PINa	2100h + (a × 4h)

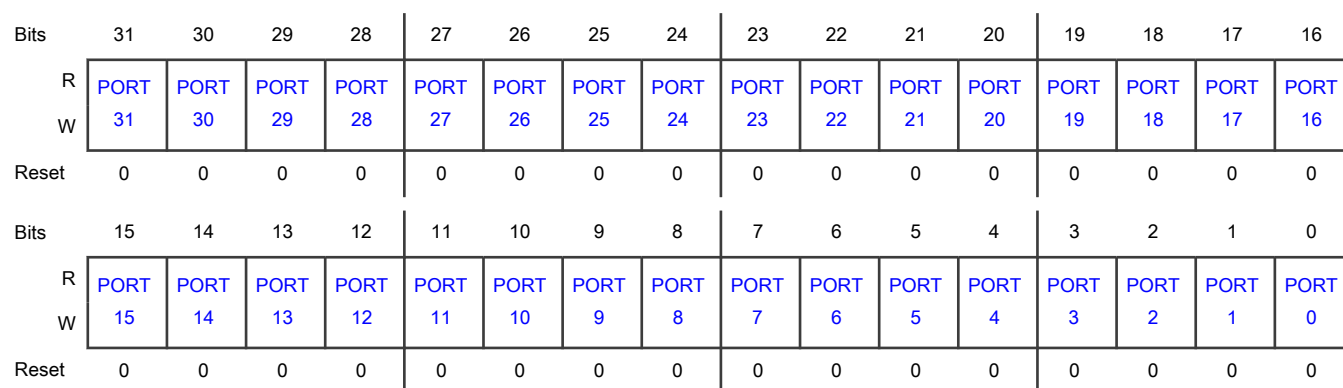
Function

Reading these registers returns the current state of the pins, regardless of direction, masking, or alternate digital functions. Pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Diagram



Fields

Field	Function
31-0 PORTn	Port pins Reads pin states or loads output bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Read- pin is low; Write- clear output bit 1b - Read- pin is high; Write- set output bit

14.5.2.7 Masked Port Pin (MPIN0 - MPIN7)

Offset

For a = 0 to 7:

Register	Offset
MPINa	2180h + (a × 4h)

Function

These registers are similar to the PIN registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR
W	TP31	TP30	TP29	TP28	TP27	TP26	TP25	TP24	TP23	TP22	TP21	TP20	TP19	TP18	TP17	TP16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR	MPOR
W	TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 MPORTPn	Mask bits for port pins Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. NOTE The data read after reset depends on the state of the pin, which in turn may depend on an external source. 0b - Read- pin is LOW and/or the corresponding bit in the MASK register is 1; write- clear output bit if the corresponding bit in the MASK register is 0 1b - Read- pin is HIGH and the corresponding bit in the MASK register is 0; write- set output bit if the corresponding bit in the MASK register is 0

14.5.2.8 Port set (SET0 - SET7)**Offset**

For a = 0 to 7:

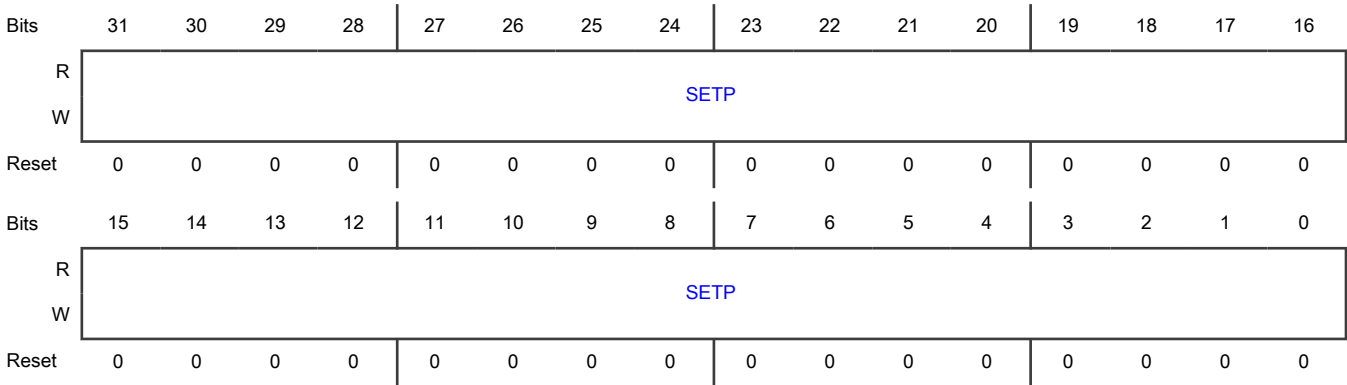
Register	Offset
SETa	2200h + (a × 4h)

Function

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram



Fields

Field	Function
31-0	Read or set output bits
SETP	Sets output bits for port pins 0000_0000_0000_0000_0000_0000_0000b - Read- output bit; write- no operation 0000_0000_0000_0000_0000_0000_0001b - Read- output bit; write- set output bit

14.5.2.9 Port clear (CLR0 - CLR7)

Offset

For a = 0 to 7:

Register	Offset
CLRa	2280h + (a × 4h)

Function

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLRP 31	CLRP 30	CLRP 29	CLRP 28	CLRP 27	CLRP 26	CLRP 25	CLRP 24	CLRP 23	CLRP 22	CLRP 21	CLRP 20	CLRP 19	CLRP 18	CLRP 17	CLRP 16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLRP 15	CLRP 14	CLRP 13	CLRP 12	CLRP 11	CLRP 10	CLRP 9	CLRP 8	CLRP 7	CLRP 6	CLRP 5	CLRP 4	CLRP 3	CLRP 2	CLRP 1	CLRP 0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 CLRPn	Clear output bits Clear output bits for port pins 0b - No operation 1b - Clears output bit

14.5.2.10 Port toggle (NOT0 - NOT7)

Offset

For a = 0 to 7:

Register	Offset
NOTa	2300h + (a × 4h)

Function

Output bits can be toggled by writing ones to these write-only registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

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

Fields

Field	Function
31-0 NOTPn	Toggle output bits Toggle direction bits for port pins. 0b - No operation 1b - Toggle output bit

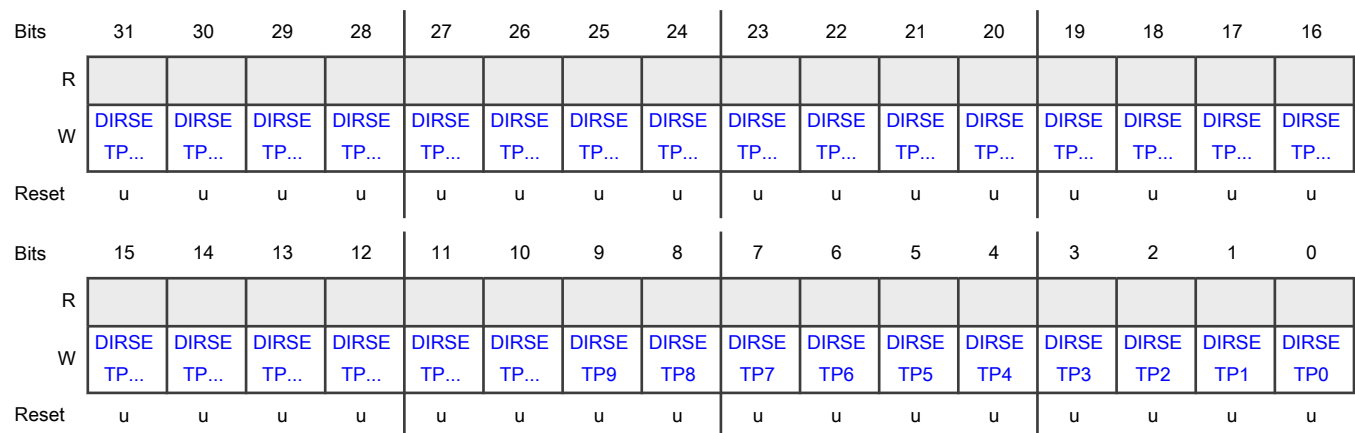
14.5.2.11 Port direction set (DIRSET0 - DIRSET7)**Offset**

For a = 0 to 7:

Register	Offset
DIRSETa	2380h + (a × 4h)

Function

Direction bits can be set by writing ones to these registers.

Diagram**Fields**

Field	Function
31-0 DIRSETPn	Direction set bits for Port pins Set pin direction for port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - No operation 1b - Sets direction bit

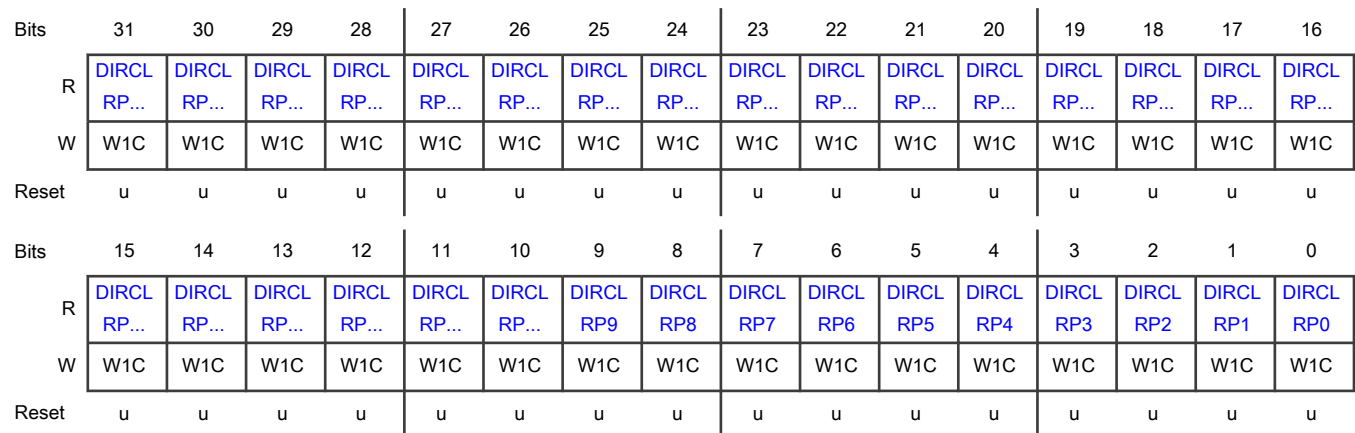
14.5.2.12 Port direction clear (DIRCLR0 - DIRCLR7)**Offset**

For a = 0 to 7:

Register	Offset
DIRCLRa	2400h + (a × 4h)

Function

Direction bits can be cleared by writing ones to these write-only registers.

Diagram**Fields**

Field	Function
31-0 DIRCLRPn	Clear direction bits. Clear pin direction bits for Port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - No operation 1b - Clears direction bits

14.5.2.13 Port direction toggle (DIRNOT0 - DIRNOT7)**Offset**

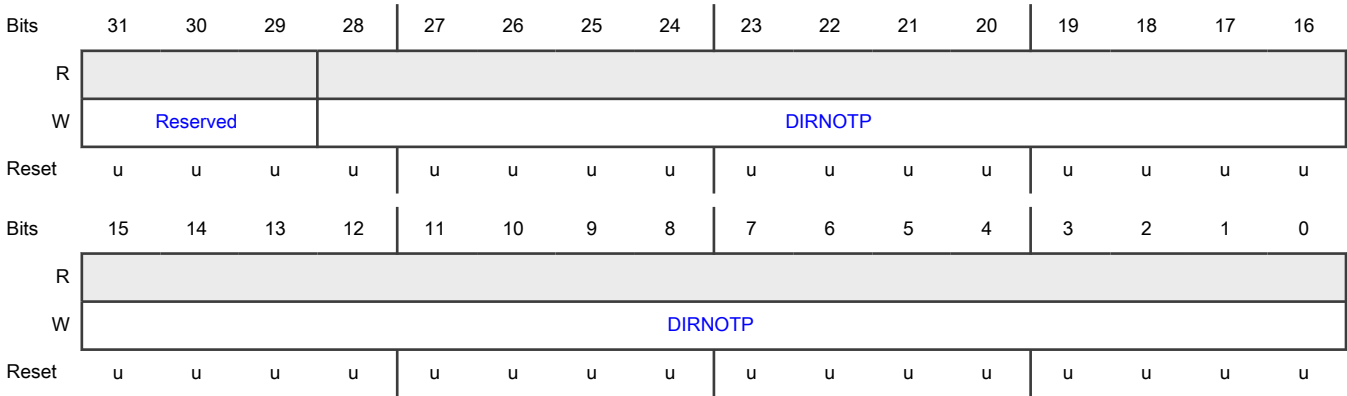
For a = 0 to 7:

Register	Offset
DIRNOTa	2480h + (a × 4h)

Function

Direction bits can be toggled by writing ones to these write-only registers.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-0 DIRNOTP	Toggle direction bits. Toggle direction bits for each pin. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0_0000_0000_0000_0000_0000_0000_0000b - No operation 0_0000_0000_0000_0000_0000_0000_0001b - Toggles direction bit

14.5.2.14 Interrupt A enable control (INTENA0 - INTENA7)

Offset

For a = 0 to 7:

Register	Offset
INTENAa	2500h + (a × 4h)

Function

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt A.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INT_ENn	Interrupt A enable bits. Interrupt enable control for each pin. 0b - Pin does not contribute to GPIO interrupt A 1b - Pin contributes to GPIO interrupt A

14.5.2.15 Interrupt B enable control (INTENB0 - INTENB7)**Offset**

For a = 0 to 7:

Register	Offset
INTENBa	2580h + (a × 4h)

Function

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt B.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_	INT_
W	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INT_ENn	Interrupt B enable bits. Interrupt enable control for each pin. 0b - Pin does not contribute to GPIO interrupt B 1b - Pin contributes to GPIO interrupt B

14.5.2.16 Interrupt polarity control (INTPOL0 - INTPOL7)

Offset

For a = 0 to 7:

Register	Offset
INTPOLa	2600h + (a × 4h)

Function

These registers select the polarity of GPIO interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_
W	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_	POL_
W	CTL...	CTL...	CTL...	CTL...	CTL...	CTL...	CTL9	CTL8	CTL7	CTL6	CTL5	CTL4	CTL3	CTL2	CTL1	CTL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 POL_CTLn	Polarity control for each pin Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - High level or rising edge triggered 1b - Low level or falling edge triggered

14.5.2.17 Interrupt edge select (INTEDG0 - INTEDG7)

Offset

For a = 0 to 7:

Register	Offset
INTEDGa	2680h + (a × 4h)

Function

These registers allow GPIO interrupts to be selected as level or edge triggered.

Diagram

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

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

Fields

Field	Function
31-0 EDGE _n	Edge or level mode select bits. These bits allow GPIO interrupts to be selected as level or edge triggered. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0b - Level mode 1b - Edge mode

14.5.2.18 Interrupt status for interrupt A (INTSTATA0 - INTSTATA7)

Offset

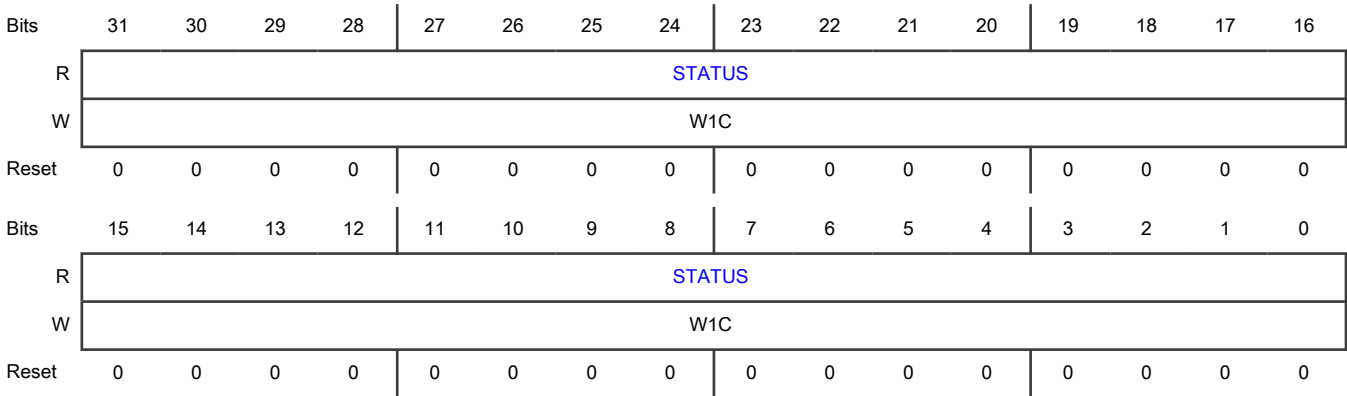
For a = 0 to 7:

Register	Offset
INTSTATAa	2700h + (a × 4h)

Function

These registers report the status of interrupt group A for each GPIO port.

Diagram



Fields

Field	Function
31-0 STATUS	Interrupt status. Interrupt group A status bits Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read: 0 - not pending; Read: 1 - pending; Write: 0 - no action. Write: 1 - clear status bit.

14.5.2.19 Interrupt status for interrupt B (INTSTATB0 - INTSTATB7)

Offset

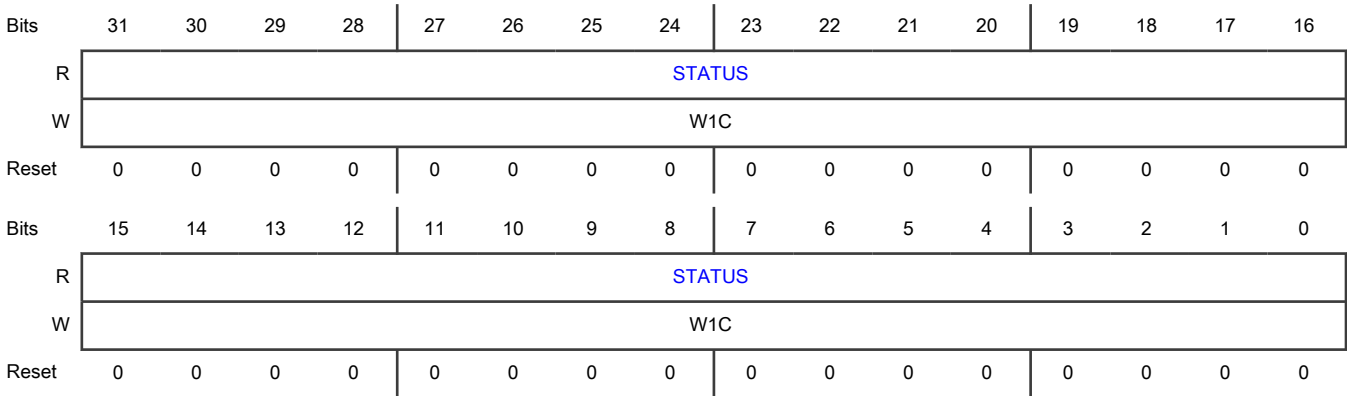
For a = 0 to 7:

Register	Offset
INTSTATBa	2780h + (a × 4h)

Function

These registers report the status of interrupt group B for each GPIO port.

Diagram



Fields

Field	Function
31-0 STATUS	Interrupt status Interrupt group B status bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read 0 - not pending; Read 1 - pending; Write 0 - no action. Write 1 - clear status bit.

Chapter 15

Pin Interrupt and Pattern Match (PINT)

15.1 Chip-specific PINT information

Table 56. Reference links to related information

Topic	Related module	Reference
Full description	PINT	PINT
System memory map		System memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing

15.1.1 Module instances

The chip includes one instance of PINT module, PINT.

15.1.2 Initialization

- Select up to eight external interrupt pins from all available digital port pins on ports 0 and 1 in INPUTMUX. The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
- Enable the clock to the PINT module via GPIOINTCTL bit in the PSCCTL2 register (See CLKCTL).
- Clear the PINT peripheral reset via GPIOINTCTL bit in the PRSTCTL2 register by writing to the PRSTCTL2_CLR register (See RSTCTL).

15.1.2.1 Pin Interrupts Initialization

In addition to the initialization steps described above, for Pin Interrupts initialization do the following -

- Each selected pin interrupt is assigned to one interrupt in the NVIC (see NVIC). Note that all PINT slot (PINT 0~7) can be used by CM33 or DSP.

15.1.2.2 Pattern Match Initialization

In addition to the initialization steps described above, for Pattern match engine initialization do the following -

- Each bit slice of the pattern match engine is assigned to one interrupt (see NVIC). Note that only the first 4 are available to the Cortex M33, which is present on selected devices.

15.1.2.3 Configure pins as pin interrupts or as inputs to the pattern match engine

1. Determine the pins that you want to serve as pin interrupts or pattern match inputs. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin selected, program the GPIO port pin number from ports 0 and 1 into one of the eight PINT_SEL registers in the INPUTMUX module.

NOTE

The port pin number serves to identify the pin to the PINT_SEL register. Any function, including GPIO, can be assigned to this pin via IOPCTL.

3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can be set up.

NOTE

The inputs to the Pin interrupt select registers bypass the IOPCTL function selection. They do not have to be selected as GPIO in IOPCTL. Make sure that no analog function is selected on pins that are input to the pin interrupts.

15.2 Overview

PINT uses standard GPIO functions as inputs. Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine.

15.2.1 Features

- Pin interrupts -
 - Up to 8 pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
 - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
 - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine -
 - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
 - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
 - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU.
 - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

15.2.2 Block diagram

From all available GPIO pins, up to 8 pins can be selected in the INPUTMUX module to serve as external interrupt pins (see INPUTMUX for more details). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

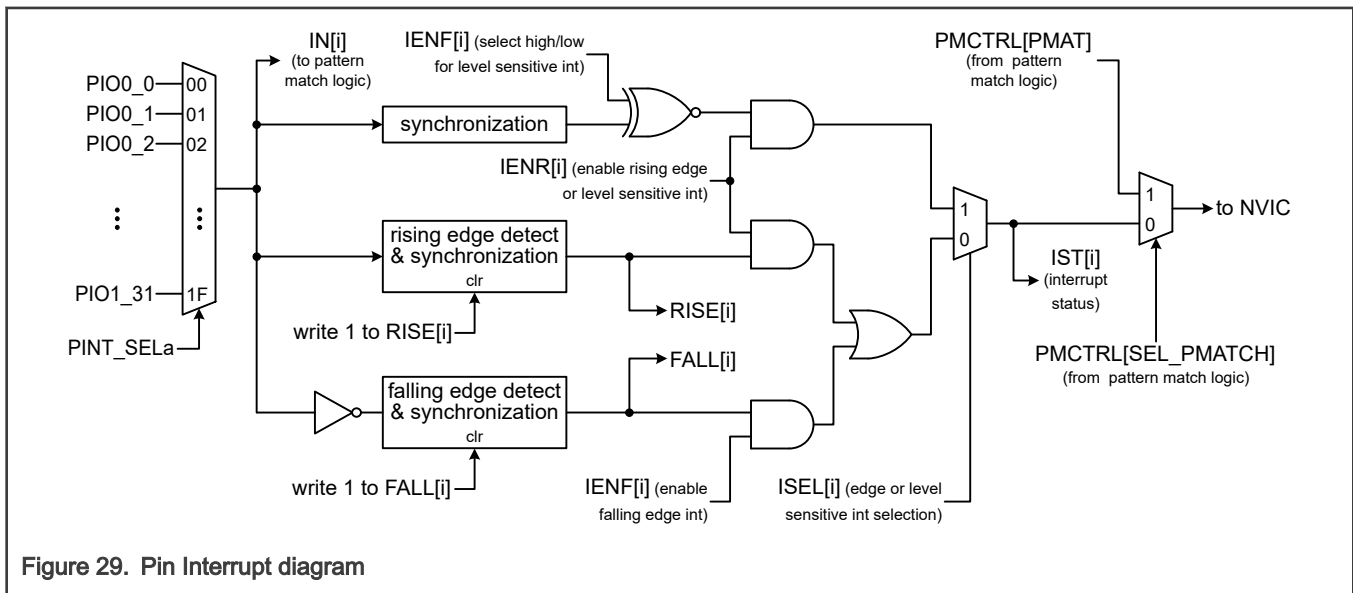


Figure 29. Pin Interrupt diagram

15.3 Signals

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the INPUTMUX. See INPUTMUX for more details.

15.4 Functional description

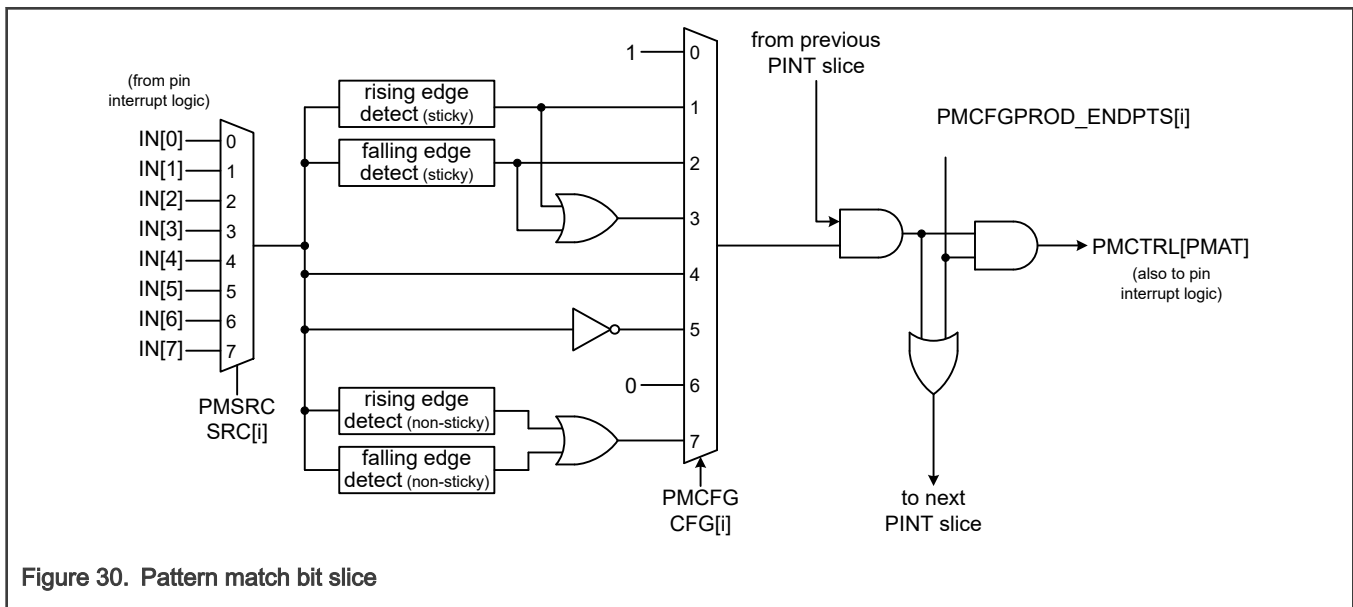
15.4.1 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of 8 GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and detection logic that monitors the selected input continuously and creates a HIGH output (true). If the input qualifies as detected, several terms can be combined to a minterm, and a pin interrupt is asserted when the minterm evaluates as true.

The detection logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detection logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detection logic output goes HIGH. This bit is cleared after one clock cycle, and the detection logic can detect another edge.
- Level: A HIGH or LOW level on the selected input.

The figure below shows the details of the edge detection logic for each slice.



Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See [Pattern match engine edge detect examples](#) for details.

The following figure shows connections between the pins and the pattern match engine. All pins that are inputs to the pattern match engine are selected in the SYSTCL block and can be GPIO port pins or other pin functions depending on the IOPCTL configuration.

NOTE

The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

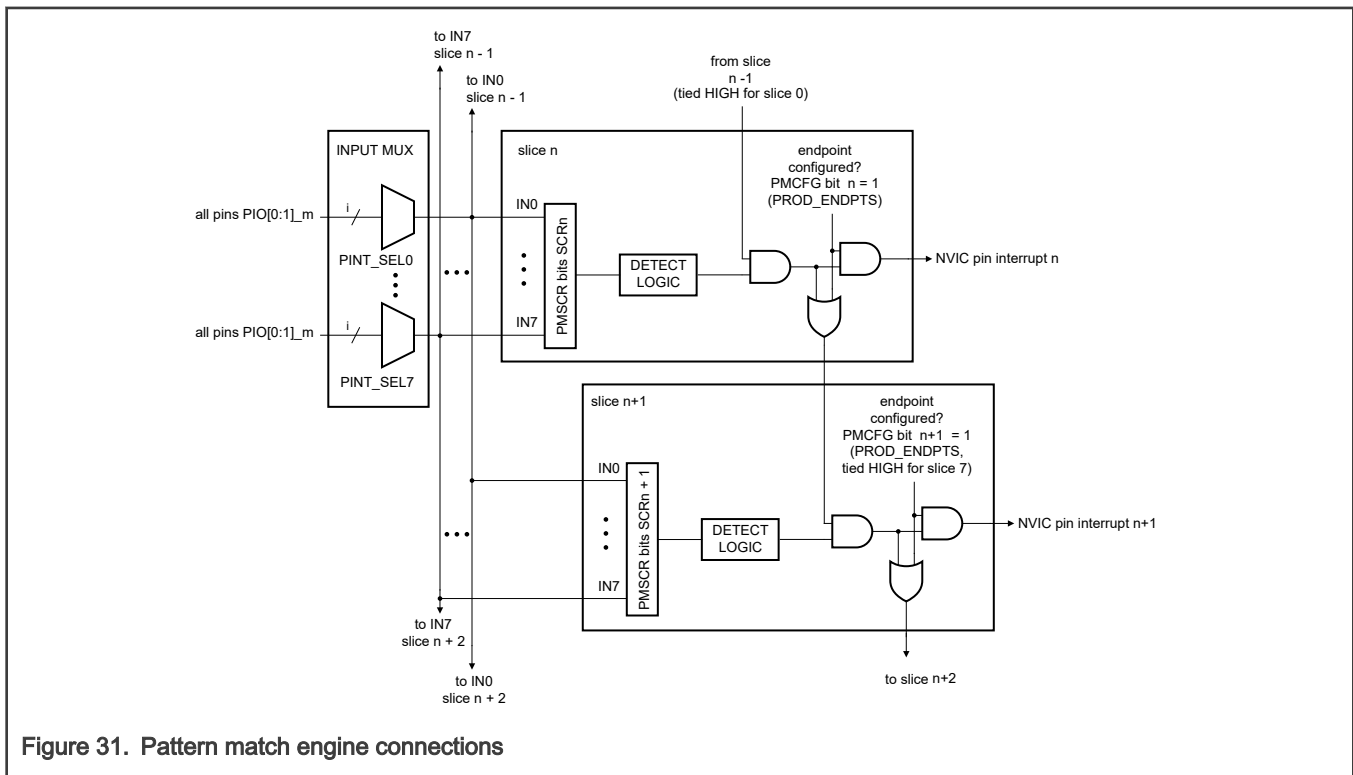


Figure 31. Pattern match engine connections

The pattern match logic continuously monitors the 8 inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM core when the entire boolean expression is true, when any minterm is matched).

The pattern match function utilizes the same 8 interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

NOTE

Pattern matching cannot be used to wake the part up from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of 8 bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

15.5 Application information

The following expression is specified through the registers [Pattern Match Interrupt Bit-Slice Source \(PMSRC\)](#) and [Pattern Match Interrupt Bit Slice Configuration \(PMCFG\)](#) :

```
IN0 AND NOT IN1 AND IN3 rising edge OR IN1 AND IN2 OR IN0 AND NOT IN3 AND NOT IN4
```

Each term in the boolean expression, IN0, NOT IN1, IN3 rising edge, etc., represents one bit slice of the pattern match engine.

- In the first AND function IN0 AND NOT IN1 AND IN3 rising edge, bit slice 0 monitors for a high-level on input IN0, bit slice 1 monitors for a low level on input IN1 and bit slice 2 monitors for a rising-edge on input IN3. If this combination is detected (all three terms are true) the interrupt associated with bit slice 2 will be asserted.
- In the second AND function IN1 AND IN2, bit slice 3 monitors input IN1 for a high level, bit slice 4 monitors input IN2 for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third AND function IN0 AND NOT IN3 AND NOT IN4, bit slice 5 monitors input IN0 for a high level, bit slice 6 monitors input IN3 for a low level, and bit slice 7 monitors input IN4 for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three AND functions asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

15.5.1 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:

$$(IN1) + (IN1 * IN2) + (\sim IN2 * \sim IN3 * IN6fe) + (IN5 * IN7ev)$$

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register [Pattern Match Interrupt Bit-Slice Source \(PMSRC\)](#)
 - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
 - SRC0: 001 - select input 1 for bit slice 0
 - SRC1: 001 - select input 1 for bit slice 1
 - SRC2: 010 - select input 2 for bit slice 2
 - SRC3: 010 - select input 2 for bit slice 3
 - SRC4: 011 - select input 3 for bit slice 4
 - SRC5: 110 - select input 6 for bit slice 5
 - SRC6: 101 - select input 5 for bit slice 6
 - SRC7: 111 - select input 7 for bit slice 7
- PMCFG register [Pattern Match Interrupt Bit Slice Configuration \(PMCFG\)](#)
 - PROD_ENDPTS0 = 1
 - PROD_ENDPTS2 = 1
 - PROD_ENDPTS5 = 1

- All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
- PROD_ENDPTS= 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
- CFG0: 000 - high level on the selected input (input 1) for bit slice 0
- CFG1: 000 - high level on the selected input (input 1) for bit slice 1
- CFG2: 000 - high level on the selected input (input 2) for bit slice 2
- CFG3: 101 - low level on the selected input (input 2) for bit slice 3
- CFG4: 101 - low level on the selected input (input 3) for bit slice 4
- CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
- CFG6: 000 - high level on the selected input (input 5) for bit slice 6
- CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register [Pattern Match Interrupt Control \(PMCTRL\)](#)
 - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the standard pin interrupt mechanism.

For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

Pin interrupt 2 will be asserted in response to a match on the second product term.

Pin interrupt 5 will be asserted when there is a match on the third product term.

Pin interrupt 7 will be asserted on a match on the last term.
 - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
 - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
 - The remaining bits will always be low.

15.5.2 Pattern match engine edge detect examples

The figures below show pattern match functionality only; accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

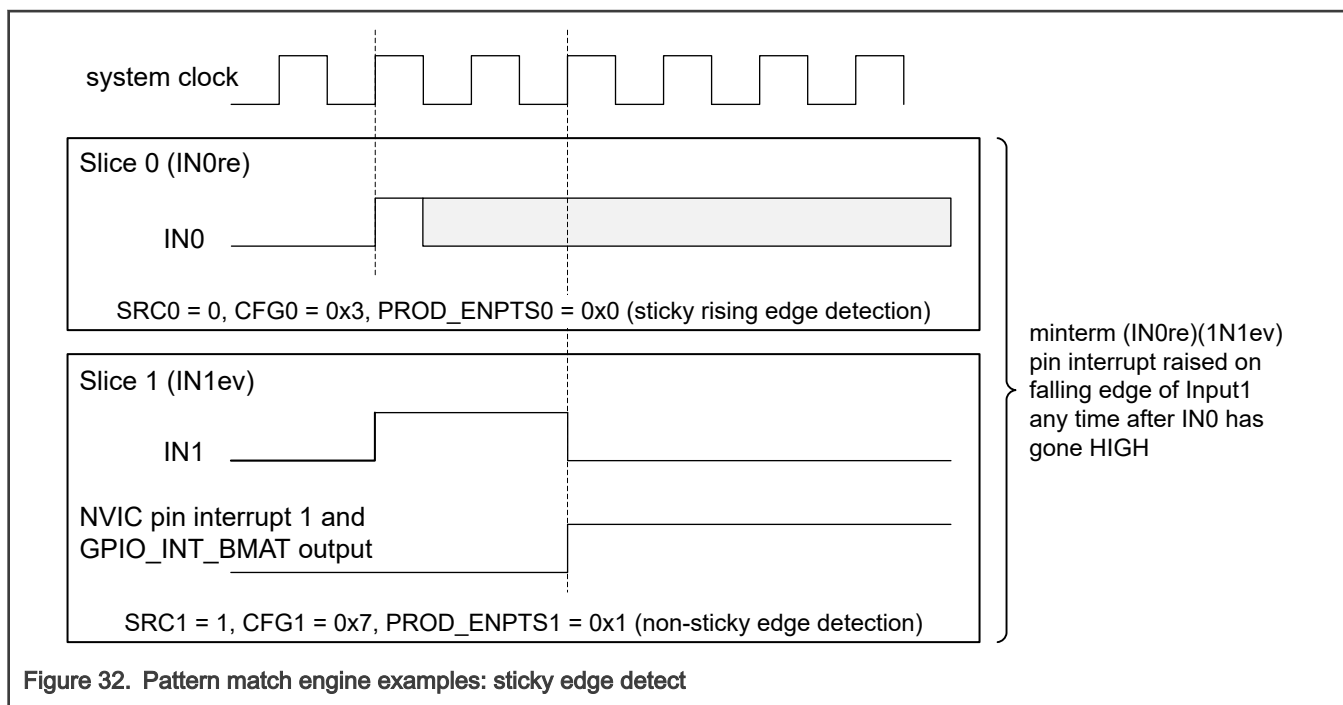


Figure 32. Pattern match engine examples: sticky edge detect

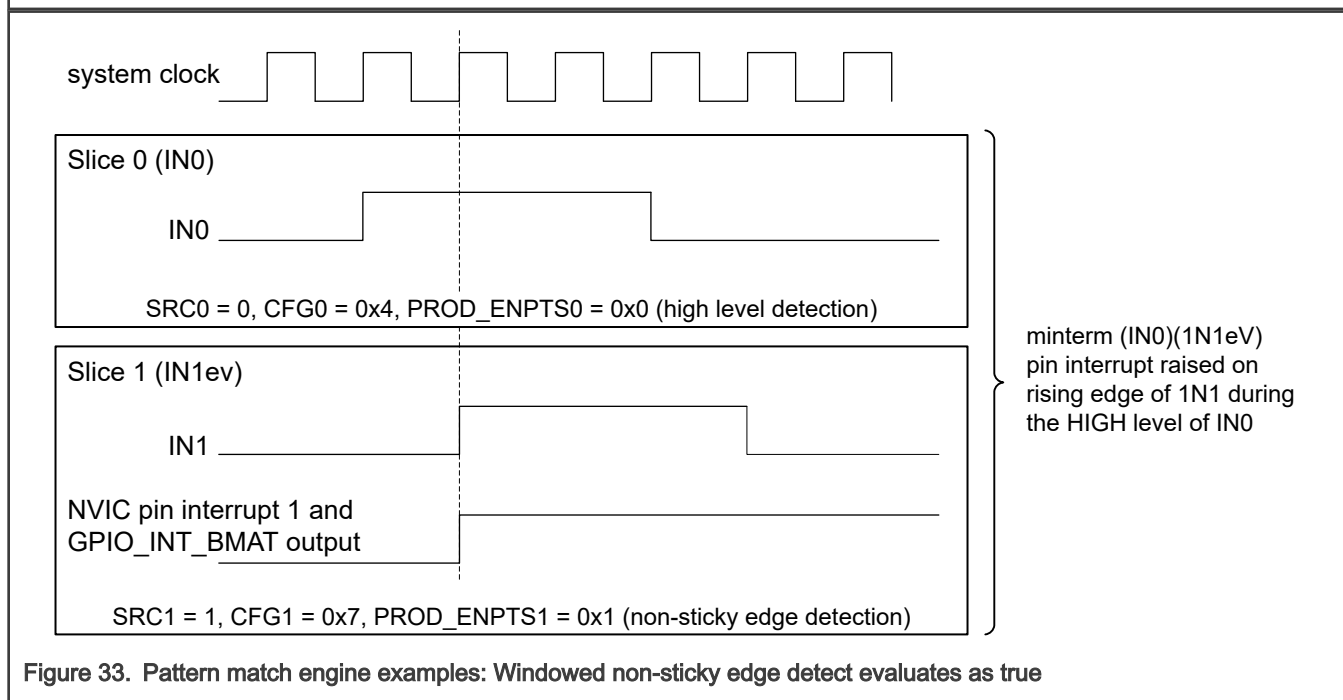
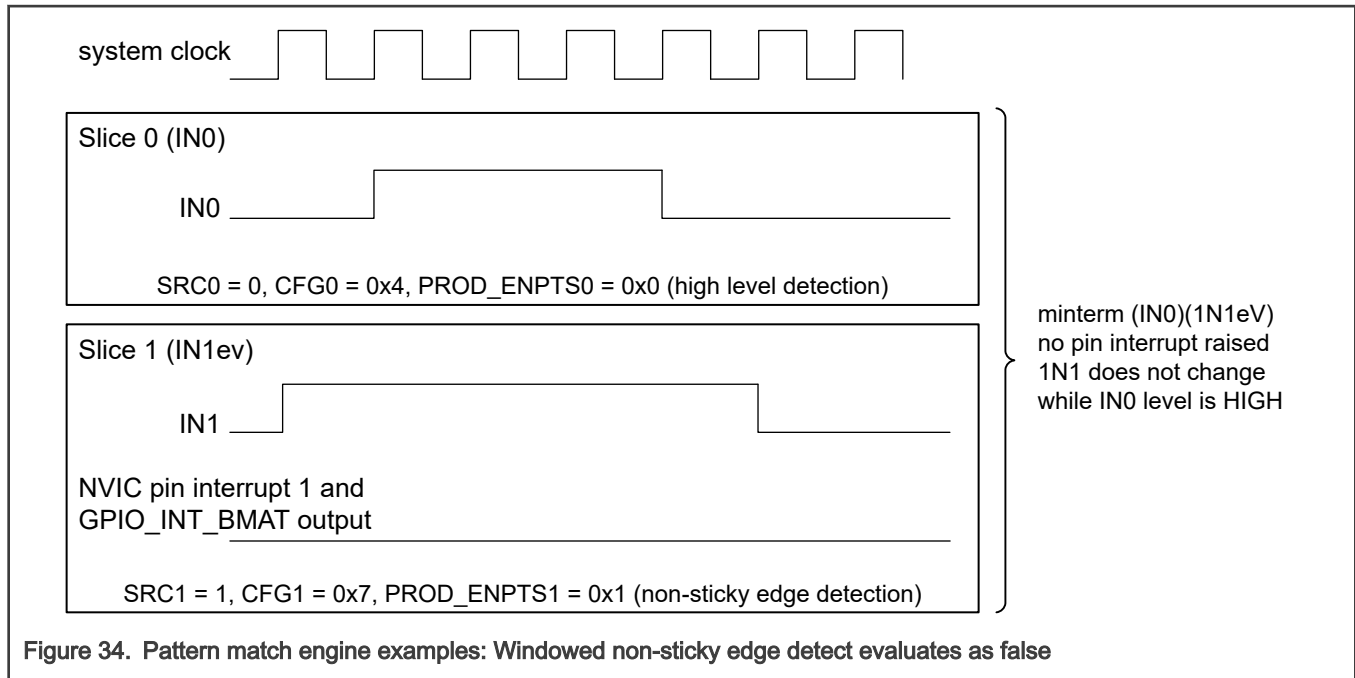


Figure 33. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true



15.6 Memory Map and register definition

This section includes the PINT memory map and detailed descriptions of all registers.

The chip includes, up to 8 pins identified as interrupt sources by the Pin Interrupt Select registers (PINT_SEL_a). The registers have 8 bitfields that correspond to the pins called out by the PINT_SEL registers.

Table 57. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

15.6.1 Pin Interrupts and Pattern Match register descriptions

15.6.1.1 PINT memory map

PINT base address: 4002_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Pin Interrupt Mode (ISEL)	32	RW	See section
4h	Pin Interrupt Level or Rising Edge Interrupt Enable (IENR)	32	RW	See section
8h	Pin Interrupt Level or Rising Edge Interrupt Set (SIENR)	32	RW	See section
Ch	Pin Interrupt Level (Rising Edge Interrupt) Clear (CIENR)	32	RW	See section
10h	Pin Interrupt Active Level or Falling Edge Interrupt Enable (IENF)	32	RW	See section
14h	Pin Interrupt Active Level or Falling Edge Interrupt Set (SIENF)	32	RW	See section
18h	Pin Interrupt Active Level or Falling Edge Interrupt Clear (CIENF)	32	RW	See section
1Ch	Pin Interrupt Rising Edge (RISE)	32	RW	See section
20h	Pin Interrupt Falling Edge (FALL)	32	RW	See section
24h	Pin Interrupt Status (IST)	32	RW	See section
28h	Pattern Match Interrupt Control (PMCTRL)	32	RW	See section
2Ch	Pattern Match Interrupt Bit-Slice Source (PMSRC)	32	RW	See section
30h	Pattern Match Interrupt Bit Slice Configuration (PMCFG)	32	RW	See section

15.6.1.2 Pin Interrupt Mode (ISEL)

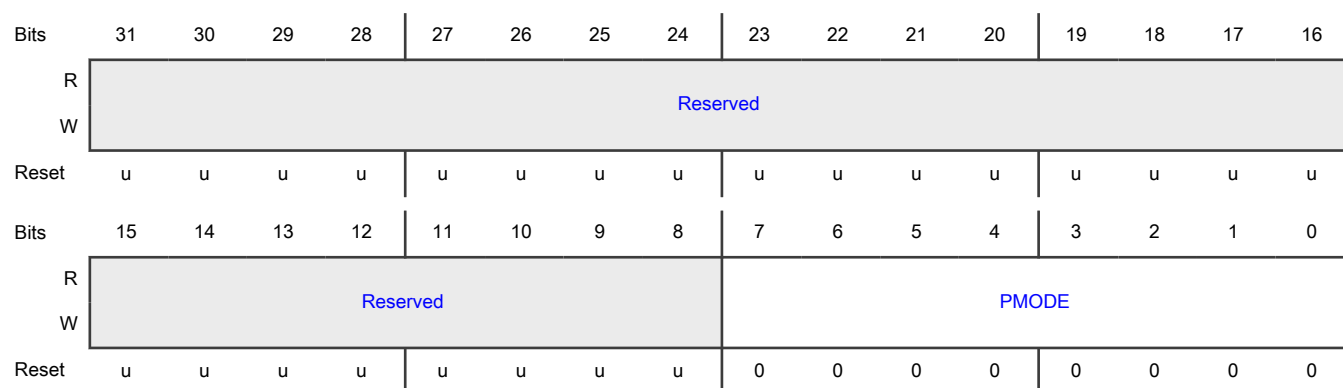
Offset

Register	Offset
ISEL	0h

Function

The ISEL register determines whether an interrupt is edge or level sensitive.

Diagram



Fields

Field	Function
31-8 —	Read value is undefined, only zero should be written.
7-0 PMODE	Interrupt mode Selects the interrupt mode for each pin interrupt. Bit a configures the pin interrupt selected in PINT_SEL _a register in INPUTMUX. 0000_0000b - Edge-sensitive 0000_0001b - Level-sensitive

15.6.1.3 Pin Interrupt Level or Rising Edge Interrupt Enable (IENR)

Offset

Register	Offset
IENR	4h

Function

Each bit in the IENR register enables the interrupt depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Diagram

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

Fields

Field	Function
31-8	Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7-0 ENRL	Enable Interrupt Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINT_SEL _a . 0000_0000b - Disable rising edge or level interrupt 0000_0001b - Enable rising edge or level interrupt

15.6.1.4 Pin Interrupt Level or Rising Edge Interrupt Set (SIENR)

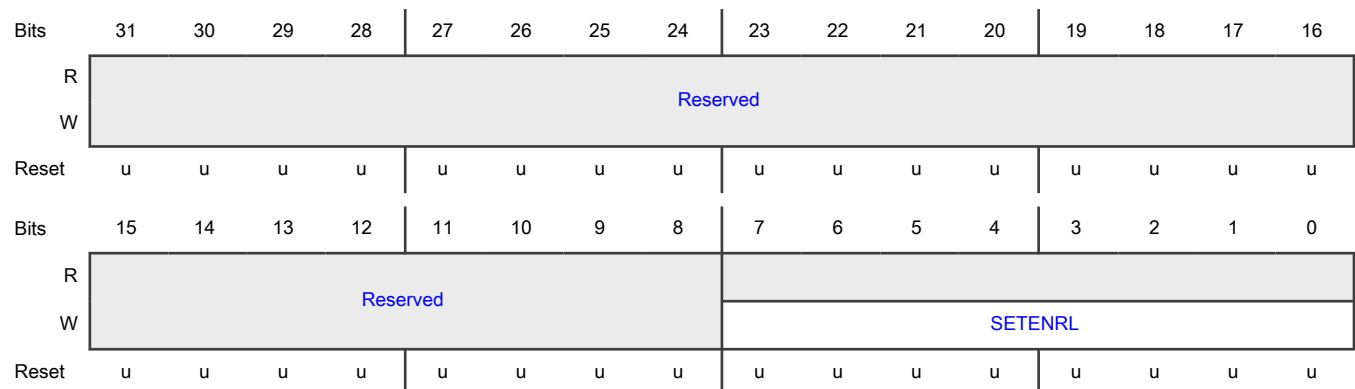
Offset

Register	Offset
SIENR	8h

Function

Each bit in the SIENR register sets the corresponding bit in the IENR register depending on the PMODE configured in the ISEL register.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SETENRL	Set bits in the IENR

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Ones written to this address set bits in the IENR, thus enabling interrupts. Bit a sets bit n in the IENR register. 0000_0000b - No operation 0000_0001b - Enable rising edge or level interrupt

15.6.1.5 Pin Interrupt Level (Rising Edge Interrupt) Clear (CIENR)

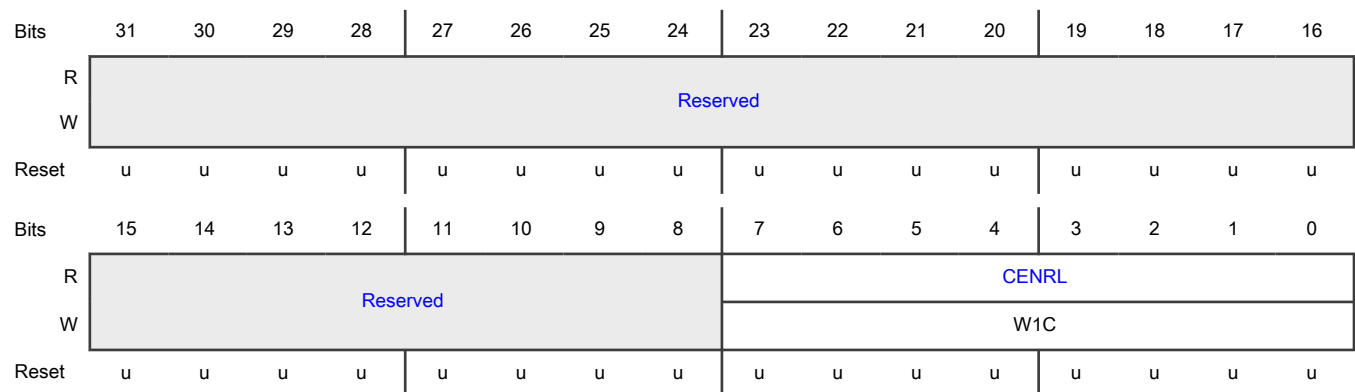
Offset

Register	Offset
CIENR	Ch

Function

Each bit in the CIENR register clears the corresponding bit in the IENR register depending on the PMODE configured in the ISEL register. Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit a clears bit n in the IENR register.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 CENRL	Clear bits in the IENR 0000_0000b - No operation 0000_0001b - Disable rising edge or level interrupt

15.6.1.6 Pin Interrupt Active Level or Falling Edge Interrupt Enable (IENF)

Offset

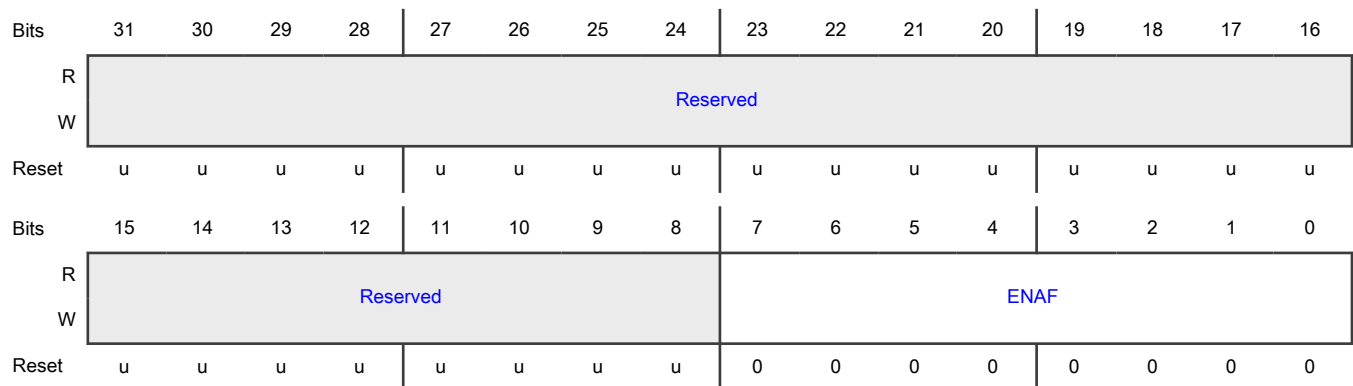
Register	Offset
IENF	10h

Function

Each bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 ENAF	Enable Interrupt Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit a configures the pin interrupt selected in PINT_SEL _a . 0000_0000b - Disable falling edge interrupt or set active interrupt level LOW 0000_0001b - Enable falling edge interrupt enabled or set active interrupt level HIGH

15.6.1.7 Pin Interrupt Active Level or Falling Edge Interrupt Set (SIENF)

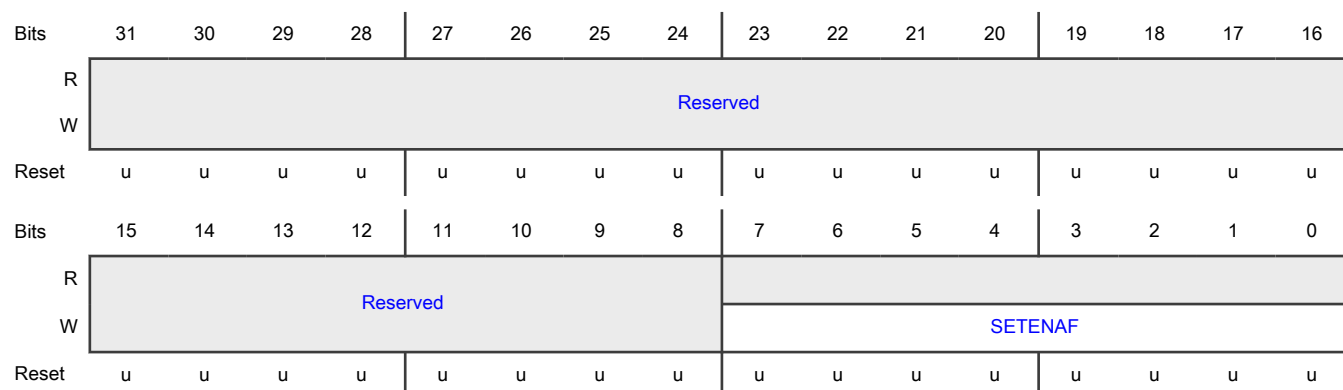
Offset

Register	Offset
SIENF	14h

Function

Each bit in the SIENF register sets the corresponding bit in the IENF register depending on how PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is set.
- If the PMODE is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 SETENAF	Set bits in the IENF Ones written to this address set bits in the IENF, thus enabling interrupts. Bit a sets bit n in the IENF register. 0000_0000b - No operation 0000_0001b - Select HIGH-active interrupt or enable falling edge interrupt

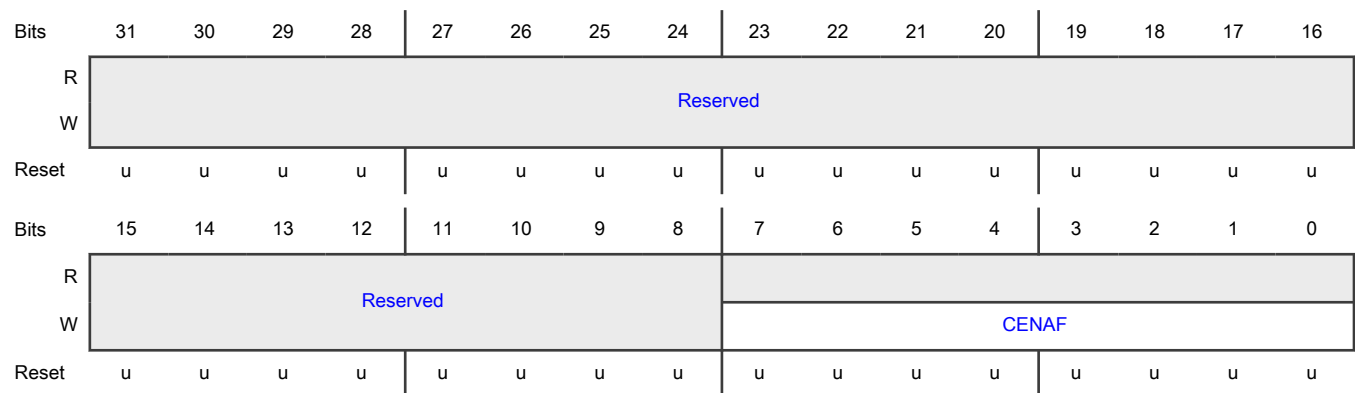
15.6.1.8 Pin Interrupt Active Level or Falling Edge Interrupt Clear (CIENF)**Offset**

Register	Offset
CIENF	18h

Function

Each bit in the CIENF register sets the corresponding bit in the IENF register depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the PMODE is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Diagram**Fields**

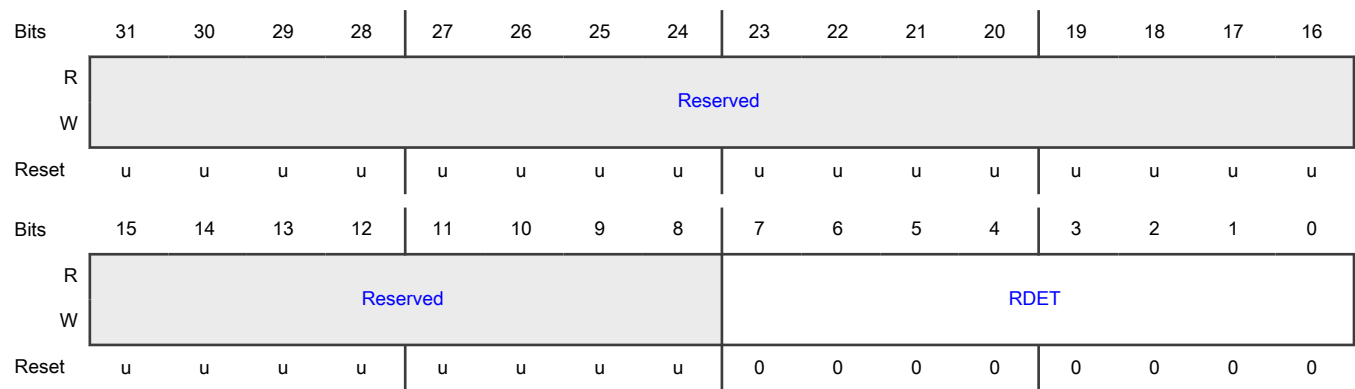
Field	Function
31-8 —	Reserved
7-0 CENAF	Clear bits in the IENF Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit a clears bit n in the IENF register. 0000_0000b - No operation 0000_0001b - LOW-active interrupt selected or falling edge interrupt disabled

15.6.1.9 Pin Interrupt Rising Edge (RISE)**Offset**

Register	Offset
RISE	1Ch

Function

This register contains ones for pin interrupts selected in the PINT_SEL_a registers (see INPUTMUX) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINT_SEL registers, regardless of whether they are interrupt-enabled.

Diagram**Fields**

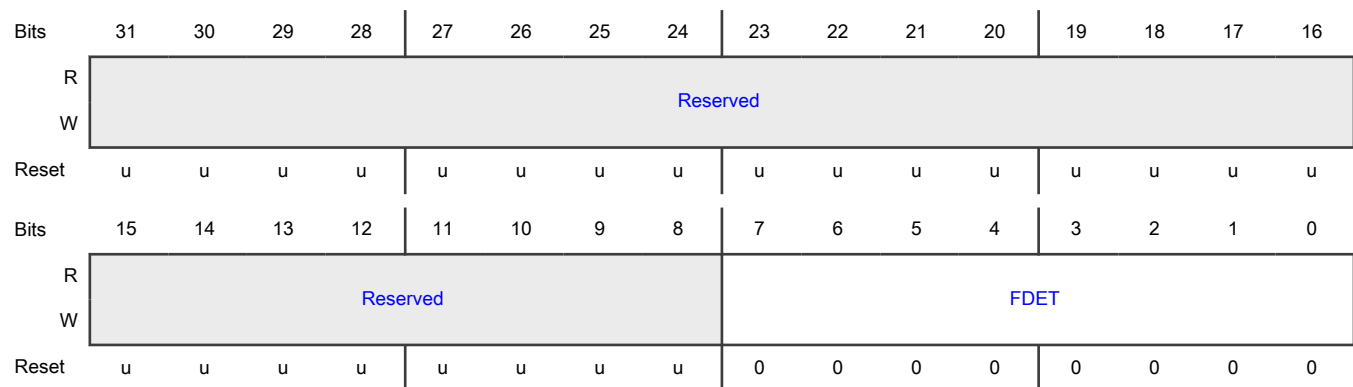
Field	Function
31-8 —	Reserved
7-0 RDET	Rising edge detect Bit a detects the rising edge of the pin selected in PINT_SEL_a. 0000_0000b - Read 0- No rising edge has been detected on this pin since Reset or the last time a one was written to this bit, Write 0- no operation 0000_0001b - Read 1- a rising edge has been detected since Reset or the last time a one was written to this bit, Write 1- clear rising edge detection for this pin

15.6.1.10 Pin Interrupt Falling Edge (FALL)**Offset**

Register	Offset
FALL	20h

Function

It contains ones for pin interrupts selected in the PINT_SEL_a registers on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINT_SEL_a registers, regardless of whether they are interrupt-enabled.

Diagram**Fields**

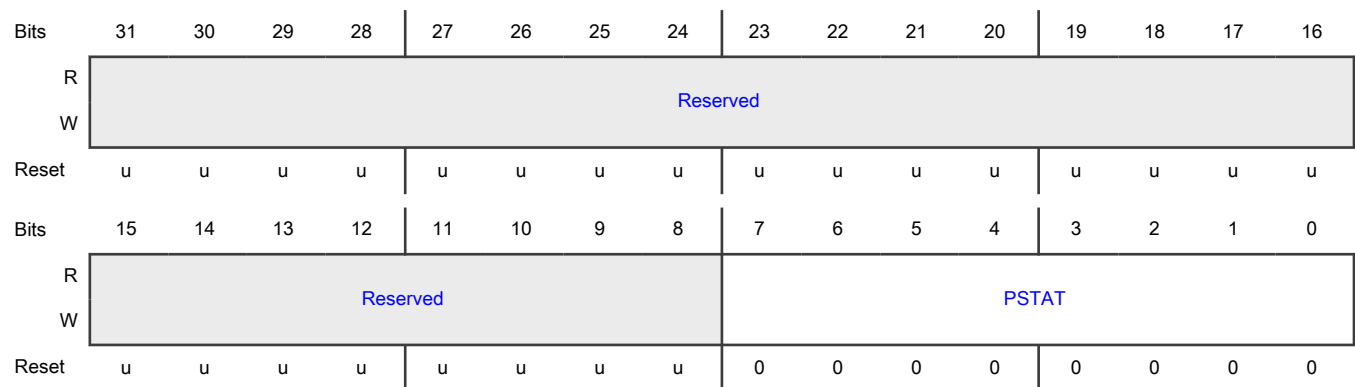
Field	Function
31-8 —	Reserved
7-0 FDET	Falling edge detect Bit a detects the falling edge of the pin selected in PINT_SEL _a . 0000_0000b - Read 0- No falling edge has been detected on this pin since Reset or the last time a one was written to this bit, Write 0- no operation 0000_0001b - Read 1- a falling edge has been detected since Reset or the last time a one was written to this bit, Write 1- clear falling edge detection for this bit

15.6.1.11 Pin Interrupt Status (IST)**Offset**

Register	Offset
IST	24h

Function

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 PSTAT	Pin interrupt status Bit a returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINT_SEL_a. 0000_0000b - Read 0- interrupt is not being requested for this pin, Write 0- no operation. 0000_0001b - Read 1- interrupt is being requested for this pin, Write 1 (edge-sensitive)- clear rising- and falling-edge detection for this pin, Write 1 (level-sensitive)- switch the active level for this pin (in the IENF register).

15.6.1.12 Pattern Match Interrupt Control (PMCTRL)**Offset**

Register	Offset
PMCTRL	28h

Function

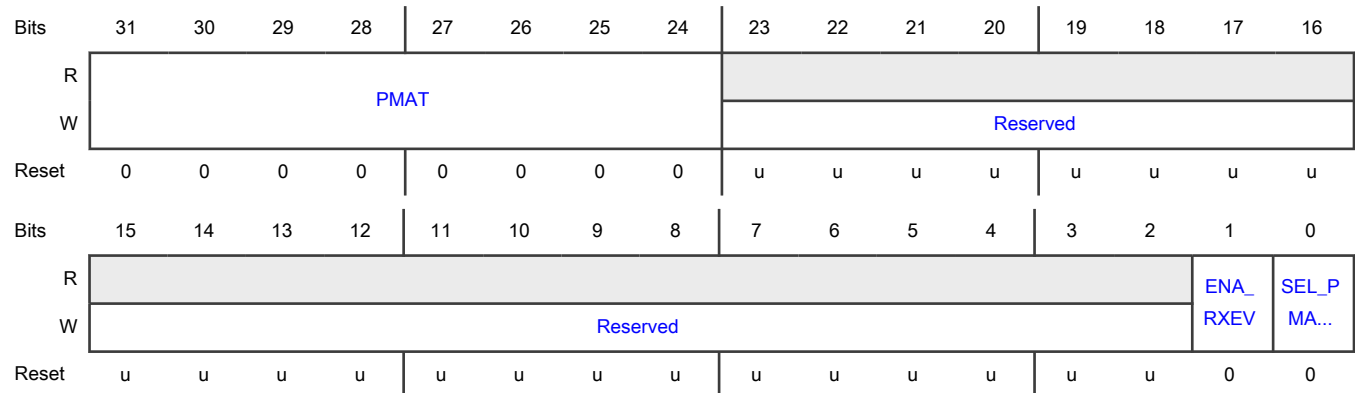
The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read. If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

NOTE

Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

NOTE

The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Diagram**Fields**

Field	Function
31-24 PMAT	Pattern Matches Displays the current state of pattern matches. 0000_0001b - The corresponding product term is matched by the current state of the appropriate inputs.
23-2 —	Do not write 1s to unused bits.
1 ENA_RXEV	Enables the RXEV output to the CPU and/or to a GPIO output, when the specified boolean expression evaluates to true. 0b - Disabled- RXEV output to the CPU is disabled. 1b - Enabled- RXEV output to the CPU is enabled.
0 SEL_PMATCH	Specifies whether the pin interrupts are controlled by the pin interrupt function or by the pattern match function. 0b - Pin interrupt- interrupts are driven in response to the standard pin interrupt function. 1b - Pattern match- interrupts are driven in response to pattern matches.

15.6.1.13 Pattern Match Interrupt Bit-Slice Source (PMSRC)**Offset**

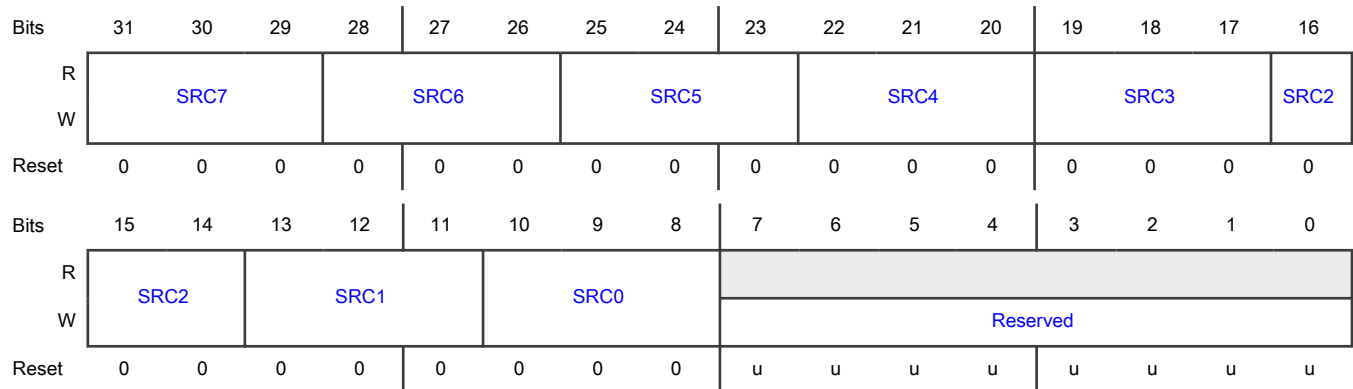
Register	Offset
PMSRC	2Ch

Function

This register specifies the input source for each of the pattern match bit slices. Each of the possible 8 inputs is selected in the pin interrupt select registers in INPUTMUX. Input 0 corresponds to the pin selected in the PINT_SEL0 register, input 1 corresponds to the pin selected in the PINT_SEL1 register, and so on.

NOTE

Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Diagram**Fields**

Field	Function
31-29: SRC7	Selects the input source for bit slice n
28-26: SRC6	000b - Input 0. Selects the pin selected in the PINT_SEL0 register as the source to bit slice n.
25-23: SRC5	001b - Input 1. Selects the pin selected in the PINT_SEL1 register as the source to bit slice n.
22-20: SRC4	010b - Input 2. Selects the pin selected in the PINT_SEL2 register as the source to bit slice n.
19-17: SRC3	011b - Input 3. Selects the pin selected in the PINT_SEL3 register as the source to bit slice n.
16-14: SRC2	100b - Input 4. Selects the pin selected in the PINT_SEL4 register as the source to bit slice n.
13-11: SRC1	101b - Input 5. Selects the pin selected in the PINT_SEL5 register as the source to bit slice n.
10-8: SRC0	110b - Input 6. Selects the pin selected in the PINT_SEL6 register as the source to bit slice n. 111b - Input 7. Selects the pin selected in the PINT_SEL7 register as the source to bit slice n.
7-0	Software should not write 1s to unused bits.
—	

15.6.1.14 Pattern Match Interrupt Bit Slice Configuration (PMCFG)**Offset**

Register	Offset
PMCFG	30h

Function

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

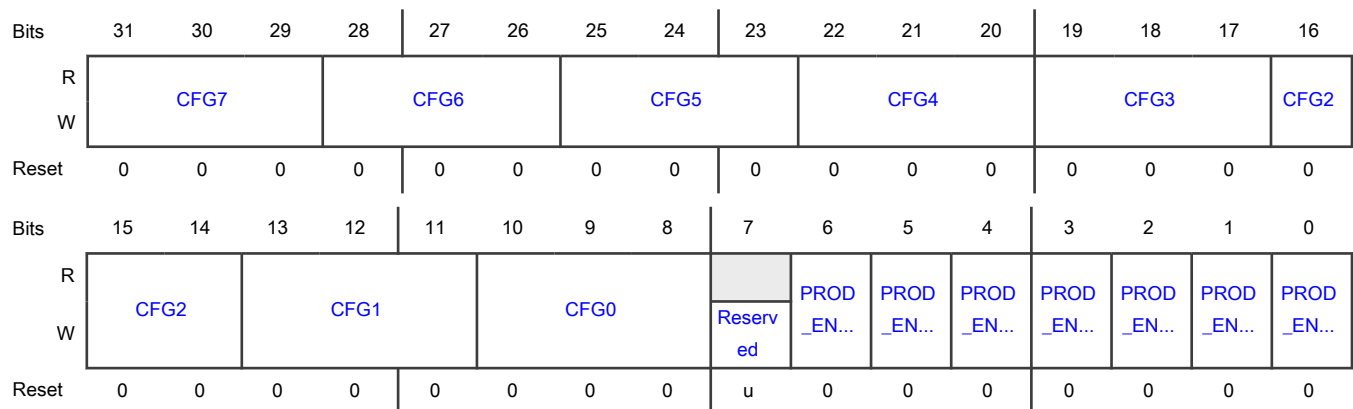
- **Sticky:** A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- **Non-sticky:** Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge.

To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTSa bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression, an OR will be inserted in the boolean expression following the element controlled by this bit slice.

Diagram



Fields

Field	Function
31-29: CFG7	Specifies the match contribution condition for bit slice n.
28-26: CFG6	000b - Constant HIGH. This bit slice always contributes to a product term match.
25-23: CFG5	001b - Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to.
22-20: CFG4	
19-17: CFG3	010b - Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to.
16-14: CFG2	
13-11: CFG1	

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-8: CFG0	011b - Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to. 100b - High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. 101b - Low level. Match occurs when there is a low level on the specified input. 110b - Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). 111b - Event. Non-sticky rising or falling edge. Match occurs on an event when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after 1 clock cycle.
7 —	Bit slice 7 is automatically considered a product end point.
6-0 PROD_ENDPT Sn	Determines whether slice n is an endpoint. 0b - No effect. Slice 0 is not an endpoint. 1b - Endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.

Chapter 16

DMA Controller

16.1 Chip-specific DMA controller information

Table 58. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller	DMA Controller
System memory map		System memory map
	INPUTMUX	INPUTMUX
	System controller	SYSCTL
	Reset controller	RSTCTL
	Clock controller	CLKCTL
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

16.1.1 Module Instances

The chip includes two DMA controllers. Each has the same DMA request and trigger input possibilities.

16.1.2 Channel Descriptors for DMA channels

Table 59. Channel Descriptor offsets for DMA channels on the chip

Descriptor	Offset
Channel Descriptor for DMA channel 0	0x000
Channel Descriptor for DMA channel 1	0x010
Channel Descriptor for DMA channel 2	0x020
Channel Descriptor for DMA channel 3	0x030
Channel Descriptor for DMA channel 4	0x040
Channel Descriptor for DMA channel 5	0x050
Channel Descriptor for DMA channel 6	0x060
Channel Descriptor for DMA channel 7	0x070
Channel Descriptor for DMA channel 8	0x080
Channel Descriptor for DMA channel 9	0x090
Channel Descriptor for DMA channel 10	0x0A0

Table continues on the next page...

Table 59. Channel Descriptor offsets for DMA channels on the chip (continued)

Descriptor	Offset
Channel Descriptor for DMA channel 11	0x0B0
Channel Descriptor for DMA channel 12	0x0C0
Channel Descriptor for DMA channel 13	0x0D0
Channel Descriptor for DMA channel 14	0x0E0
Channel Descriptor for DMA channel 15	0x0F0
Channel Descriptor for DMA channel 16	0x100
Channel Descriptor for DMA channel 17	0x110
Channel Descriptor for DMA channel 18	0x120
Channel Descriptor for DMA channel 19	0x130
Channel Descriptor for DMA channel 20	0x140
Channel Descriptor for DMA channel 21	0x150
Channel Descriptor for DMA channel 22	0x160
Channel Descriptor for DMA channel 23	0x170
Channel Descriptor for DMA channel 24	0x180
Channel Descriptor for DMA channel 25	0x190
Channel Descriptor for DMA channel 26	0x1A0
Channel Descriptor for DMA channel 27	0x1B0
Channel Descriptor for DMA channel 28	0x1C0
Channel Descriptor for DMA channel 29	0x1D0
Channel Descriptor for DMA channel 30	0x1E0
Channel Descriptor for DMA channel 31	0x1F0
Channel Descriptor for DMA channel 32	0x200
Channel Descriptor for DMA channel 33	0x210
Channel Descriptor for DMA channel 34	0x220
Channel Descriptor for DMA channel 35	0x230
Channel Descriptor for DMA channel 36	0x240

16.1.3 DMA requests

See section [DMA with I2C monitor mode](#) for more details.

the trigger sources go into a 27-input/37-output periphinmux. Any of the 27 inputs can be routed to any of the 37 DMA channels. Trigger PMUX outputs 0-36 connecting to DMA channels 0-36.

NOTE

Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

	INPUT SOURCES
0	FLEXCOM0_RX_DMA
1	FLEXCOM0_TX_DMA
2	FLEXCOM1_RX_DMA
3	FLEXCOM1_TX_DMA
4	FLEXCOM2_RX_DMA
5	FLEXCOM2_TX_DMA
6	FLEXCOM3_RX_DMA
7	FLEXCOM3_TX_DMA
8	FLEXCOM4_RX_DMA
9	FLEXCOM4_TX_DMA
10	FLEXCOM5_RX_DMA
11	FLEXCOM5_TX_DMA
12	FLEXCOM6_RX_DMA
13	FLEXCOM6_TX_DMA
14	FLEXCOM7_RX_DMA
15	FLEXCOM7_TX_DMA
16	DMIC_CH0_DMA/FLEXCOM8_RX_DMA
17	DMIC_CH1_DMA/FLEXCOM8_TX_DMA
18	DMIC_CH2_DMA/FLEXCOM9_RX_DMA
19	DMIC_CH3_DMA/FLEXCOM9_TX_DMA
20	DMIC_CH4_DMA/FLEXCOM10_RX_DMA
21	DMIC_CH5_DMA/FLEXCOM10_TX_DMA
22	DMIC_CH6_DMA/FLEXCOM13_RX_DMA
23	DMIC_CH7_DMA/FLEXCOM13_TX_DMA
24	I3C0_RX_DMA
25	I3C0_TX_DMA
26	FLEXCOM14_RX_DMA

Table continues on the next page...

Table continued from the previous page...

27	FLEXCOM14_TX_DMA
28	FLEXCOM16_RX_DMA
29	FLEXCOM16_TX_DMA
30	I3C1_RX_DMA
31	I3C1_TX_DMA
32	FLEXCOM11_RX_DMA
33	FLEXCOM11_TX_DMA
34	FLEXCOM12_RX_DMA
35	FLEXCOM12_TX_DMA
36	HASHCRYPT_IN_DMA

16.1.4 DMA Trigger Muxes

Table 60. DMA Trigger Input

Triggers	Input Sources	Software trigger
0	GPIO_INT0	Yes
1	GPIO_INT1	Yes
2	GPIO_INT2	Yes
3	GPIO_INT3	Yes
4	CTIMER0_DMAREQ_M0	Yes
5	CTIMER0_DMAREQ_M1	Yes
6	CTIMER1_DMAREQ_M0	Yes
7	CTIMER1_DMAREQ_M1	Yes
8	CTIMER2_DMAREQ_M0	Yes
9	CTIMER2_DMAREQ_M1	Yes
10	CTIMER3_DMAREQ_M0	Yes
11	CTIMER3_DMAREQ_M1	Yes
12	CTIMER4_DMAREQ_M0	Yes
13	CTIMER4_DMAREQ_M1	Yes
14	DMA _n _TRIGOUT_A	Yes
15	DMA _n _TRIGOUT_B	Yes
16	DMA _n _TRIGOUT_C	Yes
17	DMA _n _TRIGOUT_D	Yes
18	SCT_DMA_REQ0	Yes

Table continues on the next page...

Table 60. DMA Trigger Input (continued)

Triggers	Input Sources	Software trigger
19	SCT_DMA_REQ1	Yes
20	HASHCRYPT_OUT_DMA	Yes
21	ACMP_DMA	Yes
22	FlexSPI0_RX_DMA, also requires DMA request fixed on channel 22	Yes
23	FlexSPI0_TX_DMA, also requires DMA request fixed on channel 23	Yes
24	ADC_DMA	Yes
25	FlexSPI1_RX_DMA, also requires DMA request fixed on channel 24	Yes
26	FlexSPI1_TX_DMA, also requires DMA request fixed on channel 25	Yes

Table 61. DMA Trigger Output

Triggers	Output Sources
0	DMA _n _CH0_TRIGOUT
1	DMA _n _CH1_TRIGOUT
2	DMA _n _CH2_TRIGOUT
3	DMA _n _CH3_TRIGOUT
4	DMA _n _CH4_TRIGOUT
5	DMA _n _CH5_TRIGOUT
6	DMA _n _CH6_TRIGOUT
7	DMA _n _CH7_TRIGOUT
8	DMA _n _CH8_TRIGOUT
9	DMA _n _CH9_TRIGOUT
10	DMA _n _CH10_TRIGOUT
11	DMA _n _CH11_TRIGOUT
12	DMA _n _CH12_TRIGOUT
13	DMA _n _CH13_TRIGOUT
14	DMA _n _CH14_TRIGOUT
15	DMA _n _CH15_TRIGOUT
16	DMA _n _CH16_TRIGOUT
17	DMA _n _CH17_TRIGOUT
18	DMA _n _CH18_TRIGOUT

Table continues on the next page...

Table 61. DMA Trigger Output (continued)

Triggers	Output Sources
19	DMA _n _CH19_TRIGOUT
20	DMA _n _CH20_TRIGOUT
21	DMA _n _CH21_TRIGOUT
22	DMA _n _CH22_TRIGOUT
23	DMA _n _CH23_TRIGOUT
24	DMA _n _CH24_TRIGOUT
25	DMA _n _CH25_TRIGOUT
26	DMA _n _CH26_TRIGOUT
27	DMA _n _CH27_TRIGOUT
28	DMA _n _CH28_TRIGOUT
29	DMA _n _CH29_TRIGOUT
30	DMA _n _CH30_TRIGOUT
31	DMA _n _CH31_TRIGOUT
32	DMA _n _CH32_TRIGOUT
33	DMA _n _CH33_TRIGOUT
34	DMA _n _CH34_TRIGOUT
35	DMA _n _CH35_TRIGOUT
36	DMA _n _CH36_TRIGOUT

16.1.5 Chip-specific DMA Initialization

It is assumed that system security is put in place prior to initializing the DMA controllers. So DMA0 will be secured and selected memory spaces and peripherals secured if that is the intended use model.

Configure each DMA controller as follows:

- Use the CLKCTL1_PSCCTL1 register (See CLKCTL) to enable the clock to the DMA registers interface.
- Clear the appropriate DMA peripheral reset in the RSTCTL1_PRSTCTL1 register by writing to the RSTCTL1_PRSTCTL1_CLR register (See RSTCTL).
- The DMA controllers provide interrupts to the NVIC, (see NVIC). These interrupts can alternatively be connected to the DSP (See INPUTMUX).
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in [DMA Requests](#) and [Hardware Triggers](#).
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signalled by one of the hardware triggers, through the input mux registers DMA_ITRIG_SEL (See INPUTMUX).
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_OTRIG_SEL (See INPUTMUX) as inputs to DMA triggers.

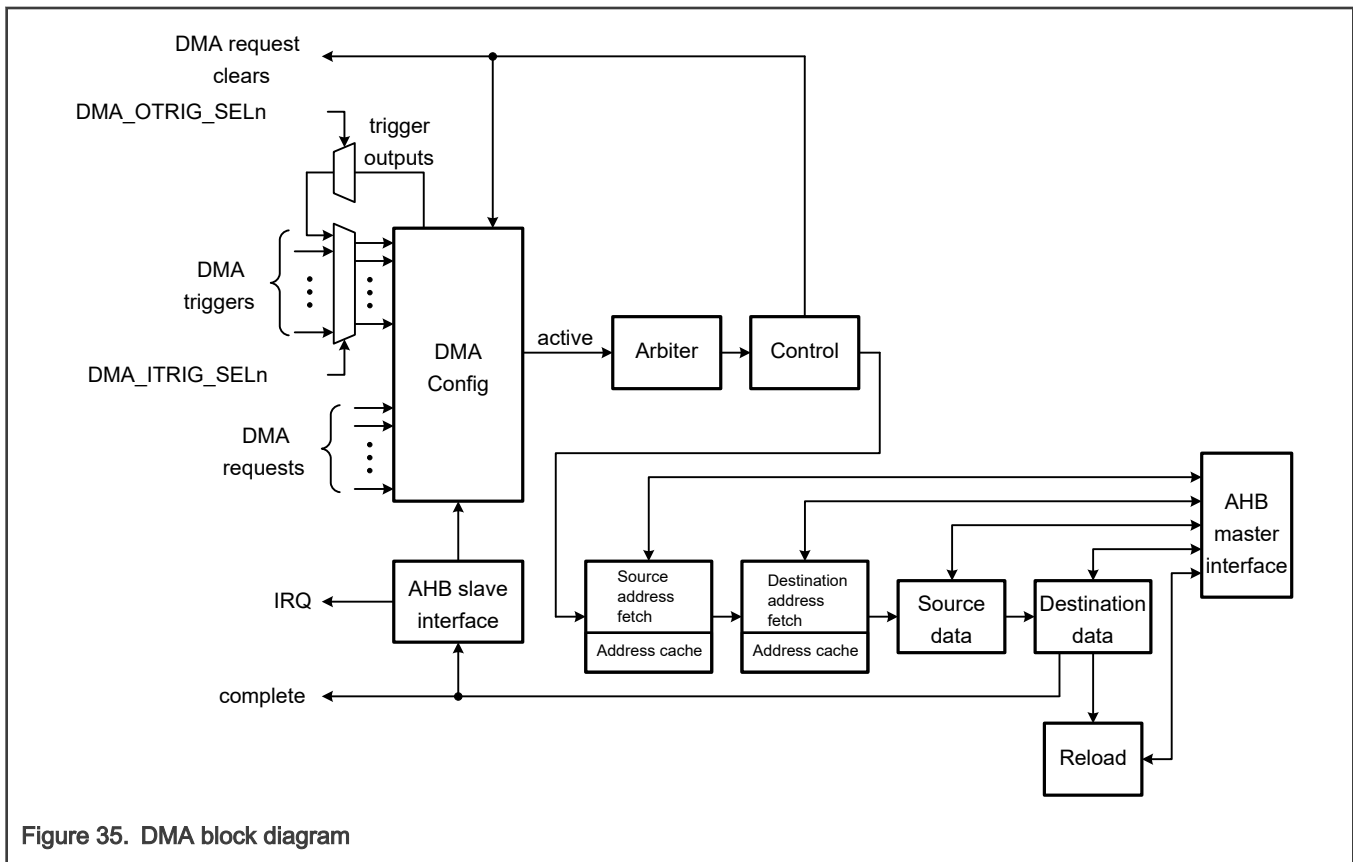
- The HWWAKE feature may be used in conjunction with certain peripherals to allow DMA to operate while the device is in the deep-sleep reduced power mode, without waking up the CPU. (See SYSCTL).
- The DMA controllers have an AHB master function. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSCTL0_AHBMATRIXPRIOR register (See SYSCTL).

For details on the trigger input and output multiplexing, see INPUTMUX.

16.2 Overview

This section provides an introduction to the DMA Controller.

16.2.1 Block diagram



16.2.2 Features

DMA controller has the features below:

- 37 channels, 37 of which are connected to fixed peripheral DMA requests, dedicated to that particular channel. DMA channels with no DMA request connected or that are otherwise unused can be used for functions such as memory-to-memory moves, or triggered transfers.
- 27 potential DMA triggers for each channel: Some peripherals use the trigger instead of a DMA request for pacing the transfer. Other peripherals can have their transfer started by a trigger, then paced by the DMA request. Simple transfers, such as memory-to-memory, can be triggered by (for instance) a timer match or other event. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines, among others.
- Channels can be chained together, allowing one channel to be triggered by a completion of an operation on another channel. For instance, a memory-to-memory transfer could be triggered by completion of a USART buffer being filled.

- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with eight entries (each entry is a pair of transfer addresses).
- Supports single transfers up to 1,024 words. A single transfer is defined as one read from the source address, followed by one write to the destination address.
- Address increment options allow packing and/or unpacking data.

16.3 Signals

The DMA controller is not associated with any device pins. However, some DMA triggers can be taken from device pins (see [Hardware triggers](#)).

16.4 Functional description

The following sections describe functional details of the DMA controller.

16.4.1 Modes of operation

The DMA controller does not have separate operating modes.

Using any specific DMA channel requires initializing the device registers associated with that channel, and setting up the Channel Descriptor. The Channel Descriptor is located in system memory, typically in on-chip SRAM (see [SRAM address of the channel configuration table \(SRAMBASE\)](#)).

The memory locations in a Channel Descriptor are actively used by the DMA controller when it operates on the related channel. So it must be re-initialized if the channel is used again. This is not true for additional reload descriptors that are linked to from the Channel Descriptor.

Table 62. Memory locations in a Channel Descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

Table 63. Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (the size of a descriptor).

16.4.1.1 DMA in low power modes

DMA in sleep mode:

- In sleep mode, the DMA can operate and access all enabled SRAM blocks without waking up the CPU.

DMA in deep-sleep mode:

- Some peripherals support DMA during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm functions that include FIFO support (USART, SPI, and I2S), and the DMIC.
- These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See SYSCCTL for more information.

16.4.2 DMA Transfers

DMA transfer is initiated by writing [Configuration register for DMA channel \(CFG0 - CFG36\)](#) and [Transfer configuration register for DMA channel \(XFERCFG0 - XFERCFG36\)](#) registers and setting ENA bits in [Channel Enable read and set for all DMA channels \(ENABLESET0\)](#). DMA transfer can be started by a hardware or software trigger.

For data transfer from memory to a peripheral or from a peripheral to memory, the address used for SRCINC or DSTINC, whichever corresponds to the fixed peripheral data address, is the address of the peripheral data register. See [Transfer configuration register for DMA channel \(XFERCFG0 - XFERCFG36\)](#). The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer)

Buffer ending address = Buffer starting address + (XFERCOUNT * Transfer increment)

NOTE

XFERCOUNT = Actual count - 1

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the Channel Descriptor will be reloaded with data from memory pointed to by the "Link to next descriptor" entry of the Channel Descriptor. Descriptors loaded in this manner are shown in [Reload Descriptors](#). The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

16.4.2.1 Single buffer

This generally applies to memory to memory transfers, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this operation, only the initial Channel Descriptor as shown in the table below is needed.

Table 64. Channel Descriptor for a single buffer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

For single buffer transfer, [RELOAD](#) bit is 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the Channel Descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

16.4.2.2 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

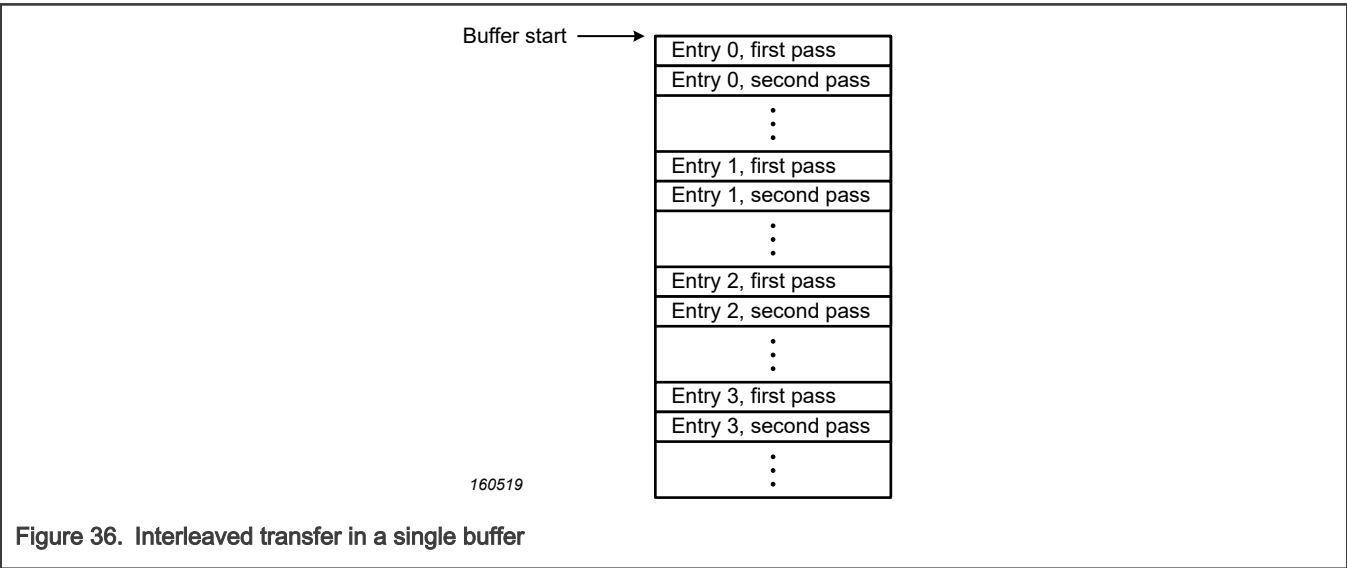


Figure 36. Interleaved transfer in a single buffer

16.4.2.3 Linked Transfers (Linked List)

A linked transfer is transfer across more than one descriptor. A single DMA request or trigger initiates a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 65). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Any descriptor not currently in use can be altered by software.

16.4.2.3.1 Ping-Pong

Ping-pong is a special and more frequently used application of linked transfers. A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. The table below shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 65. Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor		Descriptor B		Descriptor A	
+ 0x0	(not used)	+ 0x0	Buffer B transfer configuration	+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address	+ 0x8	Buffer B memory end address	+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B	+ 0xC	Address of descriptor A	+ 0xC	Address of descriptor B

In addition to the Channel Descriptor, two reload descriptors (A and B) are needed because the Channel Descriptor is actively modified during DMA operation. Descriptor A is typically a copy of the original Channel Descriptor plus the transfer configuration information for buffer A.

The Channel Descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the Channel Descriptor is exhausted, a transfer interrupt informs the CPU that buffer A is available for processing. Descriptor B is then loaded using the link to descriptor B, and a new transfer begins using buffer B.

When Descriptor B is exhausted, a transfer interrupt informs the CPU that buffer B is available for processing. Descriptor A is then loaded using the link to descriptor A. This ping-pong process repeats, alternating between the two memory buffers.

16.4.2.4 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

16.4.2.5 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. For example, this can be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
 - If channel x is configured to auto reload the descriptor on exhausting of the descriptor, **RELOAD** is set, then enable **CLRTRIG** in the channel's transfer configuration in the descriptor.
- For channel y:
 - Configure the DMA_ITRIG_SEL for channel y to use any of the available DMA trigger muxes (see INPUTMUX and **DMA requests**).
 - Configure the chosen DMA trigger mux to select DMA channel x.
 - Enable hardware triggering by setting **HWTRIGEN**.
 - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
 - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

NOTE

After completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above.

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
 - Configure the DMA_ITRIG_SEL (See INPUTMUX) for channel y to use the same DMA trigger mux as chosen for channel y.
 - Enable hardware triggering by setting **HWTRIGEN** in the channel configuration register.
 - Set the trigger type to edge sensitive by clearing **TRIGTYPE** in the channel configuration register.
 - Configure the trigger edge to falling edge by clearing **TRIGPOL** in the channel configuration register.

16.4.3 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request and one trigger input which is multiplexed to many possible input sources.

DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty.

16.4.3.1 DMA with I2C monitor mode

The I²C monitor function may be used with DMA if one of the channels related to the same Flexcomm module is available.

Table 66. DMA with the I²C Monitor function

I ² C Master DMA	I ² C Slave DMA	I ² C Monitor DMA
Not enabled	-	If I ² C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same Flexcomm module.
Enabled	Not enabled	If I ² C Monitor DMA is enabled, it will use the DMA channel for the Slave function of the same Flexcomm module.
Enabled	Enabled	The I ² C Monitor function cannot use DMA.

16.4.4 Hardware triggers

Triggers start the transfer. DMA transfer happens for any particular DMA channel after it is triggered by software or hardware. In typical cases, only a software trigger is used. Other possibilities are occurrence of certain timer or pin related events. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

The DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

Each DMA channel can use one trigger that is independent of the request input for the channel. The trigger input is selected in the DMA_ITRIG_SEL registers (See INPUTMUX). In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See DMA requests and INPUTMUX.

16.4.4.1 Trigger operation detail

A trigger is always needed to start a transfer on a DMA channel. The trigger can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the **SWTRIG** bit equal to 0, the channel can be later triggered by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate **SETTRIG0** bit. Hardware triggering requires setup of the **HWTRIGEN**, **TRIGPOL**, **TRIGTYPE**, and **TRIGBURST** fields in the **Configuration register for DMA channel (CFG0 - CFG36)** for the related channel. When a channel is initially set up, the **SWTRIG** bit can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the **PERIPHREQEN** bit in the related **CFG** register is set. Otherwise, the transfer will proceed at full speed.

The **TRIG** bit can be cleared at the end of a transfer, determined by the value **CLRTRIG**. When **CLRTRIG** = 1, the trigger is cleared when the descriptor is exhausted.

16.4.4.2 Trigger output detail

Each DMA channel provides a trigger output. This allows the possibility of using trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another

peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called "chained operation" or "channel chaining".

16.5 Application information

The two intended scenarios for the use of DMA controllers are:

- One DMA controller (DMA0) is used by the CM33, the other (DMA1) is used by the DSP. This scenario can apply to systems where there is no need to differentiate security between the CPUs or between different tasks.
- One DMA controller (DMA0) is secured and has access to secure spaces and peripherals, the other (DMA1) is not secured and does not have access to secure spaces and peripherals. In this scenario, only the secure code running on the CM33 has access to the secure DMA controller. The non-secure DMA controller will be shared by other code and by the DSP (if needed).

16.6 Memory map and register definition

This section includes the DMA Controller memory map and detailed descriptions of all registers for one DMA controller. DMA transfers are controlled by a set of three registers per channel, the CFG[0:37], CTRLSTAT[0:37], and XFRCFG[0:37] registers.

16.6.1 DMA0 controller register descriptions

16.6.1.1 DMA memory map

DMA0 base address: 4010_4000h

DMA1 base address: 4010_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA control (CTRL)	32	RW	See section
4h	Interrupt status (INTSTAT)	32	R	See section
8h	SRAM address of the channel configuration table (SRAMBASE)	32	RW	See section
20h	Channel Enable read and set for all DMA channels (ENABLESET0)	32	RW	0000_0000h
24h	Channel Enable read and set for all DMA channels (ENABLESET1)	32	RW	0000_0000h
28h	Channel Enable Clear for all DMA channels (ENABLECLR0)	32	RW	See section
2Ch	Channel Enable Clear for all DMA channels (ENABLECLR1)	32	RW	See section
30h	Channel Active status for all DMA channels (ACTIVE0)	32	R	0000_0000h
34h	Channel Active status for all DMA channels (ACTIVE1)	32	R	0000_0000h
38h	Channel Busy status for all DMA channels (BUSY0)	32	R	0000_0000h
3Ch	Channel Busy status for all DMA channels (BUSY1)	32	R	0000_0000h
40h	Error Interrupt status for all DMA channels (ERRINT0)	32	RW	0000_0000h
44h	Error Interrupt status for all DMA channels (ERRINT1)	32	RW	0000_0000h
48h	Interrupt Enable read and Set for all DMA channels (INTENSET0)	32	RW	0000_0000h
4Ch	Interrupt Enable read and Set for all DMA channels (INTENSET1)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
50h	Interrupt Enable Clear for all DMA channels (INTENCLR0)	32	W	See section
54h	Interrupt Enable Clear for all DMA channels (INTENCLR1)	32	RW	See section
58h	Interrupt A status for all DMA channels (INTA0)	32	RW	0000_0000h
5Ch	Interrupt A status for all DMA channels (INTA1)	32	RW	0000_0000h
60h	Interrupt B status for all DMA channels (INTB0)	32	RW	0000_0000h
64h	Interrupt B status for all DMA channels (INTB1)	32	RW	0000_0000h
68h	Set ValidPending control bits for all DMA channels (SETVALID0)	32	W	See section
6Ch	Set ValidPending control bits for all DMA channels (SETVALID1)	32	RW	See section
70h	Set Trigger control bits for all DMA channels (SETTRIG0)	32	W	See section
74h	Set Trigger control bits for all DMA channels (SETTRIG1)	32	RW	See section
78h	Channel Abort control for all DMA channels (ABORT0)	32	W	See section
7Ch	Channel Abort control for all DMA channels (ABORT1)	32	RW	See section
400h - 640h	Configuration register for DMA channel (CFG0 - CFG36)	32	RW	See section
404h - 644h	Control and status register for DMA channel (CTLSTAT0 - CTLSTAT36)	32	R	See section
408h - 648h	Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG36)	32	RW	See section

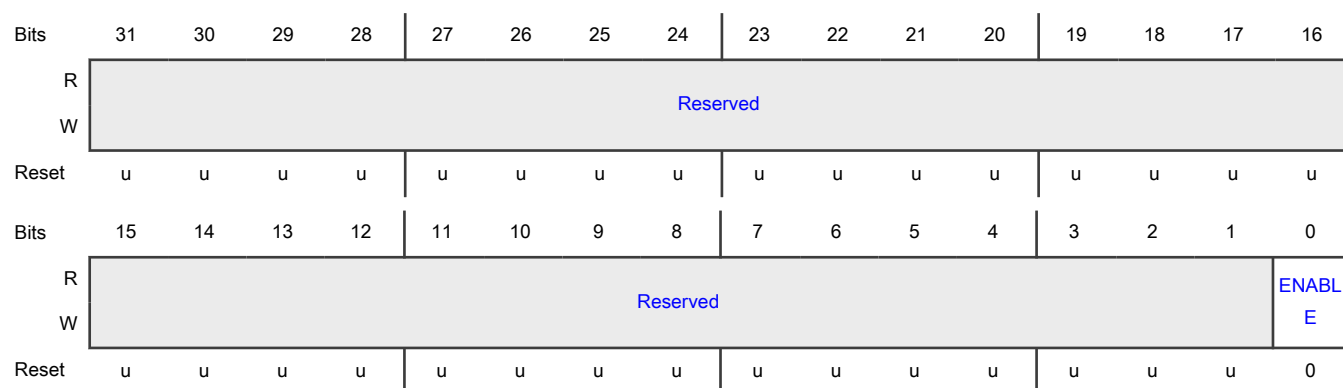
16.6.1.2 DMA control (CTRL)

Offset

Register	Offset
CTRL	0h

Function

This register has the global control bit for enabling the DMA controller.

Diagram**Fields**

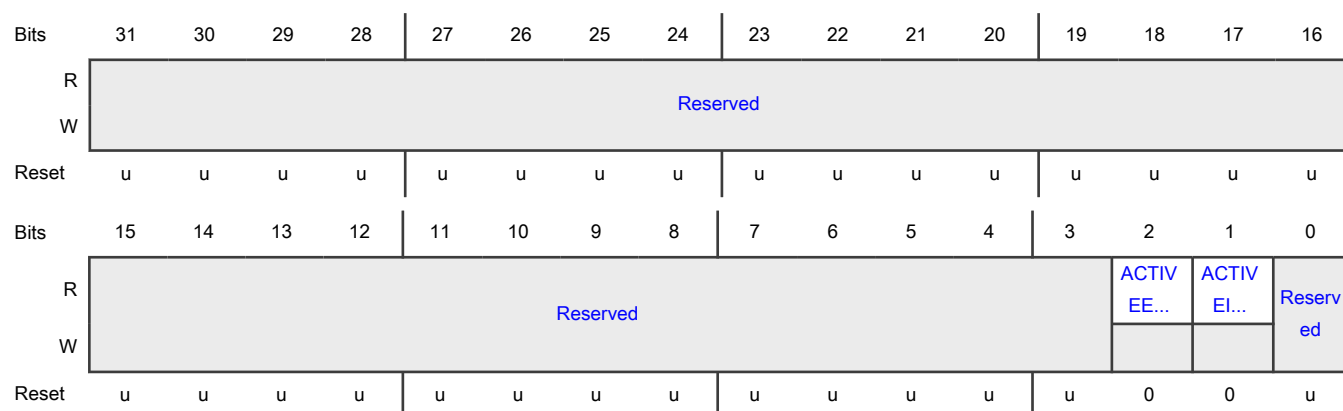
Field	Function
31-1 —	Reserved
0 ENABLE	DMA controller master enable. 0b - DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled. 1b - Enabled. DMA controller is enabled.

16.6.1.3 Interrupt status (INTSTAT)**Offset**

Register	Offset
INTSTAT	4h

Function

It provides an overview of DMA status and allows quick determination of whether any enabled interrupts are pending. For details regarding the channels involved see [Interrupt A status for all DMA channels \(INTA0\)](#) and [Interrupt B status for all DMA channels \(INTB0\)](#)

Diagram**Fields**

Field	Function
31-3 —	Reserved
2 ACTIVEERRINT	Summarizes whether any error interrupts are pending. 0b - No error interrupts are pending. 1b - At least one error interrupt is pending.
1 ACTIVEINT	Summarizes whether any enabled interrupts (other than error interrupts) are pending. 0b - No enabled interrupts are pending. 1b - At least one enabled interrupt is pending.
0 —	Reserved

16.6.1.4 SRAM address of the channel configuration table (SRAMBASE)**Offset**

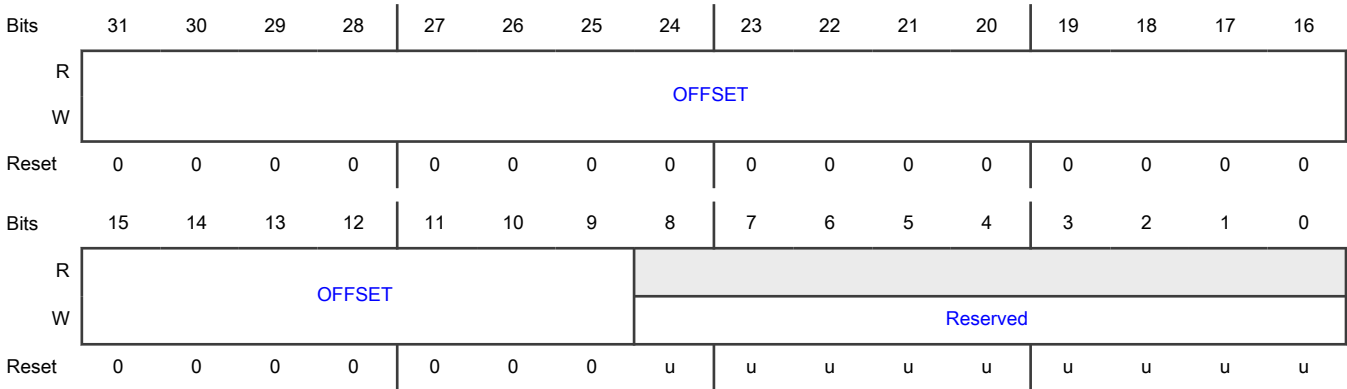
Register	Offset
SRAMBASE	8h

Function

It must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for the DMA channels that will be used in the application.

Each DMA channel has an entry for the Channel Descriptor in the SRAM table. See Chip-specific DMA information section.

Diagram



Fields

Field	Function
31-9 OFFSET	Offset Address bits 31:9 of the beginning of the Channel descriptor table. For 18 channels, the table must begin on a 512 byte boundary. See table above.
8-0 —	Reserved

16.6.1.5 Channel Enable read and set for all DMA channels (ENABLESET0)

Offset

Register	Offset
ENABLESET0	20h

Function

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB
W	LE31	LE30	LE29	LE28	LE27	LE26	LE25	LE24	LE23	LE22	LE21	LE20	LE19	LE18	LE17	LE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB	ENAB
W	LE15	LE14	LE13	LE12	LE11	LE10	LE9	LE8	LE7	LE6	LE5	LE4	LE3	LE2	LE1	LE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 ENABLEn	Enable for DMA channel 0b - DMA channel is disabled. 1b - DMA channel is enabled.

16.6.1.6 Channel Enable read and set for all DMA channels (ENABLESET1)

Offset

Register	Offset
ENABLESET1	24h

Function

The ENABLESET1 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET1 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET1 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR1.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												ENAB	ENAB	ENAB	ENAB
W													LE36	LE35	LE34	LE33
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4-0 ENABLEn	Enable for DMA channel 0b - DMA channel is disabled. 1b - DMA channel is enabled.

16.6.1.7 Channel Enable Clear for all DMA channels (ENABLECLR0)

Offset

Register	Offset
ENABLECLR0	28h

Function

The ENABLECLR0 register is used to clear the enable of one or more DMA channels.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 CLRn	Writing ones to this register clears the corresponding bits in ENABLESET0. 0b - No effect. 1b - DMA channel is cleared.

16.6.1.8 Channel Enable Clear for all DMA channels (ENABLECLR1)

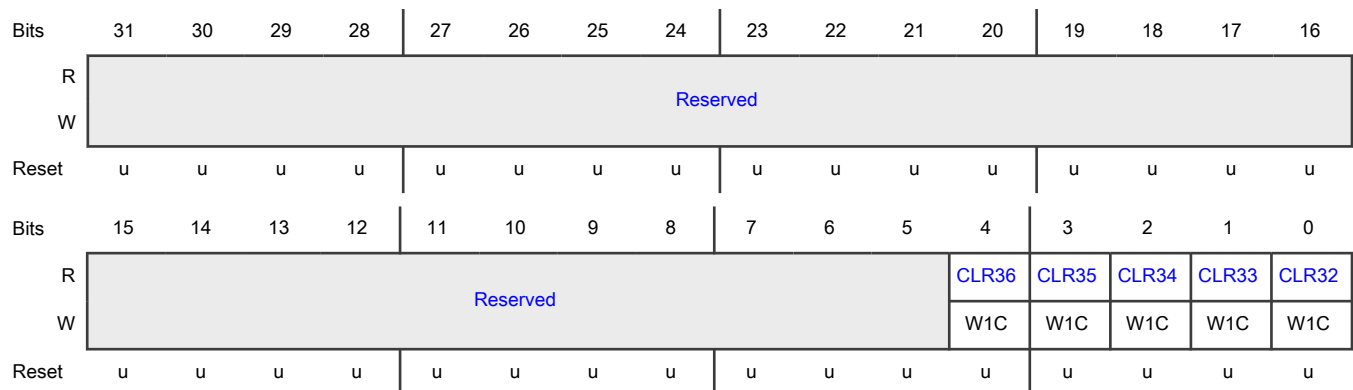
Offset

Register	Offset
ENABLECLR1	2Ch

Function

The ENABLECLR1 register is used to clear the enable of one or more DMA channels.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 CLRn	Writing ones to this register clears the corresponding bits in ENABLESET1. 0b - No effect. 1b - DMA channel is cleared.

16.6.1.9 Channel Active status for all DMA channels (ACTIVE0)

Offset

Register	Offset
ACTIVE0	30h

Function

It indicates which DMA channels are active at the point when the read occurs. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The active status persists another DMA operation from being started, until the pipeline is empty. After end of the last descriptor (when there is no reload), a new DMA operation can be started. An active channel may be aborted by software by setting the appropriate bit in the ABORT register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACTIV E31	ACTIV E30	ACTIV E29	ACTIV E28	ACTIV E27	ACTIV E26	ACTIV E25	ACTIV E24	ACTIV E23	ACTIV E22	ACTIV E21	ACTIV E20	ACTIV E19	ACTIV E18	ACTIV E17	ACTIV E16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ACTIV E15	ACTIV E14	ACTIV E13	ACTIV E12	ACTIV E11	ACTIV E10	ACTIV E9	ACTIV E8	ACTIV E7	ACTIV E6	ACTIV E5	ACTIV E4	ACTIV E3	ACTIV E2	ACTIV E1	ACTIV E0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

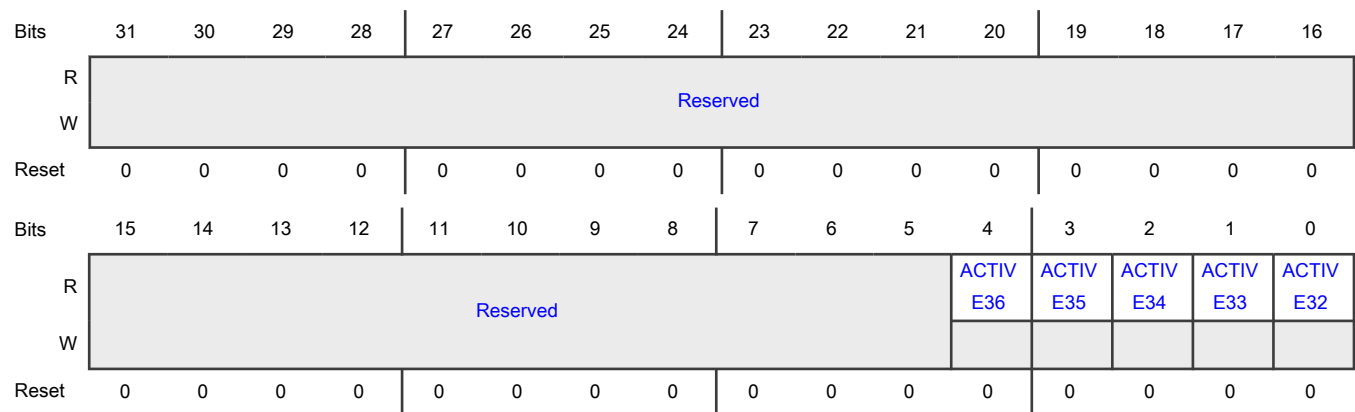
Field	Function
31-0 ACTIVEn	Active flag for DMA channel. 0b - DMA channel is not active. 1b - DMA channel is active.

16.6.1.10 Channel Active status for all DMA channels (ACTIVE1)**Offset**

Register	Offset
ACTIVE1	34h

Function

It indicates which DMA channels are active at the point when the read occurs. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The active status persists another DMA operation from being started, until the pipeline is empty. After end of the last descriptor (when there is no reload), a new DMA operation can be started. An active channel may be aborted by software by setting the appropriate bit in the ABORT register.

Diagram**Fields**

Field	Function
31-5 —	Reserved
4-0 ACTIVE _n	Active flag for DMA channel. 0b - DMA channel is not active. 1b - DMA channel is active.

16.6.1.11 Channel Busy status for all DMA channels (BUSY0)**Offset**

Register	Offset
BUSY0	38h

Function

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Diagram

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

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

Fields

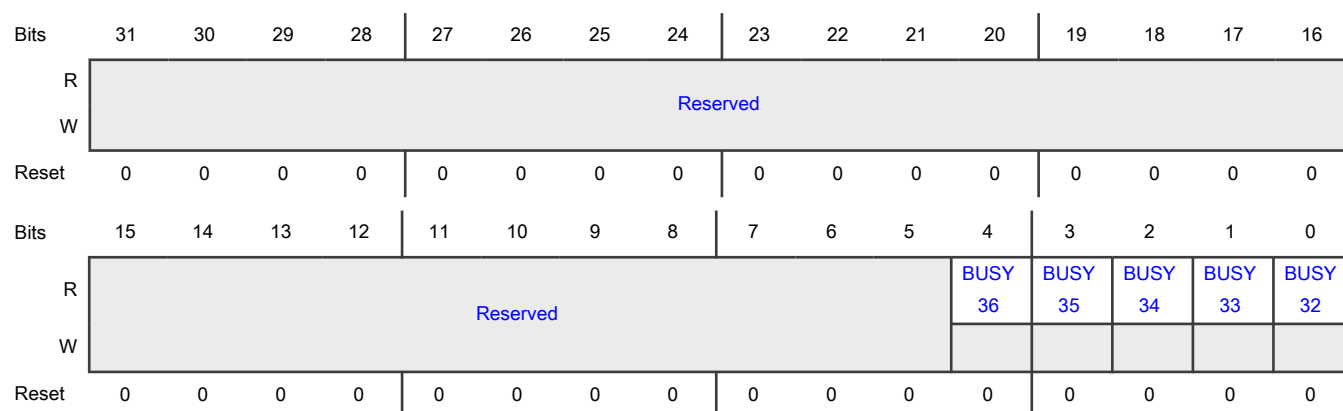
Field	Function
31-0 BUSYn	Busy flag for DMA channel. 0b - DMA channel is not busy. 1b - DMA channel is busy.

16.6.1.12 Channel Busy status for all DMA channels (BUSY1)**Offset**

Register	Offset
BUSY1	3Ch

Function

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Diagram**Fields**

Field	Function
31-5 —	Reserved
4-0 BUSYn	Busy flag for DMA channel. 0b - DMA channel is not busy. 1b - DMA channel is busy.

16.6.1.13 Error Interrupt status for all DMA channels (ERRINT0)**Offset**

Register	Offset
ERRINT0	40h

Function

It contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output. Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERR3	ERR3	ERR2	ERR2	ERR2	ERR2	ERR2	ERR2	ERR2	ERR2	ERR2	ERR2	ERR1	ERR1	ERR1	ERR1
W	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR1	ERR1	ERR1	ERR1	ERR1	ERR1	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	5	4	3	2	1	0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 ERRn	Error Interrupt flag for DMA channel. 0b - The Error Interrupt is not active for DMA channel. 1b - The Error Interrupt is pending for DMA channel.

16.6.1.14 Error Interrupt status for all DMA channels (ERRINT1)

Offset

Register	Offset
ERRINT1	44h

Function

It contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output. Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												ERR36	ERR35	ERR34	ERR33
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4-0 ERRn	Error Interrupt flag for DMA channel. 0b - The Error Interrupt is not active for DMA channel. 1b - The Error Interrupt is pending for DMA channel.

16.6.1.15 Interrupt Enable read and Set for all DMA channels (INTENSET0)

Offset

Register	Offset
INTENSET0	48h

Function

It controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output. Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Diagram

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

Fields

Field	Function
31-0 INTENn	Interrupt Enable read and set for DMA channel. 0b - The Interrupt for DMA channel is disabled. 1b - The Interrupt for DMA channel is enabled.

16.6.1.16 Interrupt Enable read and Set for all DMA channels (INTENSET1)

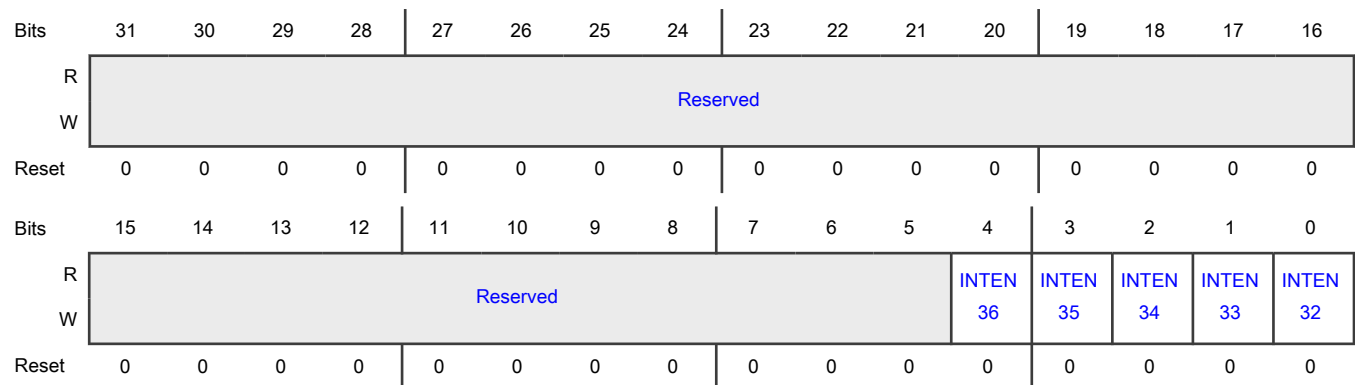
Offset

Register	Offset
INTENSET1	4Ch

Function

It controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output. Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 INTEN _n	Interrupt Enable read and set for DMA channel. 0b - The Interrupt for DMA channel is disabled. 1b - The Interrupt for DMA channel is enabled.

16.6.1.17 Interrupt Enable Clear for all DMA channels (INTENCLR0)

Offset

Register	Offset
INTENCLR0	50h

Function

This register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 CLRn	Writing ones to this register clears corresponding bits in the DMAIntEnSet0.

16.6.1.18 Interrupt Enable Clear for all DMA channels (INTENCLR1)**Offset**

Register	Offset
INTENCLR1	54h

Function

This register is used to clear interrupt enable bits in INTENSET1. The register is write-only.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W													CLR36	CLR35	CLR34	CLR33
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-5 —	Reserved
4-0 CLRn	Writing ones to this register clears corresponding bits in the DMAIntEnSet1.

16.6.1.19 Interrupt A status for all DMA channels (INTA0)

Offset

Register	Offset
INTA0	58h

Function

It contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTA3	INTA3	INTA2	INTA2	INTA2	INTA2	INTA2	INTA2	INTA2	INTA2	INTA2	INTA2	INTA1	INTA1	INTA1	INTA1
W	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTA1	INTA1	INTA1	INTA1	INTA1	INTA1	INTA9	INTA8	INTA7	INTA6	INTA5	INTA4	INTA3	INTA2	INTA1	INTA0
W	5	4	3	2	1	0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 INTAn	Interrupt A status for DMA channel. 0b - The DMA channel interrupt A is not active. 1b - The DMA channel interrupt A is active.

16.6.1.20 Interrupt A status for all DMA channels (INTA1)

Offset

Register	Offset
INTA1	5Ch

Function

It contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												INTA3	INTA3	INTA3	INTA3
W													6	5	4	3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4-0 INTAn	Interrupt A status for DMA channel. 0b - The DMA channel interrupt A is not active. 1b - The DMA channel interrupt A is active.

16.6.1.21 Interrupt B status for all DMA channels (INTB0)

Offset

Register	Offset
INTB0	60h

Function

It contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTB3	INTB3	INTB2	INTB2	INTB2	INTB2	INTB2	INTB2	INTB2	INTB2	INTB2	INTB2	INTB1	INTB1	INTB1	INTB1
W	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTB1	INTB1	INTB1	INTB1	INTB1	INTB1	INTB9	INTB8	INTB7	INTB6	INTB5	INTB4	INTB3	INTB2	INTB1	INTB0
W	5	4	3	2	1	0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

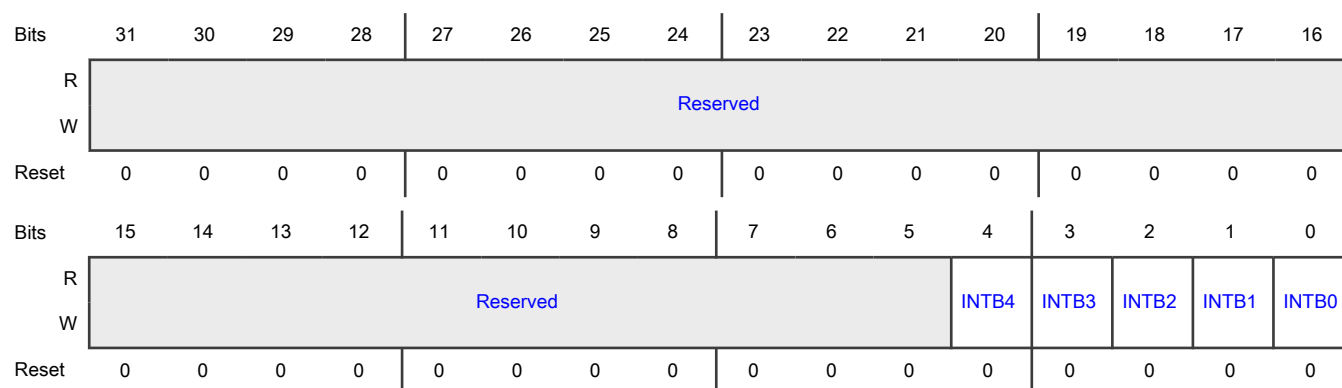
Field	Function
31-0 INTBn	Interrupt B status for DMA channel. 0b - The DMA channel interrupt B is not active. 1b - The DMA channel interrupt B is active.

16.6.1.22 Interrupt B status for all DMA channels (INTB1)**Offset**

Register	Offset
INTB1	64h

Function

It contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Diagram**Fields**

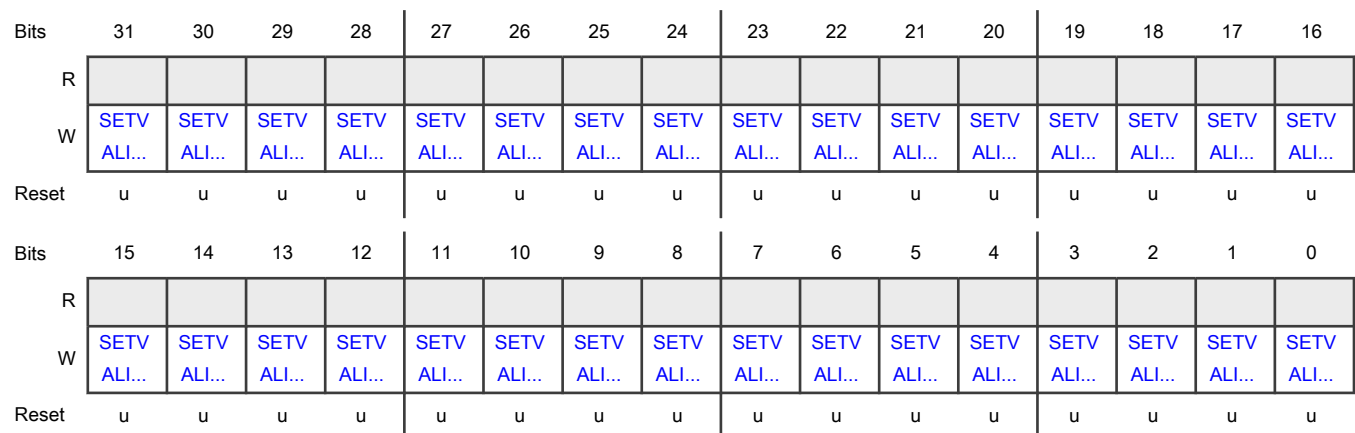
Field	Function
31-5 —	Reserved
4-0 INTBn	Interrupt B status for DMA channel. 0b - The DMA channel interrupt B is not active. 1b - The DMA channel interrupt B is active.

16.6.1.23 Set ValidPending control bits for all DMA channels (SETVALID0)**Offset**

Register	Offset
SETVALID0	68h

Function

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Diagram**Fields**

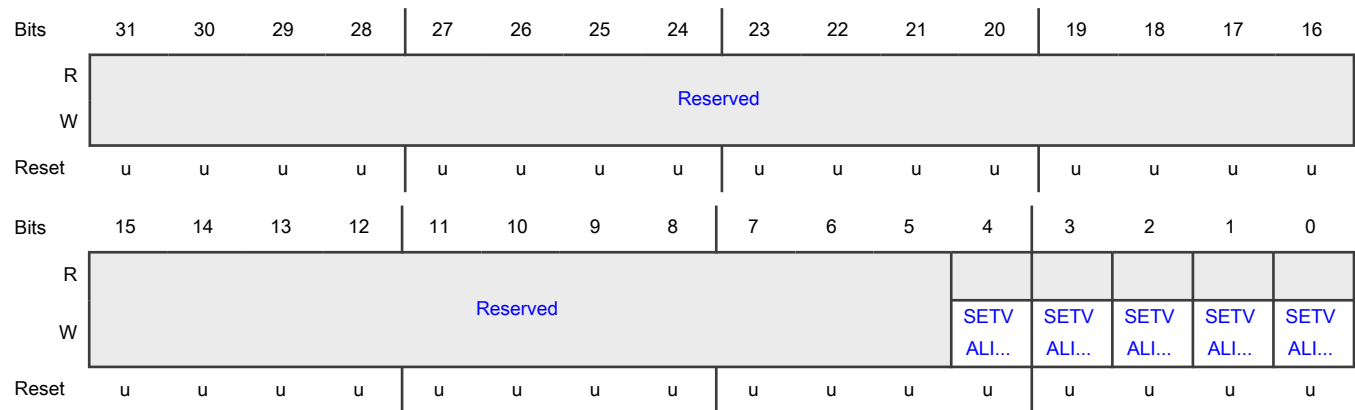
Field	Function
31-0 SETVALIDn	SetValid control for DMA channel. 0b - No effect. 1b - Sets the ValidPending control bit for DMA channel.

16.6.1.24 Set ValidPending control bits for all DMA channels (SETVALID1)**Offset**

Register	Offset
SETVALID1	6Ch

Function

The SETVALID1 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID1 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID1 bit is set.

Diagram**Fields**

Field	Function
31-5 —	Reserved
4-0 SETVALIDn	SetValid control for DMA channel. 0b - No effect. 1b - Sets the ValidPending control bit for DMA channel.

16.6.1.25 Set Trigger control bits for all DMA channels (SETTRIG0)**Offset**

Register	Offset
SETTRIG0	70h

Function

This register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG9	SETT RIG8	SETT RIG7	SETT RIG6	SETT RIG5	SETT RIG4	SETT RIG3	SETT RIG2	SETT RIG1	SETT RIG0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 SETTRIGn	Set Trigger control bit for DMA channel. 0b - No effect. 1b - Sets the Trig bit for DMA channel.

16.6.1.26 Set Trigger control bits for all DMA channels (SETTRIG1)

Offset

Register	Offset
SETTRIG1	74h

Function

register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W													SETT RIG...	SETT RIG...	SETT RIG...	SETT RIG...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-5 —	Reserved
4-0 SETTRIGn	Set Trigger control bit for DMA channel. 0b - No effect. 1b - Sets the Trig bit for DMA channel.

16.6.1.27 Channel Abort control for all DMA channels (ABORT0)

Offset

Register	Offset
ABORT0	78h

Function

The Abort 0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	ABOR T31	ABOR T30	ABOR T29	ABOR T28	ABOR T27	ABOR T26	ABOR T25	ABOR T24	ABOR T23	ABOR T22	ABOR T21	ABOR T20	ABOR T19	ABOR T18	ABOR T17	ABOR T16
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	ABOR T15	ABOR T14	ABOR T13	ABOR T12	ABOR T11	ABOR T10	ABOR T9	ABOR T8	ABOR T7	ABOR T6	ABOR T5	ABOR T4	ABOR T3	ABOR T2	ABOR T1	ABOR T0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 ABORTn	Abort control for DMA channel. 0b - No effect. 1b - Aborts DMA operations on channel.

16.6.1.28 Channel Abort control for all DMA channels (ABORT1)

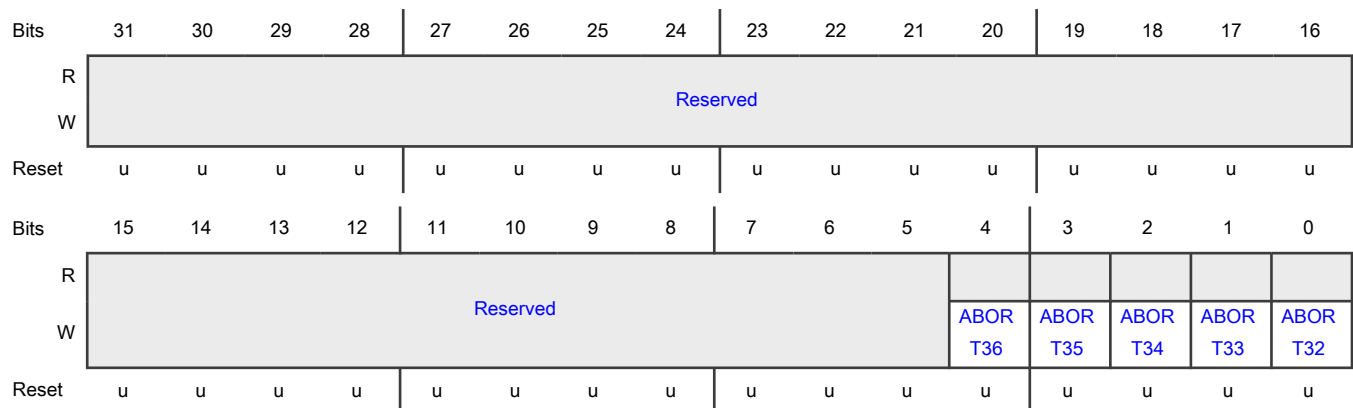
Offset

Register	Offset
ABORT1	7Ch

Function

The Abort 1 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 ABORT _n	Abort control for DMA channel. 0b - No effect. 1b - Aborts DMA operations on channel.

16.6.1.29 Configuration register for DMA channel (CFG0 - CFG36)

Offset

For a = 0 to 36:

Register	Offset
CFGa	400h + (a × 10h)

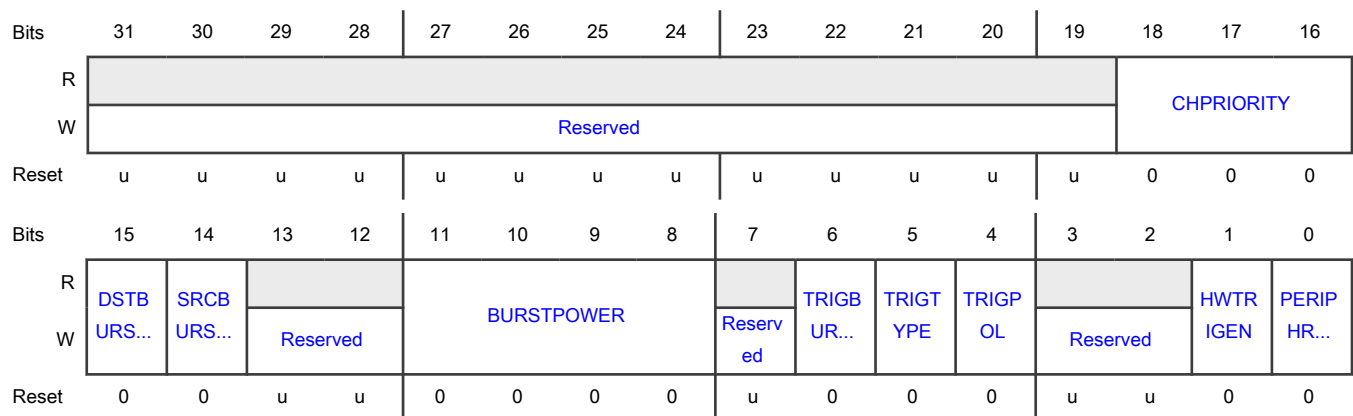
Function

The CFG register contains various configuration options for DMA channels.

Table 67. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

Diagram



Fields

Field	Function
31-19 —	Reserved Read value is undefined, only zero should be written.
18-16 CHPRIORITY	Priority of channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.
15 DSTBURSTWR AP	Destination Burst Wrap. When enabled, the destination data address for the DMA is 'wrapped', meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst. 0b - Disabled. Destination burst wrapping is not enabled for this DMA channel. 1b - Enabled. Destination burst wrapping is enabled for this DMA channel.
14 SRCBURSTWR AP	Source Burst Wrap. When enabled, the source data address for the DMA is 'wrapped', meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst. 0b - Disabled. Source burst wrapping is not enabled DMA channel. 1b - Enabled. Source burst wrapping is enabled DMA channel.
13-12 —	Reserved Read value is undefined, only zero should be written.
11-8 BURSTPOWER	Burst Power. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected. When the TRIGBURST field = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). This corresponds to a single transfer. 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. Others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an integer multiple of the burst size.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE Total number of bytes transferred is: (XFERCOUNT + 1) x data width (Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG36)).
7 —	Reserved Read value is undefined, only zero should be written.
6 TRIGBURST	Trigger Burst. Selects whether hardware triggers cause a single or burst transfer. 0b - Single transfer. Hardware trigger causes a single transfer. 1b - Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.
5 TRIGTYPE	Trigger Type. Selects hardware trigger as edge triggered or level triggered. 0b - Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger. 1b - Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.
4 TRIGPOL	Trigger Polarity. Selects the polarity of a hardware trigger for this channel. 0b - Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE. 1b - Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.
3-2 —	Read value is undefined, only zero should be written.
1 HWTRIGEN	Hardware Triggering Enable for channel. 0b - Hardware triggering not used for channel. 1b - Hardware triggering used for channel.
0 PERIPHREQE N	Peripheral request Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If a DMA channel is used to perform a memory-to-memory transfer, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller. 0b - Peripheral DMA requests disabled. 1b - Peripheral DMA requests enabled.

16.6.1.30 Control and status register for DMA channel (CTLSTAT0 - CTLSTAT36)

Offset

For a = 0 to 36:

Register	Offset
CTLSTATa	404h + (a × 10h)

Function

It provides status flags specific to the DMA channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												TRIG	Reserved	VALID PE...	
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	0	u	0

Fields

Field	Function
31-3 —	Reserved Read value is undefined, only zero should be written.
2 TRIG	Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out. 1b - Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.
1 —	Reserved Read value is undefined, only zero should be written.
0 VALIDPENDING	Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel. 0b - No effect on DMA operation. 1b - Valid pending.

16.6.1.31 Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG36)

Offset

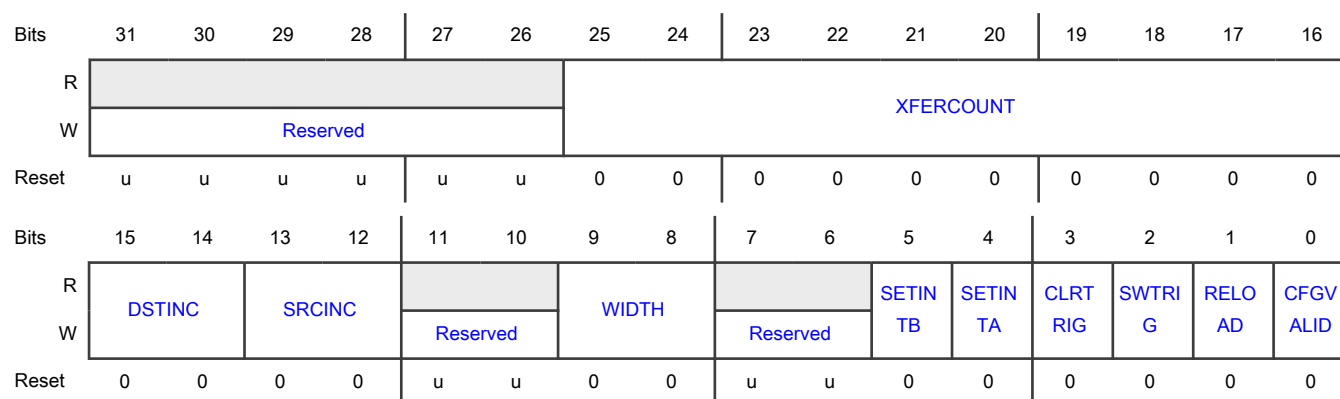
For a = 0 to 36:

Register	Offset
XFERCFGa	408h + (a × 10h)

Function

The registers contains transfer related configuration information for the DMA channel. Using the Reload bit, the register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	Reserved Read value is undefined, only zero should be written.
25-16 XFERCOUNT	Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: $(XFERCOUNT + 1) \times \text{data width}$ (as defined by the WIDTH field). The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = a total of 1 transfer will be performed. 0x1 = a total of 2 transfers will be performed. 0x3FF = a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination address increment Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.
13-12 SRCINC	Source address increment Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	Reserved Read value is undefined, only zero should be written.
9-8 WIDTH	Transfer width used for this DMA channel. 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved.
7-6	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 SETINTB	Set Interrupt flag B for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt flag A for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.
3 CLRTRIG	Clear Trigger. 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger. This bit is always read as 0. 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Reload. Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. The channels' control structure should not be reloaded when the current descriptor is exhausted. 1b - Enabled. The channels' control structure should be reloaded when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Valid. The current channel descriptor is considered valid.

Chapter 17

SmartDMA

17.1 Introduction

The SmartDMA is a new micro-engine featured on the RT500 microcontroller that operates in parallel with the ARM Cortex M33 core. The SmartDMA performs Event/IO Driven handling and accesses shared data without the core's intervention.

17.1.1 Features

17.1.1.1 SmartDMA Summary

The SmartDMA is targeted for graphics enablement support in the form of a graphics library.

17.1.1.2 SmartDMA Graphics Library

The SmartDMA Graphics Library consists of the following functions to further enhance our graphics handling:

```

kSMARTDMA_FlexIO_DMA_Endian_Swap,                               /*!< Send data to FlexIO with
Endian Swap. */                                                  /*!< Send data to FlexIO
kSMARTDMA_FlexIO_DMA_Reverse32,                                  /*!< Send data to FlexIO
with reverse32 order. */                                         /*!< Send data to
kSMARTDMA_FlexIO_DMA,                                           /*!< Send data to
FlexIO. */                                                       /*!< Send data to FlexIO
kSMARTDMA_FlexIO_DMA_Reverse,                                   /*!< Send data to FlexIO
with reverse order. */                                           /*!< Convert RGB565
kSMARTDMA_RGB565To888,                                          /*!< Convert RGB565
to RGB888 and save to output memory,
use

parametersmartdma_rgb565_rgb888_param_t. */
kSMARTDMA_FlexIO_DMA_RGB565To888,                               /*!< Convert RGB565 to RGB888 and
send to FlexIO, use parameter smartdma_flexio_mculcd_param_t. */
kSMARTDMA_FlexIO_DMA_ARGB2RGB,                                  /*!< Convert ARGB to RGB and
send to FlexIO, use parameter smartdma_flexio_mculcd_param_t. */
kSMARTDMA_FlexIO_DMA_ARGB2RGB_Endian_Swap,                     /*!< Convert ARGB to RGB, then swap
endian, and send to FlexIO, use
parameter
smartdma_flexio_mculcd_param_t.
*/
kSMARTDMA_FlexIO_DMA_ARGB2RGB_Endian_Swap_Reverse, /*!< Convert ARGB to RGB, then swap endian
and reverse, and send to FlexIO, use
parameter
smartdma_flexio_mculcd_param_t. */
kSMARTDMA_MIPI_RGB565_DMA,                                       /*!< Send RGB565 data to
MIPI DSI, use parameter smartdma_dsi_param_t. */
kSMARTDMA_MIPI_RGB888_DMA,                                       /*!< Send RGB888 data to
MIPI DSI, use parameter smartdma_dsi_param_t. */

```

17.2 External Signal Description

There are 22 external signals associated with the SmartDMA: 22 input/output signals.

Chapter 18

Non-Secure Boot ROM

18.1 Overview

This section provides an introduction to the System Non-Secure Boot ROM.

NOTE

The term kb_api is same as iap_api.

18.1.1 Features

192KB on-chip boot ROM with boot ROM that allows various boot options and APIs.

- Support serial interface booting (UART, SPI, I2C, USB-HID) from an application processor, automated booting from a Serial NOR (QSPI FLASH, Octal FLASH, Hyper FLASH), eMMC device, based on the state of ISP pins or OTP setting (PRIMARY_BOOT_SRC).
- Support Execute in place (XIP) from a Serial NOR Flash (in single, dual, quad or octal mode) with/without OTFAD.
- Provide OTP API for OTP memory programming.
- Provide External FLASH API to access the Serial FLASH device.
- Provide KB API for image update in the field.

18.2 Functional Description

The internal ROM memory is used to store the boot code. After a reset, the ARM processor starts its code execution from this memory. The boot ROM code is executed every time the part is powered-on or is reset.

Since the RT500 has no internal FLASH for code and data storage, images must be stored elsewhere for download upon reset or the CPU can execute from an external memory(XIP). Images can be loaded into on-chip SRAM from external FLASH or downloaded via the serial ports (UART, SPI, I2C, USB-HID). The code is then validated, and boot ROM will vector to on-chip SRAM.

Depending on the values of the OTP bits and ISP pins, and the image header type definition, the boot ROM decides whether to download code into the on-chip SRAM or run from external memory (XIP). The boot ROM checks the OTP bit settings first and then the ISP pins. The OTP is a one-time-program fuse word. Before programming the OTP fuse, user can use the OTP related shadow register first before the final programming. More details please refer to OTP chapter about the OTP shadow register. If bit [3:0] in OTP word BOOT_CFG [0] is programmed (4b'0000), the boot source is determined by the states of the ISP boot pins (PIO1_15, PIO3_28, and PIO3_29).

Table 68. Primary Boot Source based on PRIMARY_BOOT_SRC bits[3:0] in BOOT_CFG [0]

Boot Mode	Field	Primary boot Source (Master boot source)
ISP_PIN_BOOT	b'0000	ISP pins will determine boot source
FLEXSPI_BOOT	b'0001	Boot from Octal/Quad SPI flash device. If boot fails falls into ISP mode.
SDHC0_BOOT	b'0010	Boot from eMMC device connected to SDHC0 port. If boot fails falls into ISP mode.

Table continues on the next page...

Table 68. Primary Boot Source based on PRIMARY_BOOT_SRC bits[3:0] in BOOT_CFG [0] (continued)

Boot Mode	Field	Primary boot Source (Master boot source)
SDHC1_BOOT	b'0011	Boot from eMMC device connected to SDHC1 port. If boot fails falls into ISP mode.
SPI_SLV_BOOT	b'0100	Use the SPI slave interface to download the boot image into SRAM and boot.
	b'0101	Reserved
UART_BOOT	b'0110	Use the UART slave interface to download the boot image into SRAM and boot.
SPI_FLASH_BOOT	b'0111	Boot from 1 bit NOR flash via SPI interface. The SPI instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.
USB_HID_ISP	b'1000	Use the USB-HID interface to download the boot image into SRAM and boot. USB ISP mode is not supported in the WLCSP.
ISP_FUSE_BOOT	b'1001	ISP boot, BOOT_CFG[0]: bit 4-6 determine the boot device, 0: Enable all 1:HID 2:UART 3:SPI 4:I2C 7:Disable all
	b'1010	Reserved
	b'1011	Reserved
FLEXSPI_REC_BOOT	b'1100	Boot from Quad/Octal SPI flash device. If image is not found check recovery boot using SPI-flash device through FlexComm.
SDHC0_REC_BOOT	b'1101	Boot from SDHC0 port device. If image is not found check recovery boot using SPI-flash device through FlexComm.
SDHC1_REC_BOOT	b'1110	Boot from SDHC1 port device. If image is not found check recovery boot using SPI-flash device through FlexComm.
	b'1111	Reserved

If PRIMARY_BOOT_SRC bits are not set, the RT500 will read the status of the ISP pins to determine boot source.

18.2.1 Boot and ISP Downloader Modes

Table 69. Boot mode and ISP Downloader modes based on ISP pins

Boot mode	ISP2 pin PIO3_29	ISP1 pin PIO3_28	ISP0 pin PIO1_15	Description
-	low	low	low	Reserved
	high	low	high	
	low	low	high	Reserved
USB HID ISP	low	high	low	Boot to ISP mode, using USB-HID class
FLEXSPI Boot	low	high	high	Boot from Quad/Octal Flash devices connected to the FLEXSPI interface. The RT500 will look for a valid image in external Quad/Octal Flash device. If there is no valid image found, the RT500 will enter recovery boot or ISP boot mode based on DEFAULT_ISP_MODE bits.
SDIO 0(eMMC)	high	low	low	Boot from an eMMC device connected to SDIO 0 interface. The RT500 will look for a valid image in the eMMC device. If there is no valid image found, the RT500 will enter the ISP boot mode based on the value of OTP DEFAULT_ISP_MODE bits (6:4, BOOT_CFG [0]).
Serial ISP (UART, I2C SPI)	high	high	low	The Serial Interface (UART, I2C, SPI) is used to program OTP, external FLASH, eMMC device
Serial download (UART, I2C, SPI, HID)	high	high	high	Serial Master boot (SPI Slave, or UART, I2C, HID) is used to download a boot image over the serial interface (SPI Slave, or UART, I2C, HID)

18.2.2 ISP peripheral pin mux assignments

The table below shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

Table 70. ISP peripheral pin mux assignments

Boot interface	Pins	
FLEXSPI	PIO1_18	FLEXSPI0_SCLK
	PIO1_19	FLEXSPI0_SS0_b
	PIO1_20	FLEXSPI0_DATA0
	PIO1_21	FLEXSPI0_DATA1
	PIO1_22	FLEXSPI0_DATA2
	PIO1_23	FLEXSPI0_DATA3
	PIO1_24	FLEXSPI0_DATA4
	PIO1_25	FLEXSPI0_DATA5
	PIO1_26	FLEXSPI0_DATA6
	PIO1_27	FLEXSPI0_DATA7
	PIO1_28	FLEXSPI0_DQS
	PIO1_29	FLEXSPI0_SS1_b
USB		usb1_vbus
		usb1_vdd1v8
		usb1_vdd3v3
		usb1_dm
		usb1_dp
		usb1_id
SPI SLAVE	PIO1_11	HS_SPI0_SCK (FC14)
	PIO1_12	HS_SPI0_MISO (FC14)
	PIO1_13	HS_SPI0_MOSI (FC14)
	PIO1_14	HS_SPI0_SSELN0 (FC14)
	PIO1_15	HS_SPI0_SSELN1 (FC14)
I2C SLAVE	PIO0_15	FC2_TXD_SCL_MISO
	PIO0_16	FC2_RXD_SDA_MOSI

Table 70. ISP peripheral pin mux assignments (continued)

Boot interface	Pins	
UART		
	PIO0_1	FC0_TXD_SCL_MISO_WS
	PIO0_2	FC0_RXD_SDA_MOSI_DATA
SDIO0		
	PIO1_30	SD0_CLK
	PIO1_31	SD0_CMD
	PIO2_0	SD0_D[0]
	PIO2_1	SD0_D[1]
	PIO2_2	SD0_D[2]
	PIO2_3	SD0_D[3]
	PIO2_4	SD0_WR_PRT
	PIO2_5	SD0_D[4]
	PIO2_6	SD0_D[5]
	PIO2_7	SD0_D[6]
	PIO2_8	SD0_D[7]
	PIO2_9	SD0_CARD_DET_N
	PIO2_10	SD0_RESET_N
	PIO2_11	SD0_VOLT
SDIO1		
	PIO3_8	SD1_CLK
	PIO3_9	SD1_CMD
	PIO3_10	SD1_D[0]
	PIO3_11	SD1_D[1]
	PIO3_12	SD1_D[2]
	PIO3_13	SD1_D[3]
	PIO3_14	SD1_WR_PRT
	PIO3_15	SD1_D[4]
	PIO3_16	SD1_D[5]
	PIO3_17	SD1_D[6]
	PIO3_18	SD1_D[7]
	PIO3_19	SD1_CARD_DET_N

Table 70. ISP peripheral pin mux assignments (continued)

Boot interface	Pins	
	PIO3_20	SD1_RESET_N
	PIO3_21	SD1_VOLT
SPI flash	Flexcomm port to use for recovery SPI flash boot. FC0 Use FlexCom0 pins PIO0_0 (SCK), PIO0_1 (MISO), PIO0_2 (MOSI), PIO0_3 (SEL) FC1 Use FlexCom1 pins PIO0_7 (SCK), P0_8 (MISO), PIO0_9 (MOSI), PIO0_10 (SEL) NOTE The SPI Flash Recovery Boot pin functions are multiplexed with the JTAG boundary scan functions. To ensure boundary scan mode is not inadvertently entered, the TRST pin should be externally tied low using a weak pull-down resistor (100 kohm) to ensure proper SPI Flash Recovery Boot operation. FC2 Use FlexCom2 pins PIO0_14 (SCK), PIO0_15 (MISO), PIO0_16 (MOSI), PIO0_17 (SEL) FC3 Use FlexCom3 pins PIO0_21 (SCK), PIO0_22 (MISO), PIO0_23 (MOSI), PIO0_24 (SEL) FC4 Use FlexCom4 pins PIO0_28 (SCK), PIO0_29 (MISO), PIO0_30 (MOSI), PIO0_31 (SEL) FC5 Use FlexCom5 pins PIO1_3 (SCK), PIO1_4 (MISO), PIO1_5 (MOSI), PIO1_6 (SEL) FC6 Use FlexCom6 pins PIO3_25 (SCK), PIO3_26 (MISO), PIO3_27 (MOSI), PIO3_28 (SEL) FC7 Use FlexCom7 pins PIO4_0 (SCK), PIO4_1 (MISO), PIO4_2 (MOSI), PIO4_3 (SEL)	
ISP	PIO3_29 - bit2 PIO3_28 - bit1 PIO1_15 - bit0	

18.2.3 Top-Level Boot Process

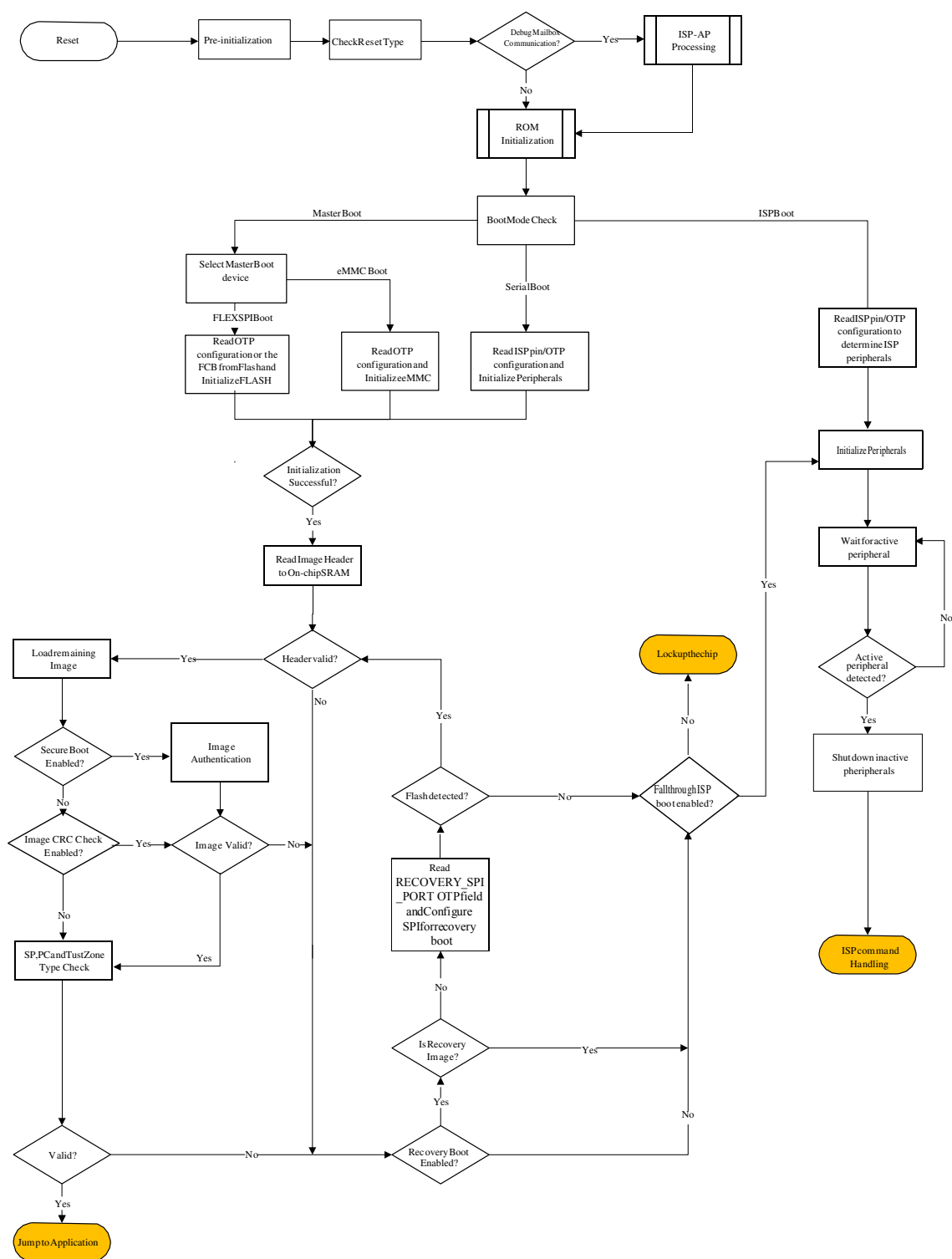


Figure 37. Top-Level Boot Process

The boot process starts after Reset is released. As part of Pre-initialization, OTP INTEGRITY CHECK is done when ENABLE_CRC_CHECK is enabled. The CPU clock is determined by the OTP field BOOT_CLK_SPEED, which is 48MHz

from the FRO48M clock source by default. The boot ROM determines the boot mode based on the PRIMARY_BOOT_SRC or the state of the ISP pins during reset if PRIMARY_BOOT_SRC = 4b'0000. When the Cortex-M33 starts the boot ROM, the SWD access is disabled until a boot image starts execution or the SWD access is enabled using mailbox to enter debug mode.

Once the boot mode is determined, and the boot image is available on the selected external Flash device (eMMC or Serial NOR FLASH), the ROM boot ROM starts to copy the first 64 bytes of image header from the external Flash device into on-chip SRAM and then validate the vector table and image header. If the boot image is downloaded from a serial interface (via UART, SPI, I2C USB HID), the entire image including the image header is already loaded into SRAM then.

The boot ROM checks the following for image integrity:

- Validate image using header and image markers.
- Validate image using CRC32 (if the CRC check feature is enabled in the image header).
- For non-XIP, imageLoadAddress points to internal RAM, outside of RAM region reserved for Boot ROM (see [SRAM used by the ISP](#))
- For non-XIP, verifies the imageLoadAddress + imageLength fits with the RAM address space.

The boot ROM begins by examining the image type located at offset 0x24 of the boot image. If the value is a valid image type, then validation of an image header will begin. Qualification of the image header continues by reading the Image Load Address from offset 0x34 and image length from offset 0x20. The boot ROM verifies the imageLoadAddress points to internal RAM, and check if it is outside of SRAM range reserved for the boot ROM, see section SRAM used by the ISP. The boot ROM also verifies the imageLoadAddress + imageLength fits within the RAM address space.

After the completion of the validation of the image header, the qualification continues by examining the image type field. If a bootable (non XIP) image resides in the external flash, the entire image will be loaded into the on-chip SRAM first, then the imageLength field in the image header will be used as the length to perform a CRC check if the CRC check feature is enabled.

CRC check will be performed on the image (either in on-chip SRAM or on an external XIP image) if the CRC feature is enabled. The CRC calculation starts at offset 0x0 from the beginning of the image and continues up to the number of bytes specified by the imageLength field with the crcChecksum field excluded. CRC is calculated over the entire image, but CRC field. Since the crcChecksum field is skipped during the calculation, the CRC calculation is on imageLength – 4. The image validity is determined by comparing the result to the crcChecksum field in the image header after completion of the CRC32 computation, the image itself is analyzed valid if a match exists, otherwise is judged invalid.

If the boot ROM decides not to boot the application, the user can use the blhost tool "get-property 8 1" command to get the last error.

Table 71. imageType Field

Bits	Description
[31 - 16]	Reserved
[15]	Key Store Included • 0 - The boot image does not have key store. • 1 - The boot image has key store. NOTE setting this flag has effect only to Load to Ram nonXip image types
[14]	TZ-M Image Type • 0 - TZ-M enabled image. ARMv8-M security extensions are included.

Table continues on the next page...

Table 71. imageType Field (continued)

Bits	Description
	1 - TZ-M disabled image. ARMv8-M security extensions are not included. See the note .
[13]	TZ-M Preset 0 - No TZ-M peripherals preset. 1 - TZ-M peripherals preset. The TZ-M related peripherals are configured by ROM boot ROM based on data stored in extended header
[12 - 8]	Reserved
[7 - 0]	Image Type 0x0: plain image 0x1: plain signed image (Load to Ram - nonXip) 0x2: plain image with CRC (Load to Ram - nonXip) 0x3: encrypted signed (Load to Ram - nonXip) 0x4: Xip plain signed 0x5: Xip plain with CRC

NOTE

NXP offers some M33 MCUs without TrustZone (ARMv8-M security extensions enabled). TrustZone disabled image is an image, which is supposed to be executed on devices without TrustZone (M33 without security extension). To keep full software compatibility between CM33 with and without security extension, this software/image must be executed in normal mode. To allow easy transition between devices with and without security extension, the ROM supports direct execution of software developed for CM33 devices without security extension. If the TrustZone disabled image is executed, the ROM, before it jumps to user application, configures all device resources into normal world, lock access to all TrustZone related configuration registers, switches from secure to normal world and finally jumps to user application. This mode allows easy reuse of the software developed for Cortex-M33 without security extension. The user doesn't need to do any software modification.

Table 72. Image Header Format (shared for all imageTypes)

Offset	Field	Description
0x00-0x1F	Reserved	
0x20	imageLength	The image length
0x24	imageType	See imageType Field
0x28	authBlockOffset/crcChecksum	Authenticate Block Offset or CRC32 checksum
0x2C-0x33	Reserved	
0x34	imageLoadAddress	Image Load Address
0x38-0x3F	Reserved	

Table 73. Plain Image Layout

Offset	Field	Description
0x00	Stack Pointer	Normal CM33 Core Exception Handlers
0x04	Reset Handler	
0x08	NMI_Handler	
0x0c	HardFault_Handler	
0x10	MemMange_Handler	
0x14	BusFault_Handler	
0x18	UsageFault_Handler	
0x1C	SecureFault_Handler	
0x20	imageLength	The image length
0x24	imageType	See imageType Field
0x28	authBlockOffset	Authenticate Block Offset or CRC32 checksum
0x2C	SVC Handler	Normal CM33 Core Exception Handlers
0x30	DebugMon_Handler	
0x34	imageLoadAddress	Image Load Address
0x38	PendSV_Handler	Normal CM33 Core Exception Handlers
0x3C	SysTick_Handler	

Table 74. XIP Image layout (FlexSPI)

Offset	Width (Bytes)	Field	Description
0x0000_0000	256	KeyBlob for OTFAD	Optional, programmed with all 0x00s if OTFAD is not enabled.
0x0000_0400	512	Flash Config Block	The FlexSPI FLASH configuration block. This block is required if the FLEXSPI_AUTO_PROBE_EN is not blown on the OTP.
0x0000_0600	4	Boot image version	When using dual image ping-pong boot, this field is used to store the boot image version. For more details please refer to FLEXSPI boot.
0x0000_0800	2048	KeyStore	Fixed KeyStore field. This field is required if the KeyStore feature is enabled.
0x0000_1000	Image size	Bootable Image	The boot image, starts with valid image header.

Table continues on the next page...

Table 74. XIP Image layout (FlexSPI) (continued)

Offset	Width (Bytes)	Field	Description
			See Image Header Format (shared for all imageTypes).

18.3 Basic boot ROM function on RT500

RT500 does not have an internal flash inside, so before the chip boots, the boot image needs to exist in an external non-volatile boot media. Boot ROM support is needed to access the external non-volatile storage when fetching the boot image. RT500 supports three different boot devices:

- External non-volatile memory such as Serial NOR FLASH
- eMMC
- 1-bit SPI NOR flash

Boot ROM fixes the pin connection between the external memory device for FlexSPI module(Serial NOR flash)/uSDHC module(eMMC memory) and the chip. The pin connection for the boot interfaces needs to be correct for board design.

The ROM selects the boot device via ISP pins, PIO3_29(ISP2), PIO3_28(ISP1), PIO1_15(ISP0) or OTP fuse word 0x60 configuration. Boot ROM supports erase/program/read external non-volatile memory via ROM ISP command in ISP mode(In-System Programming mode) to place the boot image into external non-volatile memory. For more details, please refer [External Memory Support](#).

FlexSPI serial NOR flash boot can be Load-to-RAM boot or XIP boot. For Load-to-RAM, after boot ROM runs, it initializes the FlexSPI module according to the external NOR flash type connected to the chip. ROM loads the boot image from the NOR flash device at offset 0x1000 to the internal SRAM. After that, the ROM jumps to SRAM to run the boot image. For XIP boot, the boot ROM only boot the image at the NOR flash device. The boot image header inside the boot image tells the ROM whether the boot image is Load-to-RAM image or XIP image. FlexSPI boot also supports dual image boot. If FlexSPI is selected as the boot device, it defaults support to choose which image to boot based on the image version, which we call it as Ping-Pong boot. During FlexSPI boot, ROM always tries to boot the image with the latest version, if the latest boot image boot fails, ROM tries to boot the image with old image version, if both image boots fail, ROM enters ISP mode. For more details about the Ping-Pong boot, please refer [FLEXSPI Dual Image Ping-Pong boot flow](#).

eMMC boot does not support XIP access, so boot ROM only supports the Load-to-RAM boot. ROM initializes the uSDHC interface after power-up. ROM will copy the boot image from device offset at 0x1000 to internal SRAM, and then ROM jumps to SRAM to run the boot image.

18.4 Boot modes

The boot modes supported by the ROM boot ROM include:

- Master boot mode
- Recovery boot mode
- Serial boot mode
- ISP boot mode

18.4.1 Master boot mode

The following boot devices are supported under Master boot mode:

- FlexSPI boot
- eMMC boot
- SPI 1-bit NOR Boot

The boot ROM looks for the boot image from a specified offset on a boot media. See the details in below table.

Table 75. Image offset on different boot media

Boot media	Image Offset
FlexSPI Boot (Serial NOR Flash device)	0x1000
eMMC boot (eMMC memory)	0x1000
SPI 1-bit NOR Boot	0x1000

18.4.1.1 FLEXSPI boot

The boot ROM supports access to different serial NOR Flash devices from various vendors via the FLEXSPI interface using 1-bit, 2-bit(dual), 4-bit(quad) or 8-bit(octal) mode by employing the Flash configuration block(FCB) located at offset 0x400 on the Flash device or the Flash Autoprobe feature specified in OTP.

- If FLEXSPI_FLASH_AUTO_PROBE_EN is blown, the boot ROM will perform Flash auto-probe sequence using parameters blown in FLEXSPI_FLASH_PROBE_TYPE field which defines the Flash Auto-probe type, FLEXSPI_FLASH_TYPE field which defines the FLASH type and FLEXSPI_FLASH_BOOT_FREQ which defines the Flash access speed.
- If not blown, the boot ROM will look at offset 0x400 on the Flash device, if data at offset 0x400 equal to 0x42464346, the boot ROM will read the whole 512-byte FCB into on-chip SRAM and configure the FLEXSPI controller using this FCB accordingly.

Table 76. FLASH CONFIG TABLE

Field	Offset	Size (bytes)	Description
tag	0x000	4	Config Block tag, must be 0x42464346
version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	
readSampleClkSrc	0x00C	1	Read Sampling Clock source option 0 - Internal loopback 1 - Loopback from DQS pad 2 - Loopback from SCK pad 3 - External DQS signal
csHoldTime	0x00D	1	CS Hold time in terms of Flash clock cycles Recommended value is 3
csSetupTime	0x00E	1	CS setup time in terms of Flash clock cycles Recommended value is 3
columnAddressWidth	0x00F	1	Column Address Width Set to 3 for HyperFLASH

Table continues on the next page...

Table 76. FLASH CONFIG TABLE (continued)

Field	Offset	Size (bytes)	Description
			Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
configCmdEnable	0x01C	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01D	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command types Support up to 3 sequences
reserved	0x02C	4	
configCmdArgs	0x030	12	Configure Command Argument Support up to 3 arguments
reserved	0x03C	4	
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options

Table continues on the next page...

Table 76. FLASH CONFIG TABLE (continued)

Field	Offset	Size (bytes)	Description
			bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type 1- Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads 4 - Quad pads 8 - Octal pads
serialClkFreq	0x046	1	Flash Frequency 1 - 30MHz 2 - 50MHz 3 - 60MHz 4 - 80MHz 5 - 100MHz 6 - 120MHz 7 - 133MHz 8 - 166MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH

Table continues on the next page...

Table 76. FLASH CONFIG TABLE (continued)

Field	Offset	Size (bytes)	Description
reserved	0x048	8	Reserved for future use
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
SflashB1Size	0x058	4	Size of Flash connected to B1
sflashB2Size	0x05C	4	Size of Flash connected to B2
csPadSettingOverride	0x060	4	Pad override value for CS pin Using this value to configure CS pin if it is non-zero, otherwise use default ROM setting
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06C	4	Dqs pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time for PORT A and PORT B
busyOffset	0x07C	2	Busy offset, valid value: 0-31
busyBitPolarity	0x07E	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0 when flash device is busy
lookupTable	0x080	256	16 LUT sequences each sequence consists of 4 words, see FLEXSPI chapter for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1B0	16	Reserved for future use
pageSize	0x1C0	4	Page size of Flash
sectorSize	0x1C4	4	Sector size of Flash
ipcmdSerialClkFreq	0x1C8	1	Serial Clock Frequency for IP command

Table continues on the next page...

Table 76. FLASH CONFIG TABLE (continued)

Field	Offset	Size (bytes)	Description
			0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1C9	1	If the sector size is the same with block size. 0 - No 1 - Yes
isDataOrderSwapped	0x1CA	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1CB	1	Reserved for future use
serialNorType	0x1CC	1	Serial NOR Flash type: 0/1/2/3 0- Serial Nor Type StandardSPI 1- Serial Nor Type HyperBus 2- Serial Nor Type XPI 3- Serial Nor Type NoCmd
needExitNoCmdMode	0x1CD	1	Need to exit No Cmd mode before other IP command
halfClkForNonReadCmd	0x1CE	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1CF	1	Need to Restore No Cmd mode after IP command execution
blockSize	0x1D0	4	Flash Block size
flashStateCtx	0x1D4	4	Flash State Context, which is used to restore the flash to default state
reserved	0x1D8	40	Reserved for future use

18.4.1.1.1 Serial NOR FLASH reset

During FLEXSPI boot the boot process requires the FLASH device to be in a certain mode, for example, 1-bit SPI compatible mode. The FLASH device will naturally be in this mode after a POR reset because the power-up sequence will reset it with the RT500 device together. However, the FLASH device will not be in 1-bit SPI compatible mode if the FLASH device is configured to DPI mode or QPI mode or Octal mode when any non-POR resets happen, for example, watchdog timer and external pin reset. In such case, special processing is required by the boot process to restore the FLASH device to 1-bit SPI compatible mode before continuing access to the Flash device. In general, this can be achieved by using a GPIO to assert a reset pin on the FLASH device.

The boot ROM can perform the reset process and reset the FLASH device to 1-bit SPI compatible mode if the FLEXSPI_RESET_PIN_EN is blown, using the GPIO specified by the combination of FLEXSPI_RESET_PIN_PORT and FLEXSPI_RESET_PIN_GPIO.

FLEXSPI_PWR_HOLD_TIME, FLEXSPI_HOLD_TIME, FLEXSPI_RESET_PIN are OTP bits set in BOOT_CFG[1]. FLEXSPI_RESET_PIN is active low. FLEXSPI_PWR_HOLD_TIME starts after MCU reset is deasserted. During FLEXSPI_PWR_HOLD_TIME, FLEXSPI_RESET_PIN is not driven, and left in High-Z state. After time FLEXSPI_PWR_HOLD_TIME, it drives FLEXSPI_RESET_PIN low for 25us. Then drives high for another 25us. Then it waits for time FLEXSPI_HOLD_TIME before FlexSPI chip select pin is asserted. This whole sequence is then repeated 4 times if needed before falling through to next states.

18.4.1.1.2 FlexSPI boot configurations in OTP

The FLEXSPI boot related configurations are allocated in BOOT_CFG1 OTP word, see the details of the assignment in below table.

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_AUTO_PROBE_EN		Quad/Octal-SPI flash auto probe feature enable.	0	1	
FLEXSPI_PROBE_TYPE		Quad/Octal-SPI flash probe type.	1	3	
	QSPI_NOR	QuadSPI NOR			b'000
	MACRONIX_OCTAL	Macronix Octal FLASH			b'001
	MICRON_OCTAL	Micron Octal FLASH			b'010
	ADESTO_OCTAL	Adesto Octal FLASH			b'011
		Reserved			b'100
		Reserved			b'101
		Reserved			b'110
		Reserved			b'111
FLEXSPI_FLASH_TYPE		Define typical Serial NOR Flash types	4	3	
	QSPI_ADDR_3B	Device supports 3B read by default			b'000
		Reserved			b'001
	HYPER_1V8	HyperFLASH 1V8			b'010
	HYPER_3V3	HyperFLASH 3V3			b'011
	OSPI_DDR_MXIC	MXIC Octal DDR			b'100
	OSPI_DDR_MICRON	Micron Octal DDR			b'101
		Reserved			b'110
		Reserved			b'111

Table continues on the next page...

Table continued from the previous page...

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_DUMMY_CYCLES		Quad/Octal-SPI dummy cycles for read command.	7	4	
	PROBE_DUMMY_CYCLE	The dummy cycles are probed automatically			b'0000
		Number of dummy cycle is specified by this field			Other value
FLEXSPI_FREQUENCY		Q/O-SPI flash interface frequency.	11	3	
	FLEXSPI_100MHZ	100MHz			b'000
	FLEXSPI_120MHZ	120MHz			b'001
	FLEXSPI_133MHZ	133MHz			b'010
	FLEXSPI_166MHZ	166MHz			b'011
	Reserved	Reserved			b'100
	FLEXSPI_80MHZ	80MHz			b'101
	FLEXSPI_60MHZ	60MHz			b'110
	FLEXSPI_50MHZ	50MHz			b'111
FLEXSPI_RESET_PIN_ENABLE		Use FLEXSPI_RESET_PIN to reset the flash device.	14	1	
	NO_RESET	O/QSPI device reset pin is not connected or available.			
	EN_RESET	O/QSPI device reset pin is connected to a GPIO (QSPI_RESET_PIN).			
FLEXSPI_RESET_PIN		GPIO port and pin number to use for O/QSPI reset function.	15	8	
	GPIO_PORT	Defines GPIO port.	0	3	

Table continues on the next page...

Table continued from the previous page...

Field Name	Enum Name	Description	Offset	Width	Value
	GPIO_PIN_NUM	Defines GPIO pin number.	3	5	
QSPI_HOLD TIME		Wait time before access to Serial Flash.	23	2	
	WAIT_500_US	Wait for 500 micro seconds.			b'00
	WAIT_1_MS	Wait for 1 milli seconds.			b'01
	WAIT_3_MS	Wait for 3 milli seconds.			b'10
	WAIT_10_MS	Wait for 10 milli seconds.			b'11
FLEXSPI_PWR_HOLD_TIME		Delay after POR before accessing Quad/Octal-SPI flash devices in addition to delay defined by QSPI_HOLD TIME field.	25	4	
	NO_DELAY	No delay			b'0000
	100US_DELAY	Waits additional 100 micro-seconds.			b'0001
	500US_DELAY	Waits additional 500 micro-seconds.			b'0010
	1MS_DELAY	Waits additional 1 milli-seconds.			b'0011
	10MS_DELAY	Waits additional 10 milli-seconds.			b'0100
	20MS_DELAY	Waits additional 20 milli-seconds.			b'0101
	40MS_DELAY	Waits additional 40 milli-seconds.			b'0110
	60MS_DELAY	Waits additional 60 milli-seconds.			b'0111
	80MS_DELAY	Waits additional 80 milli-seconds.			b'1000

Table continues on the next page...

Table continued from the previous page...

Field Name	Enum Name	Description	Offset	Width	Value
	100MS_DELAY	Waits additional 100 milli-seconds.			b'1001
	120MS_DELAY	Waits additional 120 milli-seconds.			b'1010
	140MS_DELAY	Waits additional 140 milli-seconds.			b'1011
	160MS_DELAY	Waits additional 160 milli-seconds.			b'1100
	180MS_DELAY	Waits additional 180 milli-seconds.			b'1101
	200MS_DELAY	Waits additional 200 milli-seconds.			b'1110
	220MS_DELAY	Waits additional 220 milli-seconds.			b'1111
RESERVED			29	3	

18.4.1.1.3 FlexSPI Boot flow

Below Figure illustrates the brief FLEXSPI Boot flow. It shows the details of normal image load and authentication flow. For more details of Keyblob, please refer Secure boot chapter.

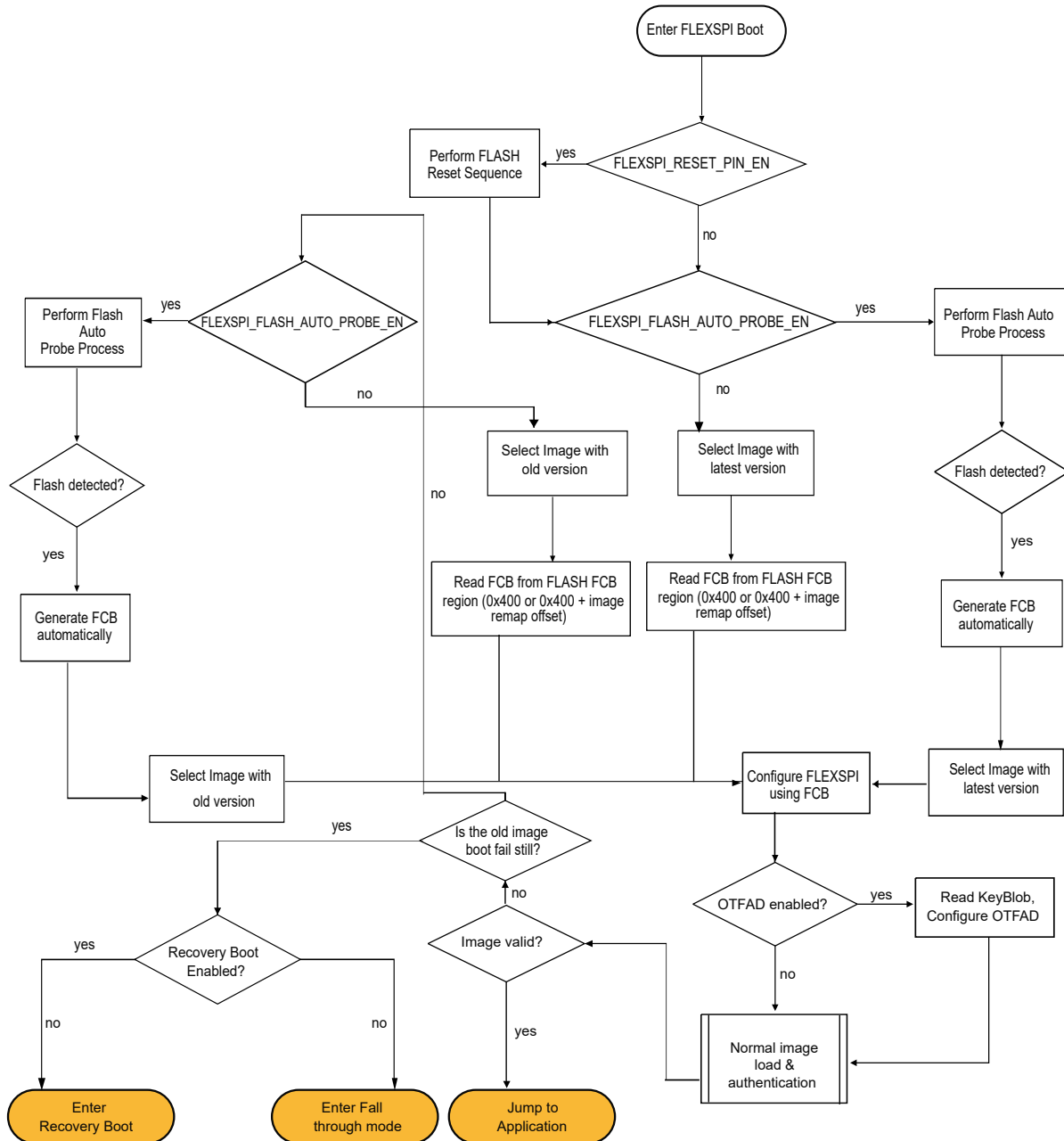


Figure 38. FLEXSPI FLASH Boot Flow

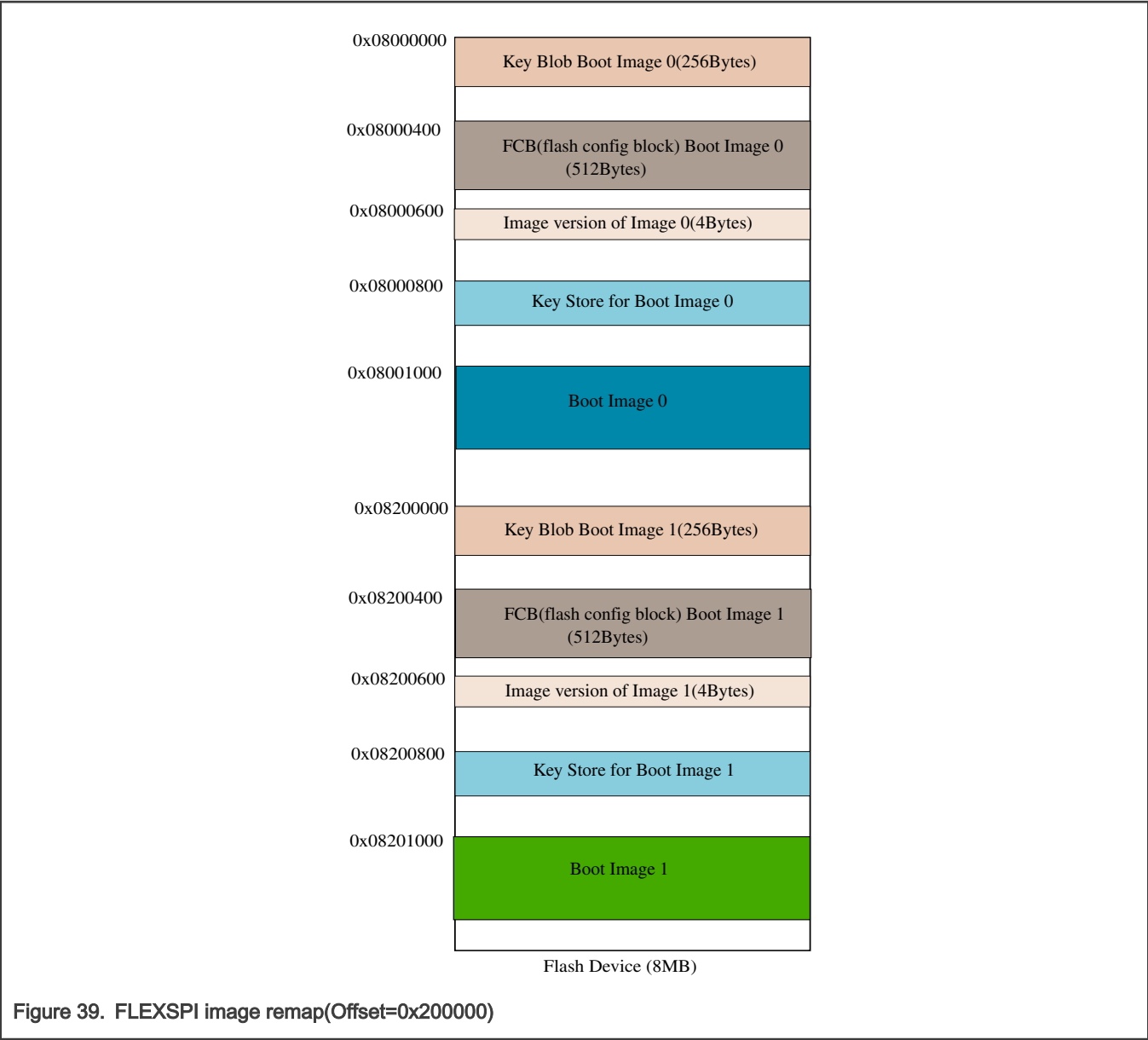
18.4.1.1.3.1 FLEXSPI Dual Image Ping-Pong Boot

FLEXSPI controller supports the remap function which can remap the AHB accessing address of the serial NOR flash to another address with the offset, in this way FLEXSPI boot supports second image boot when the first image boots fail.

18.4.1.1.3.1.1 Basic function of the FLEXSPI remap

When the remap offset is set, FLEXSPI AHB memory access will change the access address adding the offset as shown in the figure below. When the offset is set to 2MB(0x200000), the access to 0x0800000 via FLEXSPI will be remap to 0x08200000. In this way we can implement dual image boot with 2 images. Offset and the size of image 1 is set via OTP fuse by the user.

The offset and the size of remap region is set use OTP fuse by user.



18.4.1.1.3.1.2 FLEXSPI image remap offset and Boot Image 1 size configuration in OTP

The FLEXSPI dual image boot related configurations are allocated in BOOT_CFG2 and BOOT_CFG3 OTP word.

Table 77. Boot image 1 offset(BOOT_CFG3: OTP fuse word 0x63)

Field Name	Offset	Width	Value	Description
Second image offset	22	10	x	x * 256KB

Table 78. Boot Image 1 size (BOOT_CFG2)

Field Name	Offset	Width	Value	Description
First image size	28	4	0	Same size of second image offset

Table continues on the next page...

Table 78. Boot Image 1 size (BOOT_CFG2) (continued)

Field Name	Offset	Width	Value	Description
			13	256KB
			14	512KB
			15	768KB
			x	x*1MB

NOTE

FlexSPI remap size cannot exceed the start address of Boot Image 1.

18.4.1.1.3.1.3 FLEXSPI Dual Image Ping-Pong boot flow

For dual image ping-pong boot, each of the boot images has its image version which is used for ROM to first select the latest boot image, if the latest image boot fails, the old image will be used to boot again.

For FLEXSPI boot, ROM always tries to find the latest boot image and boot, if boot fails, ROM will try to find the old boot image, if the old image still boots fail, then ROM will enter fall ISP boot mode. Below is the boot flow chart for Ping-Pong boot.

Each boot image version is put at the offset 0x600 of the FLEXSPI boot device as the figure above shows. Image version is used 4 bytes to define, the lower 2 bytes are the real image version number, and the upper 2 bytes are the invert value of lower 2 bytes image version, all other values which do not follow this rule will be regarded as an invalid value, if the boot image version is not valid, ROM will not boot that image at all. But if the image version is all 0xFFFFFFFF (for the condition that the user does not program the image version) it will also be considered as a valid value. When one image version is 0xFFFFFFFF and the other one is a valid image version, ROM always first boot the image with the valid value not the one with image version as 0xFFFFFFFF. If both image version is 0xFFFFFFFF, ROM always boots the first image resides at lower address in the FLEXSPI NOR flash device.

Below are some examples of the boot image version.

0xFFFE0001, valid boot image version, image version is 1.

0xFFFD0002, valid boot image version, image version is 2.

0x00000003, invalid boot image version, does not follow the image version rules.

0xFFFFFFFF, valid image version, but with lower boot sequence.

0x0000FFFF, valid image version, image version if the largest version 0xFFFF.

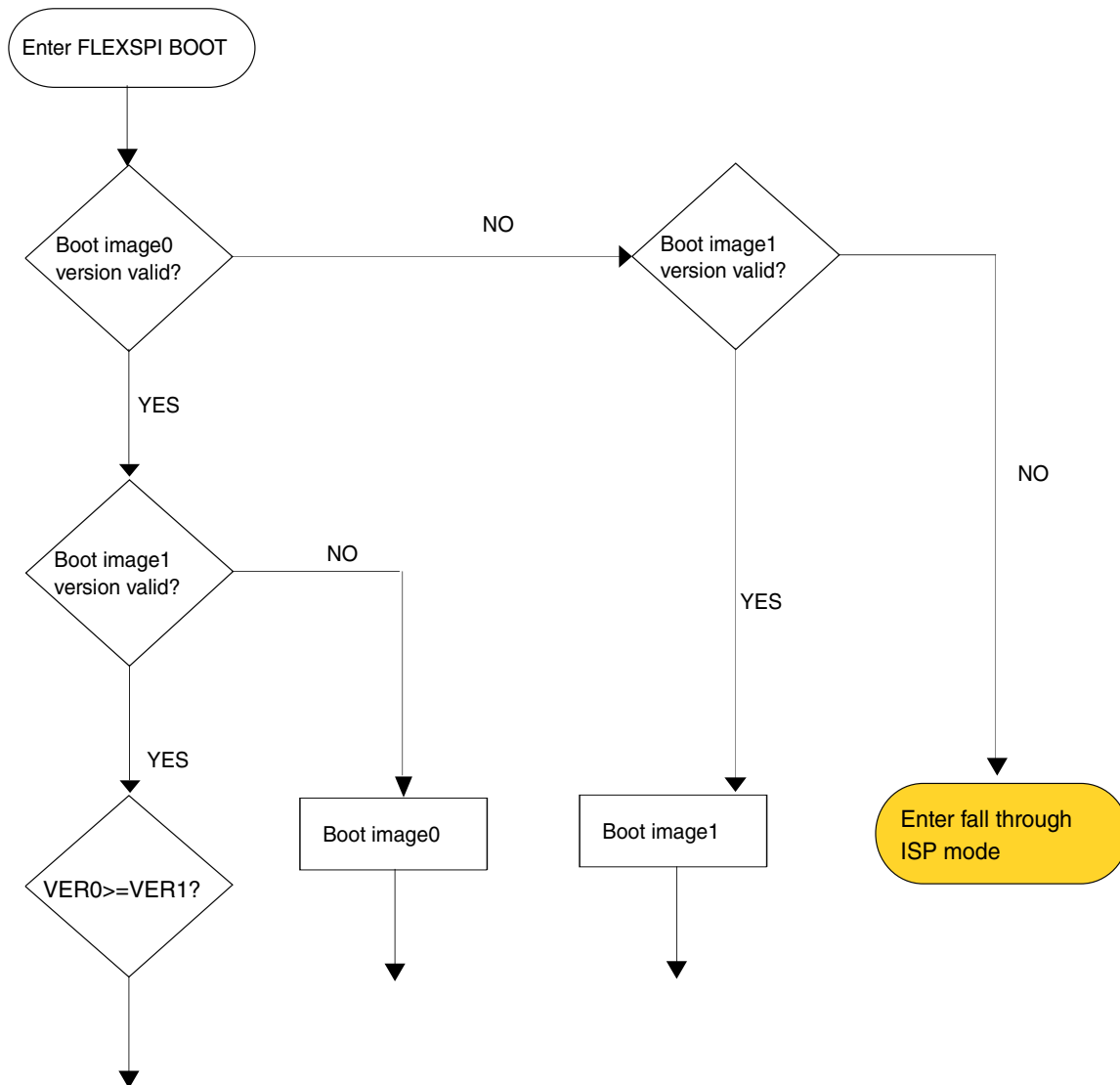


Figure 40. FLEXSPI Boot Image Selection

For FLEXSPI boot, ROM always tries to find the latest boot image and boot, if boot fails, ROM will try to find the old boot image, if the old image still boots fail, then ROM will enter fall ISP boot mode. Below is the boot flow chart for Ping-Pong boot.

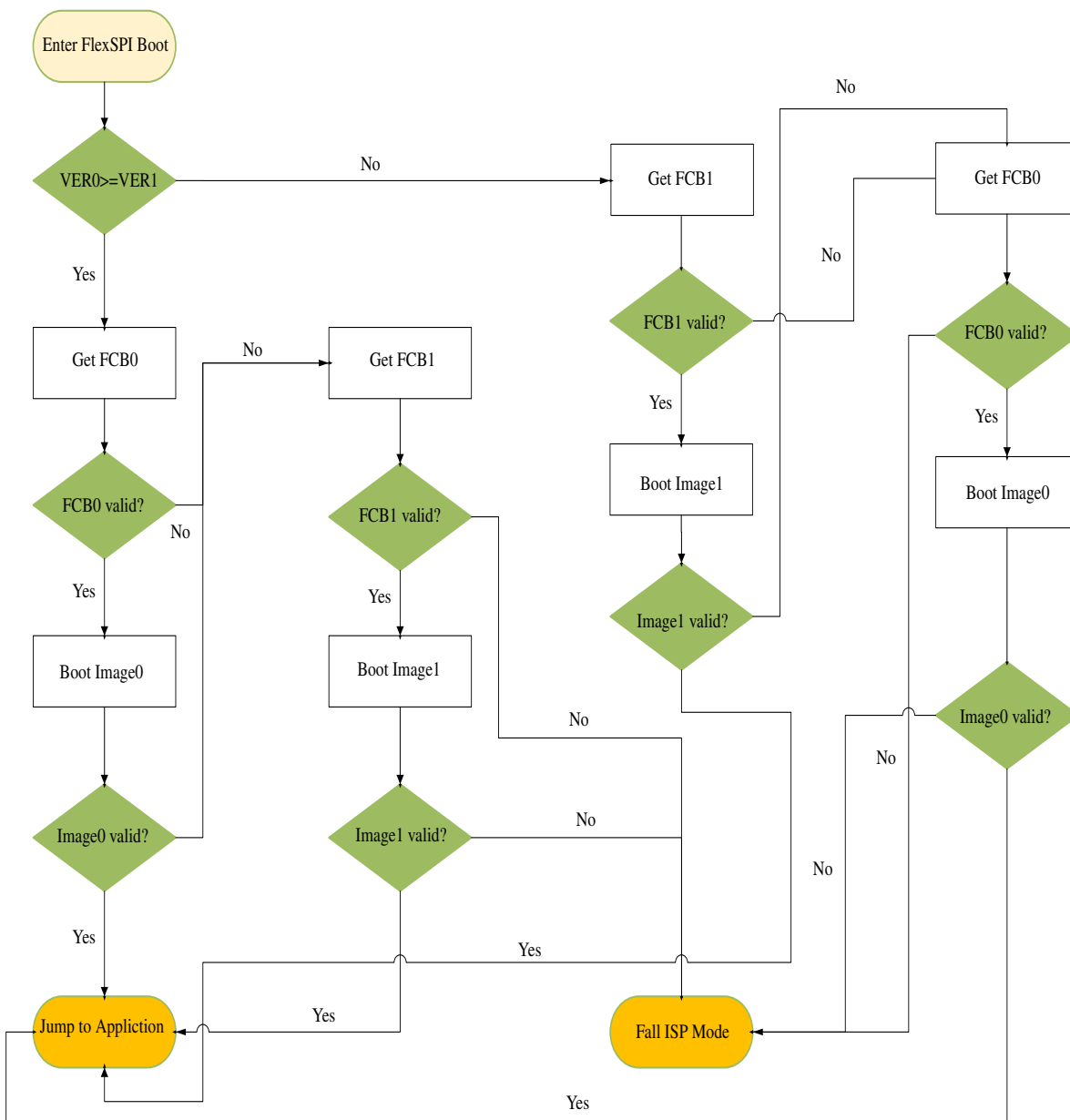


Figure 41. FLEXSPI Dual Image Ping Pong Boot Flow

18.4.1.1.3.1.4 Configure FLEXSPI NOR Device

The ROM supports booting of image from external FLEXSPI NOR flash device. This section describes how to use blhost tool to program the image into external nonvolatile memory for booting. The blhost tool uses UART, SPI, I2C, USB HID to communicate with the ROM code via ROM ISP mode.

Table 79. Pin Assignments for blhost tool communication

Boot interface	Pins	
USB		USB1_VBUS
		USB1_VDD1V8
		USB1_VDD3V3
		USB1_DM
		USB1_DP
		USB1_ID
SPI SLAVE	P1(11)	HS_SPI_SCK (FC14)
	P1(12)	HS_SPI_MISO (FC14)
	P1(13)	HS_SPI_MOSI (FC14)
	P1(14)	HS_SPI_SSELN0 (FC14)
I2C SLAVE	P0(15)	FC2_TXD_SCL_MISO
	P0(16)	FC2_RXD_SDA_MOSI
UART	P0(1)	FC0_TXD_SCL_MISO
	P0(2)	FC0_RXD_SDA_MOSI

18.4.1.1.3.1.4.1 FLEXSPI NOR FLASH boot image programming

RT500 ROM supports boot from different type NOR flash such as Quad Flash, Octal Flash, Hyper Flash which connects to FLEXSPI pins used by ROM. For FLEXSPI NOR boot, the FCB should be programmed at offset 0x08000400, and the boot image should be programmed at offset 0x08001000. The below section will provide how to config and program the boot image into NOR flash for 5 common NOR flash type.

Table 80. FLEXSPI pin assignments for NOR flash connections

Boot interface	Pins	
FlexSPI0	P1(18)	FLEXSPI0_SCLK
	P1(19)	FLEXSPI0_SS0_b
	P1(20)	FLEXSPI0_DATA0
	P1(21)	FLEXSPI0_DATA0
	P1(22)	FLEXSPI0_DATA0
	P1(23)	FLEXSPI0_DATA0
	P1(24)	FLEXSPI0_DATA0
	P1(25)	FLEXSPI0_DATA0
	P1(26)	FLEXSPI0_DATA0

Table continues on the next page...

Table 80. FLEXSPI pin assignments for NOR flash connections (continued)

Boot interface	Pins	
	P1(27)	FLEXSPI0_DATA0
	P1(28)	FLEXSPI0_DQS
	P1(29)	FLEXSPI0_SS1_b

18.4.1.1.3.1.4.2 FLEXSPI NOR FLASH configuration parameters

```
typedef struct _serial_nor_config_option
```

```
{
    union
    {
        struct
        {
            uint32_t max_freq : 4;           // Maximum supported Frequency
            uint32_t misc_mode : 4;          // miscellaneous mode
            uint32_t quad_mode_setting : 4;  // Quad mode setting
            uint32_t cmd_pads : 4;           // Command pads
            uint32_t query_pads : 4;         // SFDP read pads
            uint32_t device_type : 4;        // Device type
            uint32_t option_size : 4;        // Option size, in terms of uint32_t, size = (option_size
            + 1)
            uint32_t tag : 4;                // Tag, must be 0x0C
        } B;
        uint32_t U;
    } option0;
    union
    {
        struct
        {
            uint32_t dummy_cycles : 8;      // Dummy cycles before read
            uint32_t status_override : 8;    // Override status register value during device
            mode                             configuration
            uint32_t pinmux_group : 4;       // The pinmux group selection
            uint32_t dqs_pinmux_group : 4;  // The DQS Pinmux Group Selection
            uint32_t drive_strength : 4;     // The Drive Strength of FlexSPI Pads
            uint32_t flash_connection : 4;  // Flash connection option: 0 - Single Flash connected to
            port A, 1 - //! Parallel mode, 2 - Single Flash connected to Port B
        } B;
        uint32_t U;
    } option1;
} serial_nor_config_option_t;
```

Table 81. serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero See option1 definition for more details.

Table 82. Option0 definition

Field	Bits	Description
tag	31:28:00	The tag of the config option, fixed to 0x0C
option_size	27:24:00	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 4-Feb 3 - 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 4-Feb 3 - 8
quad_mode_setting	11:08	Quad Mode Enable Setting 0 - Not configure

Table continues on the next page...

Table 82. Option0 definition (continued)

Field	Bits	Description
		1 - Set bit 6 in Status Register 1
		2 - Set bit 1 in Status Register 2
		3 - Set bit 7 in Status Register 2
		4 - Set bit 1 in Status Register 2 vis 0x31 command
		Note: This field will be effective only if device is compliant with JESD216 only (9 longword SDFP table)
misc_mode	7:04	Miscellaneous Mode
		0 - Not enabled
		1 - Enable 0-4-4 mode for High Random Read performance
		3 - Data Order Swapped mode (for MXIC Octal Flash only)
		Note: Experimental feature, do not use in products, keep it as 0
max_freq	3:00	Max Flash Operation speed
		0 - Don't change FlexSPI clock setting

Table 83. Option1 definition

Field	Bits	Description
flash_connection	28:31:00	Flash connection option: 0 - Single Flash connected to port A, 1 - Parallel mode, 2 - Single Flash connected to Port B
drive_strength	24:27:00	The Drive Strength of FlexSPI Pads
dqs_pinmux_group	20:23	The DQS pin mux Group Selection
pinmux_group	16:19	The pin mux group selection
status_override	8:15	Override status register value during device mode configuration
dummy_cycles	7:00	Dummy cycles for read command
		0 - Use detected dummy cycle
		Others - dummy cycles provided in flash data sheet

18.4.1.1.3.1.4.3 NOR FLASH config, erase and program

Take Macronix MX25L25645G which connect to FlexSPI PORT A as an example. First, the config parameter should be stored in RAM, which will be used in configuring the FLEXSPI in next step. From the figure below, the config parameter for FLEXSPI is 0xc0000002, the config parameter is selected according to the FLASH type.

```
C:\blhost\blhost>blhost.exe -p com3 fill-memory 0x1c000 4 0xc0000002
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 42. Set the FLEXSPI config parameter in RAM

Use the config parameter stored in RAM in the previous step to config the FLEXSPI as Figure below shows, after that, you can read, erase, program the flash. For more details about the blhost tool command, you can refer to the help of the blhost tool.

```
C:\blhost\blhost>blhost.exe -p com3 configure-memory 0x9 0x1c000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 43. Config the FLEXSPI using the parameter stored in RAM

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 44. Read the NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 write-memory 0x08001000 boot_image.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 5796 (0x16a4) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 5796 of 5796 bytes.
```

Figure 45. Program the boot image into NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 46. read the boot image back via blhost tool

Generate and program the FLEXSPI NOR FCB(flash config block) into flash For FLEXSPI boot, it needs a FCB (flash config block) at offset 0x08000400, this FCB is used to config the FLEXSPI interface when boot the image from external NOR flash via FLEXSPI interface. The FCB will be generated from the previous FLEXSPI config parameter(0xc0000002) . Store the FCB generating and program parameter into the RAM which will be used in the next step to generate and program the FCB into flash at 0x08000400.

```
C:\blhost\blhost>blhost.exe -p com3 fill-memory 0x1d000 4 0xf000000f
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 47. Store the config FCB parameter into the RAM

FCB generating and program into the flash, after this step, the FCB is generated and programmed into flash at offset 0x08000400 automatically by ROM.

```
C:\blhost\blhost>blhost.exe -p com3 configure-memory 0x9 0x1d000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 48. FCB generate and program into flash at offset 0x08000400

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
46 43 46 42 00 04 01 56 00 00 00 00 01 03 03 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 01 04 02 00 00 00 00 00 00 00 00 00
00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ec 04 20 0a 00 1e 04 32 04 26 00 00 00 00 00 00
05 04 04 24 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
06 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
21 04 20 08 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 49. Read the FCB back via blhost tool

18.4.1.1.3.1.4.4 Typical NOR FLASH config parameters

Table 84. Typical NOR flash config parameters(flash connected to FlexSPI PORT A)

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
Adesto	Octal Flash	ATXP032	0xC0803006	DDR	The ATXP032 has 3 versions, REVA/B/C. For REV A, ROM only support Auto probe boot, the OTP fuse

Table continues on the next page...

Table 84. Typical NOR flash config parameters(flash connected to FlexSPI PORT A) (continued)

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
					word BOOT_CFG [0] should set to 0x7, and no need to config the FCB
Spansio	Hyper Flash	S26KS512S	0xC0233006	DDR	For Hyper flash boot, the BOOT_CFG [1] should set to 0x20 before boot
Macronix	Quad Flash	MX25L25645G	0xC0000002	SDR	
Macronix	Quad Flash	MX25L25645G	0xC0100002	DDR	
Macronix	Octal Flash	MX25UM51345G	0xC0403006	DDR	
Micron	Octal Flash	MT35XL512ABA	0xC0603006	DDR	

18.4.1.2 eMMC Boot

The boot ROM supports load image from eMMC into the on-chip SRAM and then boot from the on-chip SRAM.

18.4.1.2.1 eMMC Boot features

The eMMC boot can be performed using either uSDHC ports.

The boot ROM supports the following standard card types. The support of other card types is not guaranteed.

- eMMC Version 5.0 or earlier version.
- MMC Version 4.4 or earlier version.

And boot ROM supports the following standard mode.

- eMMC/MMC: 4-bit/8-bit; full speed SDR(26MHz), high speed SDR/DDR(52MHz).
- eMMC Version 4.4 or later version: fast boot is supported.

Boot ROM loads user application image from eMMC at a fixed address. To avoid the conflict between main boot record(MBR) and the user application image, boot ROM loads image from eMMC address 4096(byte address, block index 8).

18.4.1.2.2 eMMC NOR boot

For eMMC boot, it is mostly the same as FlexSPI Load-To-RAM boot, here give the primary step of the boot process. More details, please refer to the 4.3.1-Master boot mode chapter about eMMC boot

Set the boot ROM in ISP mode via ISP pins. Use below blhost commands to write the boot image into the eMMC.

```
blhost -p (COM XX) - fill-memory 0x1c000 0x4 0xC0000000
blhost -p (COM XX) - fill-memory 0x1c004 0x4 0x00000001
blhost -p (COM XX) - configure-memory 0x121 0x1c000
blhost -p (COM XX) - flash-erase-region 0x0 0x2000 0x121
blhost -p (COM XX) - write-memory 0x1000 image.bin 0x121
```

Use below blhost commands to write the boot image into eMMC device.

```
blhost -p (COM XX) - fill-memory 0x1c000 0x4 0xC0721625
```

```
blhost -p (COM XX) - fill-memory 0x1c004 0x4 0x00000000
blhost -p (COM XX) - configure-memory 0x121 0x1c000
blhost -p (COM XX) - flash-erase-region 0x0 0x2000 0x121
blhost -p (COM XX) - write-memory 0x1000 image.bin 0x121
```

After the image is programmed into the eMMC device, set the boot ISP pins. Boot mode and ISP Downloader modes based on ISP pins) to select the eMMC boot mode, power up the board, then the device boot from eMMC device.

18.4.1.2.3 eMMC boot configuration in OTP

The eMMC boot related configurations are allocated in BOOT_CFG2 and BOOT_CFG3 OTP word.

Table 85. BOOT_CFG2 boot configuration

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_DEVICE_TYPE		Device type connected to SDHC port.	0	1	
	eMMC_DEV	eMMC device			b'0
	Reserved	-			b'1
SDHC_BUS_WIDTH		Bus width of the device connected to SDHC.	1	2	
	MMC_4B	For eMMC use 4-bit data bus.			b'00
	MMC_8B	For eMMC use 8-bit data bus.			b'01
	MMC_4B_DDR	For eMMC use 4-bit bus in DDR mode. (SDHC_SPEED = HIGH is required)			b'10
	MMC_8B_DDR	For eMMC use 8-bit bus in DDR mode. (SDHC_SPEED = HIGH is required)			b'11
SDHC_SPEED		Speed ratings of eMMC devices.	3	2	
	NORMAL	Normal speed for eMMC devices.			b'00
	HIGH	High speed for eMMC devices.			b'01
	Reserved	-			b'10
	Reserved	-			b'11

Table continues on the next page...

Table 85. BOOT_CFG2 boot configuration (continued)

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_PWR_CYCLE_EN		Enable SDHC power cycle via USDHC_RST pad.	5	1	
	DISABLE	Disabled.			b'0
	ENABLE	Enabled.			b'1
SDHC0_PWR_POL		Polarity of power enable signal for SDHC0 port.	6	1	
	ACTIVE_LOW	Active low signal.			b'0
	ACTIVE_HIGH	Active high signal.			b'1
SDHC1_PWR_POL		Polarity of power enable signal for SDHC1 port.	7	1	
	ACTIVE_LOW	Active low signal.			b'0
	ACTIVE_HIGH	Active high signal.			b'1
SDHC_PWR_CYCLE_WAIT		Wait time after powering down the SDHC devices.	8	2	
	WAIT_20_MS	Wait 20 milli seconds.			b'00
	WAIT_10_MS	Wait 10 milli seconds.			b'01
	WAIT_05_MS	Wait 5 milli seconds.			b'10
	WAIT_2_5_MS	Wait 2.5 milli seconds.			b'11
SDHC_PWR_STABLE_WAIT		Wait time for power to stabilize after enabling power to SDHC devices.	10	1	
	WAIT_5_MS	Wait 5 milli seconds.			b'0
	WAIT_2_5_MS	Wait 2.5 milli seconds.			b'1
SDHC0_CARD_VOLTAGE		SDHC0 Voltage selection via USDHC_VSELECT	11	1	
	3V3	Uses 3v3 rail.			b'0

Table continues on the next page...

Table 85. BOOT_CFG2 boot configuration (continued)

Field Name	Enum Name	Description	Offset	Width	Value
	1V8	Uses 1v8 rail.			b'1
SDHC1_CARD_VOLTAGE		SDHC1 Voltage selection via USDHC_VSELECT	12	1	
	3V3	Uses 3v3 rail.			b'0
	1V8	Uses 1v8 rail.			b'1
MMC_RESET_PRE_IDLE_STATE		Reset the MMC to Pre-Idle state before starting boot mode	13	1	
	RESET_PRE_IDLE	Reset to pre-idle state			b'0
	NO_PRE_IDLE	Not reset to pre-idle state			b'1
SDHC_FASTBOOT_MODE		Enable eMMC fast boot mode.	14	1	
	NORMAL_MODE	eMMC operates at transfer mode			b'0
	BOOT_MODE	eMMC operates at boot mode			b'1
SDHC_FASTBOOT_ACK_EN		eMMC fast boot acknowledgement feature.	15	1	
	DISABLE	Disable ACK for boot mode			b'0
	ENABLE	Enable ACK for boot mode			b'1
SDHC_PAD		SDHC Pad setting.	16	16	
	DEFAULT	Default SDIOPADCTL			0
	OVERRIDE	SDIOPADCTL = val			val

Table 86. BOOT_CFG3 boot configuration

Field Name	Enum Name	Description	Offset	Width	Value
SDHC_DLL_TUNING_OVERRIDE_ENABLE	DEFAULT	DLL delay start 10, DLL delay step 2 (this config)	7	1	0

Table continues on the next page...

Table 86. BOOT_CFG3 boot configuration (continued)

Field Name	Enum Name	Description	Offset	Width	Value
		can cover most of user cases.)			
	OVERWRITE	Override by SDHC_DLL_TUNING_START and SDHC_DLL_TUNING_STEP			1
SDHC_DLL_TUNING_START		DLL delay start. (SDHC_DLL_TUNING_OVERRIDE_ENABLE = 1 is required)	8	8	
SDHC_DLL_TUNING_STEP		DLL delay step. (SDHC_DLL_TUNING_OVERRIDE_ENABLE = 1 is required)	16	3	
	0	DLL delay step = 1			0
	Not 0	DLL delay step = val			val

18.4.1.2.4 eMMC boot flow

The figure below shows the brief boot flow of eMMC.

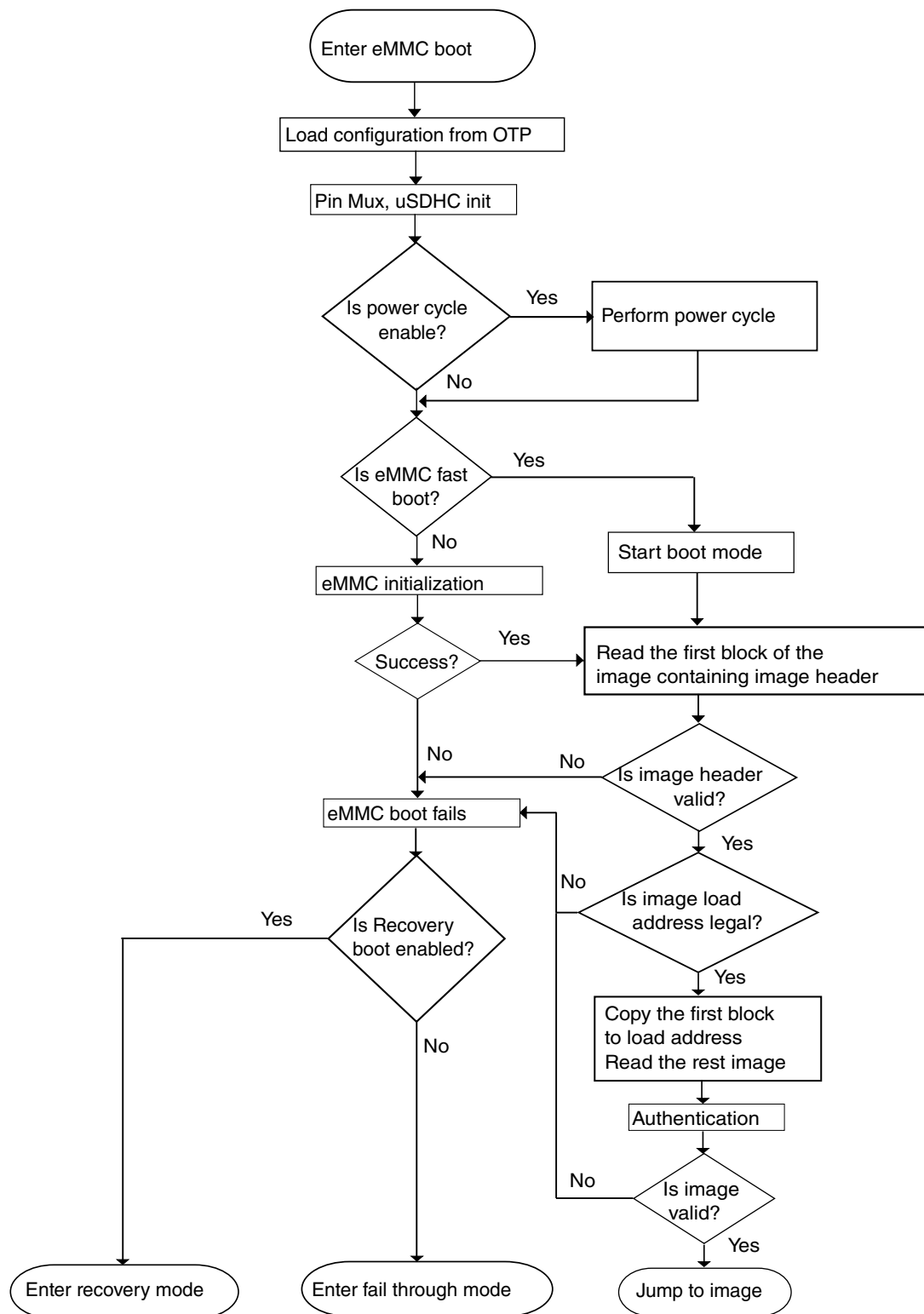


Figure 50. eMMC Boot Flow

18.4.1.3 Master boot fail log

ROM provides a way to show the reason for boot failure if master boot fails. When master boot fail, ROM will fall into ISP mode. At this time, user can use blhost tool to get the fail log, the blhost command is *get-property 8 1*.

For example, when master boot fails, then ROM enters ISP mode, in this mode, user can use below ISP command to get the last boot error, from the fig we can see that the error is 0x0501F300, which mean master boot interface initialize fail, more details can be found at below tables about the master boot fail log.

```
C:\blhost>blhost -p com16 get-property 8 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 84013824 (0x501f300)
Last Error Content = 0x0501F300
```

18.4.1.3.1 Master boot fail log table

Below tables show the probably master boot fail reason, the boot log fail value is calculated as below:

Boot fail log value = ((state) << 24) | ((substate) << 16) | ((status) << 8)

Table 87. Master boot fail log

Boot fail log	State	Substate	Status	Description	Value
Masterboot_Init_Fail	MasterBoot	SubState_Init	Status_Fail	Master boot interface initialize fail	0x0501f300
ImgLoad_InitLoad_Fail	ImgLoad	SubState_InitialLoad	Status_Fail	Master boot download image header fail	0x0a20f300
ImgLoad_RemainingLoad_Fail	ImgLoad	SubState_RemainingLoad	Status_Fail	Master boot download image header fail	0x0a21f300
BootDevice_Init_Fail	BootDevice	SubState_Init	Status_Fail	Master boot download remaining image fail	0x0901f300
BootDevice_Read_Fail	BootDevice	SubState_Call	Status_Fail	Read data from boot device fail	0x0902f300
Auth_ImageTypeCheck_Fail	Auth	AuthState_ImageTypeCheck	Status_Fail	Image type check fail	0x0b31f300
Auth_ReadKeyStore_Fail	Auth	AuthState_ReadKeyStore	Status_Fail	Read from boot device keystore fail	0x0b32f300
Auth_ReadMacKey_Fail	Auth	AuthState_ReadMacKey	Status_Fail	Get key to decrypt HMAC fail	0x0b33f300
Auth_VerifyHMAC_Fail	Auth	AuthState_VerifyHMAC	Status_Fail	Verify image HMAC fail	0x0b34f300
Auth_CrcCheck_Fail	Auth	AuthState_CrcCheck	Status_Fail	Verify image CRC fail	0x0b37f300

Table 88. Master boot, state, sub state and status value

state	value
MasterBoot	0x05
ImgLoad	0x0A
BootDevice	0x09
Auth	0x0B
substate	value
SubState_Init	0x01
SubState_Call	0x02
SubState_InitialLoad	0x20
SubState_RemainingLoad	0x21
AuthState_ImageTypeCheck	0x31
AuthState_ReadKeyStore	0x32
AuthState_ReadMacKey	0x33
AuthState_VerifyHMAC	0x34
AuthState_CrcCheck	0x37
status	value
Status_Fail	0xF3

18.4.2 Recovery Boot Mode

The ROM boot ROM supports recovery boot - an option to recover the device to a certain state once the primary boot image is corrupted due to abnormal, for example, a mistake which destroys the boot image during the image upgrade.

In RT500, the SPI NOR/EEPROM device is supported as the recovery media, and one of the SPI0- SPI7 can be specified as the interface to connect an SPI NOR/EEPROM device.

The boot ROM enters the recovery boot mode if the master boot fails and recovery boot is enabled.

When the recovery boot process starts, the boot ROM probes the presence of the SPI NOR/EEPROM device by checking the manufacturer ID, using 24MHz clock from FRO48M. Once being detected, the boot ROM starts loading the recovery image from the SPI NOR/EEPROM device to the on-chip SRAM, using the 24MHz clock, and then do integrity check/image authentication with the image.

The boot ROM jumps to the recovery boot image if the integrity check/authentication passes, otherwise, it falls through to the ISP mode if being allowed.

If the ISP boot is disabled, the boot ROM enters Lockupthechip state.

18.4.2.1 Recovery boot OTP setting

The recovery boot field is allocated at BOOT_CFG0 OTP word, starting from offset 17, 3-bit width. See the details in below table.

Table 89. Recovery boot OTP field

RECOVERY_SPI_PORT	FlexComm port to use for redundant SPI flash boot	Value
FC0	Use FlexCom0 pins P0_0 (SCK), P0_1 (MISO), P0_2 (MOSI), P0_3 (SEL)	3'b000
FC1	Use FlexCom1 pins P0_7 (SCK), P0_8 (MISO), P0_9 (MOSI), P0_10 (SEL)	3'b001
FC2	Use FlexCom2 pins P0_14 (SCK), P0_15 (MISO), P0_16 (MOSI), P0_17 (SEL)	3'b010
FC3	Use FlexCom3 pins P0_21 (SCK), P0_22 (MISO), P0_23 (MOSI), P0_24 (SEL)	3'b011
FC4	Use FlexCom4 pins P0_28 (SCK), P0_29 (MISO), P0_30 (MOSI), P0_31 (SEL)	3'b100
FC5	Use FlexCom5 pins P1_3 (SCK), P1_4 (MISO), P1_5 (MOSI), P1_6 (SEL)	3'b101
FC6	Use FlexCom6 pins P3_25 (SCK), P3_26 (MISO), P3_27 (MOSI), P3_28 (SEL)	3'b110
FC7	Use FlexCom7 pins P4_0 (SCK), P4_1 (MISO), P4_2 (MOSI), P4_3 (SEL)	3'b111

18.4.2.2 The Recovery boot flow

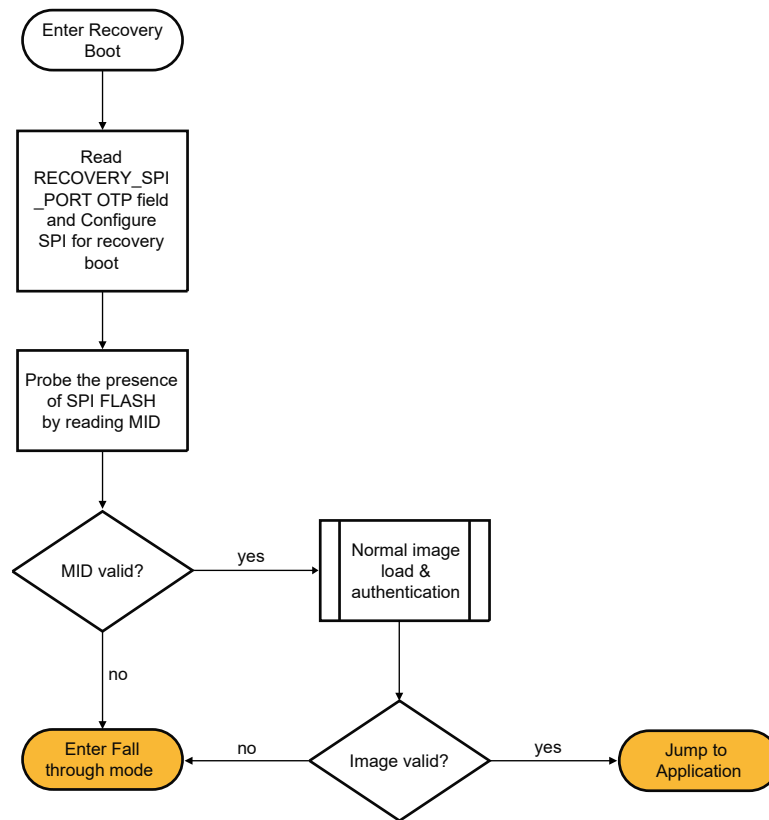


Figure 51. Recovery boot flow

18.4.3 Serial Boot Mode

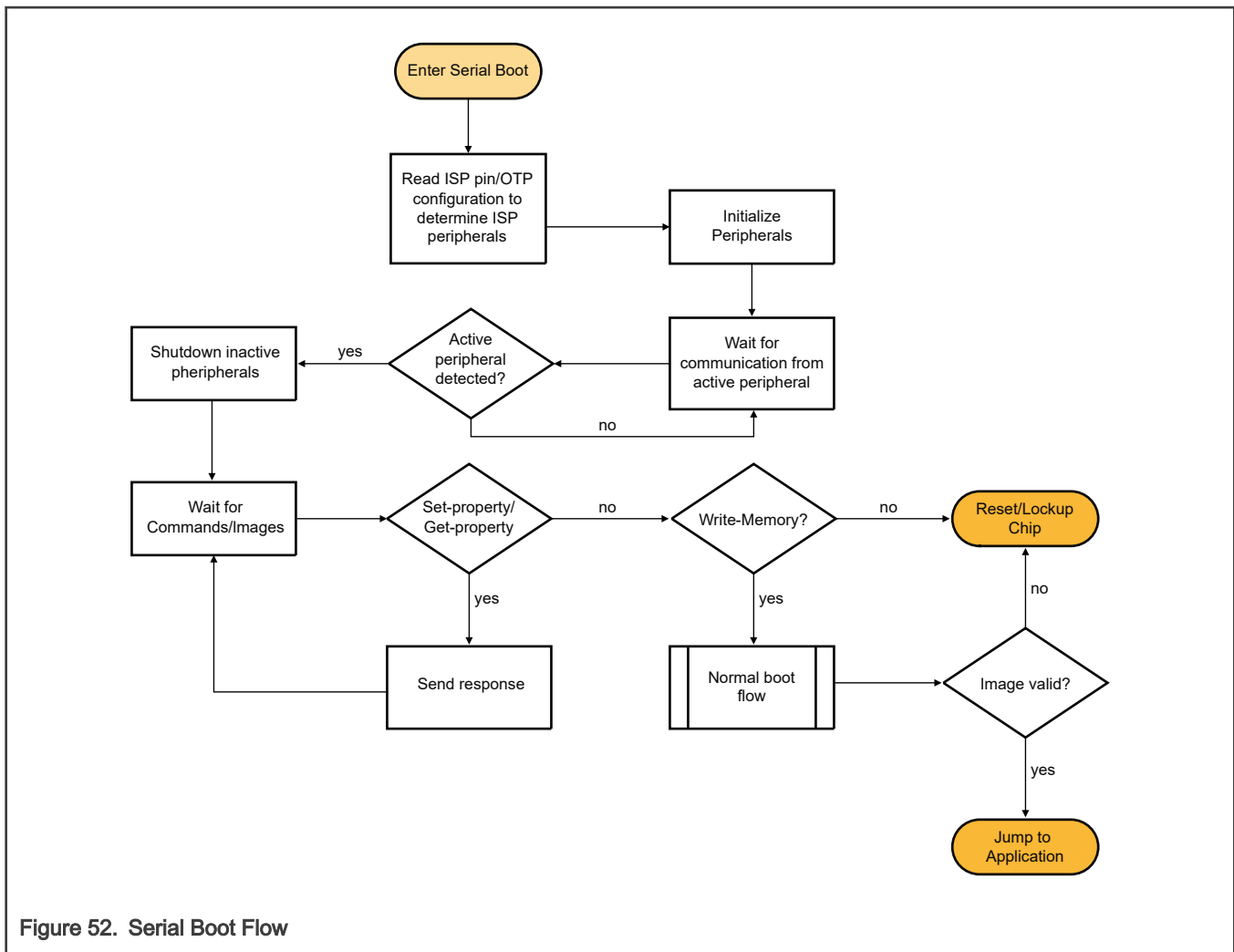
The boot ROM supports loading image from a Serial Interface (for example, UART) to on-chip SRAM and then boot to the image. The supported serial boot peripherals is UART.

The Serial Boot reuses the same packet format and simplified command set from the ISP boot.

The support commands are as following:

- Get-Property
- Set-Property
- Write-Memory

18.4.3.1 Serial Boot Flow



18.4.3.2 Serial Boot via UART

The boot ROM reads the boot image via the UART interface using 8-N-1 mode when being selected as the Serial Boot interface. The commands involved in this mode are listed as follows:

- Ping (Used for autobaud detection)
- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly).
- Write-memory

The maximum payload size in the write-memory command is 512 bytes. The default maximum UART baud rate can operate up to 1M at factory setting, and it can go as high as 4M when the core is running at high-speed boot mode.

18.4.3.3 Serial Boot via SPI

The boot ROM supports loading a boot image via the SPI interface, where the SPI works under SPI slave, Mode 3, and 8-bit data mode.

The boot ROM responds to a host system on the configured SPI interface. A transfer starts once SSEL goes LOW. The default maximum SPI speed is 24MHz at factory setting, and it can go as high as 50MHz when the core is running at high-speed boot mode.

The commands involved in this mode are as follows:

- Set-Property
- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly).
- Write-memory

The maximum payload size in the write-memory command is 512 bytes.

18.4.3.4 Serial Boot via I2C

The boot ROM can load a boot image via the I2C interface, where the I2C works under Slave mode with 0x10 as the 7-bit address. The command used in this boot mode is:

- Get-Property (Optional, it is used for the host tool to get the maximum payload size, the host can skip this command and packetize the image using 512-byte payload size directly)
- Write-Memory.

The maximum payload size in the write-memory command is 512 bytes.

18.4.3.5 Serial Boot via USB HID

The boot ROM can load a boot image and boot to it via the USB, where the USB works under HID mode. Select the boot mode as HID mode and make sure the HID is successfully emulated, then use below command to load your boot image into RAM to run. The maximum payload size in the write-memory command is 512 bytes.

Blhost.exe -u 0x1fc9,0x0023 write-memory 0x1c000 image.bin

0x2fc9,0x0023 is PID,VID of the HID device.

image.bin is the boot image

18.4.4 Lockupthechip mode

When booting the application image fails, and if FallthroughISP boot mode is disabled (BOOT_CFG[0][DEFAULT_ISP_MODE] == DISABLE_ISP), the boot ROM enters Lockupthechip mode. It will remain in this state until the MCU is reset. If the DCFG_CC_SOCU settings enable debug access, a debugger can gain control and exit the boot ROM.

Once secure boot is enabled only authenticated interactions are allowed with the device to assure it is in a trusted facility. Hence, ISP fall through is only supported where signed SB2 file could be used to reprogrammed/recover the device. Debug access in this scenario (invalid image with secure boot enabled) is disabled.

18.5 Boot ROM Status Error Codes

This section describes the status error codes that the boot ROM returns to the host.

Table 90. Boot ROM Status Error Codes, sorted by Value

Error Code	Value	Description
Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_OutOfRange	3	Requested value is out of range.

Table continues on the next page...

Table 90. Boot ROM Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FLEXSPINOR_ProgramFail	20100	The FlexSPI Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	The FlexSPI Sector Erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	The FlexSPI Chip Erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	The FlexSPI command timeout.
kStatus_FlexSPINOR_NotSupported	20104	The FlexSPI PageSize overflow.
kStatus_FlexSPINOR_WriteAlignmentError	20105	The FlexSPI Alignment error
kStatus_FlexSPINOR_CommandFailure	20106	The FlexSPI Erase/Program Verify Error
kStatus_FlexSPINOR_SFDP_NotFound	20107	The FlexSPI SFDP read failure
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	The FlexSPI Unrecognized SFDP version
kStatus_FLEXSPINOR_Flash_NotFound	20109	The FlexSPI Flash detection failure
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	The FlexSPI DDR Read dummy probe failure
kStatus_UnknownCommand	10000	The requested command value is undefined.
kStatus_SecurityViolation	10001	Command is disallowed because flash security is enabled.
kStatus_AbortDataPhase	10002	Abort the data phase early.
kStatus_Ping	10003	Internal: received ping during command phase.
kStatus_CommandUnsupported	10003	The requested command value is unsupported
kStatusRomLdrSectionOverrun	10100	The loader has finished processing the SB file.
kStatusRomLdrSignature	10101	The signature of the SB file is incorrect.
kStatusRomLdrSectionLength	10102	The section length in chunks is invalid.
kStatusRomLdrUnencryptedOnly	10103	An encrypted SB file has been sent and decryption support is not available.
kStatusRomLdrEOFReached	10104	The end of the SB file has been reached.
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.

Table continues on the next page...

Table 90. Boot ROM Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was found in the SB file.
kStatusRomLdrIdNotFound	10108	There was no bootable section found in the SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file failed.
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file sent is unencrypted, and security on the target is enabled.
kStatusRomLdrResetReturned	10114	The SB file reset operation has unexpectedly returned.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatusMemoryReadFailed	10201	Memory Read Failed
kStatusMemoryWriteFailed	10202	Memory Write failed
kStatusMemoryCumulativeWrite	10203	Cumulative Write happened due to write to un-erased FLASH region
kStatusMemoryNotConfigured	10205	Memory is not configured yet before access
kStatusMemoryAlignmentError	10206	Alignment Error happened during access to memory
kStatusMemoryVerifyFailed	10207	Verifying operation failed after erasing/programming FLASH
kStatusMemoryWriteProtected	10208	The memory to be written is protected
kStatusMemoryAddressError	10209	The Memory address is invalid/wrong
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.

Table continues on the next page...

Table 90. Boot ROM Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid, but addresses are out of range.
kStatus_OTP_InvalidAddress	52801	Invalid OTP address
kStatus_OTP_ProgramFail	52802	OTP Programming failed
kStatus_OTP_CrcFail	52803	OTP CRC check failed
kStatus_OTP_Error	52804	Error happened during OTP operation
kStatus_OTP_EccCheckFail	52805	ECC check failed during OTP operation
kStatus_OTP_Locked	52806	OTP field was locked when programming
kStatus_OTP_Timeout	52807	OTP operation timed out
kStatus_OTP_CrcCheckPass	52808	OTP CRC check passed

18.6 External Memory Support

This section describes the external memory devices supported by the boot ROM ISP command. To use an external memory device correctly, the device must be enabled with the corresponding configuration profile. If the external memory device is not enabled, then it cannot be accessed by the ROM ISP command. The boot ROM enables specific external memory devices using a pre-assigned memory identifier, supported external memory devices, and memory identifiers are shown below.

Table 91. Memory ID for external memory devices

Memory Identifier	External Memory device
0x09	'Serial NOR over FlexSPI module'
0x110	'Serial 1-bit NOR/EEPROM over SPI module'
0x121	'eMMC over uSDHC'

18.6.1 Serial NOR Flash through FlexSPI

The boot ROM supports read, write, and erase external Serial NOR Flash devices via the FlexSPI Module. Before accessing Serial NOR Flash devices, the FlexSPI module must be configured properly, using a simplified FlexSPI NOR Config block or a complete 512-byte FlexSPI NOR Configuration Block. Boot ROM can generate the 512 byte FlexSPI NOR Configuration Block based on the simplified Flash Configuration Option Block for most Serial NOR Flash devices in the market.

18.6.1.1 FlexSPI NOR boot

Example boot image is available from SDK package for RT5XX. The SDK package can be downloaded from <https://mcuxpresso.nxp.com/en/welcome>. Users can choose a demo, build the demo and get a .bin file of the boot image, called image.bin file temporarily. Since boot image needs to place into external memory, ROM supports to program the boot image into external memory. To program the external memory, the user needs first get the boot ROM Host Application (<https://www.nxp.com/webapp/sps/download/preDownload.jsp>). This tool is called blhost tool which is used to communicate with the boot ROM. The

blhost tool User's guide can also be found in the above link. Before using the tool to send the ISP command to boot ROM, boot ROM should be first configured to enter serial ISP(In-System Programming) mode. During the development cycle, it is recommended to use ISP pins to select the boot ROM to enter serial ISP mode by the ISP pins selection mentioned below.

Steps to enter ISP mode:

- Power off the power.
- Set the jumpers of ISP pins in the state listed in the below table.
- Power on the board. When all these steps are done, the boot ROM enters serial ISP mode. When the boot ROM is in ISP mode, ISP communicates with boot ROM via the blhost tool can start.

Table 92. Boot mode and ISP Downloader modes based on ISP pins

Boot mode	ISP2 PIO3_29	ISP1 PIO3_28	ISP0 PIO1_15	Description
Serial ISP (UART, SPI, I2C, USB-HID)	High	High	Low	The Serial Interface (UART, SPI, and I2C) is used to program OTP, external Flash, eMMC device.
Serial ISP (USB-HID)	Low	High	Low	The Serial Interface USB-HID is used to program OTP, external Flash, eMMC device.

Boot ROM supports four interfaces to communicate with the blhost tool, UART, I2C, SPI, HID. ROM defines a communication protocol for the command and data transition. At this, we take UART as an example. For other interfaces usage, please refer to the blhost tool user guide.

Before using the UART as the communication interface, the PC COM port should connect the TXD/RXD pins to boot ROM RXD/TXD pins. The below table shows the UART pins used for boot ROM.

Table 93. UART pins used for boot ROM

Boot interface	Pin(s)	Function/Comment
UART	PIO0_1	FC0_TXD_SCL_MISO_WS
	PIO0_2	FC0_RXD_SDA_MOSI_DATA

When the pin connection is ready, and the boot ROM is also in ISP mode, The communication between Host(blhost) and boot ROM can start. Before programing the serial NOR flash device, we need to use the blhost tool to tell the boot ROM to configure the FlexSPI interface, then the other ISP commands such as read/write/erase can be sent to boot ROM via blhost tool to access the serial NOR flash connected on the FlexSPI interface of the MCU. Take flash MX25UM51345G as an example, below command shows the steps of configuring, erase, program the boot image to the external NOR flash connected to the FlexSPI interface

```
blhost -p (COM XX) - fill-memory 0x1c000 0x04 0xc0403001
blhost -p (COM XX) - configure-memory 0x09 0x1c000
blhost -p (COM XX) - flash-erase-region 0x08000000 0x10000
blhost -p (COM XX) - fill-memory 0x1c004 0x04 0xf000000f
blhost -p (COM XX) - configure-memory 0x09 0x1c004
blhost -p (COM XX) - write-memory 0x08001000 image.bin
```

Below is some detail usage of the blhost tool command.

```
blhost -p (COM XX) - fill-memory 0x1c000 0x04
0xc0403001
```

This fills the flash config parameter into the SRAM, 0x1c000 is an SRAM address can be used on the device, 0x4 is the parameter length, 0xc0403001 is flash configure parameter, for different flash, and different flash work mode(such as quad, octal, SDR, DDR)may have different configure parameters.

```
blhost -p (COM XX) - configure-memory 0x09 0x1c000
```

This is to configure the FlexSPI interface using the parameter wrote in RAM address 0x1c00. After this command is sent, the FlexSPI finishes the configuration, and it is ready to do the operation of the external flash.

```
blhost -p (COM XX) -  
    flash-erase-region 0x08000000 0x10000
```

This is to erase the external NOR flash from address 0x08000000. The erase size is 0x10000.

```
blhost -p (COM XX) - fill-memory 0x1c004 0x04  
    0xf000000f
```

```
blhost -p (COM XX) - configure-memory 0x09 0x1c004
```

This step is to fill the parameter of generating Flash Config Block(FCB) for an external flash, and trigger the generating FCB and program the FCB at flash offset 0x400. FCB is used to configure the FlexSPI interface during the boot process, when FlexSPI boot starts, ROM first to get the FCB at 0x400 use a default 1-bit mode, and then use the FCB to configure the FlexSPI to access the NOR flash.

```
blhost -p (COM XX) -  
    write-memory 0x08001000 image.bin
```

This step is to program the boot image into the external NOR flash address at address 0x08001000, make sure that the boot image is programmed to the external flash offset 0x1000. Image offset on different boot media) Since the 0x08000000 is the FlexSPI memory-mapped AHB address, so to program the to the flash device at 0x1000, the program flash address of the blhost is 0x08001000. After all those steps done, the boot image is successfully programmed into the external memory at offset 0x1000, and the FlexSPI boot FCB is also programmed into offset 0x400. Then the boot can be started. Power off the board, set boot mode via below ISP pins, then power on the the board, then the device boots up the boot image programmed before.

Boot mode	ISP2 PIO3_29	ISP1 PIO3_28	ISP0 PIO1_15	Description
FlexSPI Boot from Port A	Low	High	High	Boot from Quad/Octal SPI Flash devices connected to the FlexSPI interface 0. The RT5xx will look for a valid image in external Quad/Octal SPI Flash device.If there is no valid image found, the RT5xx will enter ISP boot mode.

18.6.1.2 FlexSPI NOR Configuration Block(FCB)

This section describes the external memory devices supported by the boot ROM ISP command. To use an external memory device correctly, the device must be enabled with the corresponding configuration profile. If the external memory device is not enabled, then it cannot be accessed by the ROM ISP command. The boot ROM enables specific external memory devices using a pre-assigned memory identifier, supported external memory devices, and memory identifiers are shown below.

Field	Offset	Size(bytes)	Description
tag	0x000	4	Config Block tag, must be 0x42464346

Table continues on the next page...

Table continued from the previous page...

version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	-
readSampleClkSrc	0x00c	1	Read Sampling Clock source option 0 - Internal loopback 1 - Loopback from DQS pad 2 - Loopback from SCK pad 3 - External DQS signal
csHoldTime	0x00d	1	CS Hold time in terms of Flash clock cycles Recommended value is 3
csSetupTime	0x00e	1	CS setup time in terms of Flash clock cycles Recommended value is 3
columnAddressWidth	0x00f	1	Column Address WidthSet to 3 for HyperFLASH Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
deviceModeType	0x011	1	-
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register 2 for some QSPI FLASH devices

Table continues on the next page...

Table continued from the previous page...

configCmdEnable	0x01c	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01d	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command typesSupport up to 3 sequences
reserved	0x02c	4	
configCmdArgs	0x030	12	Configure Command ArgumentSupport up to 3 arguments
reserved	0x03c	4	-
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type1- Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads

Table continues on the next page...

Table continued from the previous page...

			4 - Quad pads 8 - Octal pads
serialClkFreq	0x046	1	Flash Frequency SDR mode: 1 - 30MHz 2 - 50MHz 3 - 60MHz 4 - 80MHz 5 - 100MHz 6 - 120MHz 7 - 133MHz 8 - 166MHz DDR mode: 1 - 30MHz 2 - 50MHz 3 - 60MHz 4 - 80MHz 5 - 100MHz 6 - 120MHz 7 - 133MHz 8 - 166MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/ erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH
reserved	0x048	8	Reserved
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
sflashB1Size	0x058	4	Size of Flash connected to B1
sflashB2Size	0x05C	4	Size of Flash connected to B2

Table continues on the next page...

Table continued from the previous page...

csPadSettingOverride	0x060	4	Pad override value for CS pinUsing this value to configure CS pin if it is non-zero, otherwise use default ROM setting
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06c	4	Dqs pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time for PORT A and PORT B
busyOffset	0x07c	2	Busy offset, valid value: 0-31
busyBitPolarity	0x07e	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0 when flash device is busy
lookupTable	0x080	256	16 LUT sequences, each sequence consists of 4 words, see FLEXSPI chapter for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1b0	16	Reserved
pageSize	0x1c0	4	Page size of Flash
sectorSize	0x1c4	4	Sector size of Flash
ipcndSerialClkFreq	0x1c8	1	Serial Clock Frequency for IP command 0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1c9	1	If the sector size is the same with block size. 0 - No

Table continues on the next page...

Table continued from the previous page...

			1 - Yes
isDataOrderSwapped	0x1ca	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1cb	1	Reserved
serialNorType	0x1cc	1	Serial NOR Flash type: 0/1/2/3 0- Serial Nor Type StandardSPI 1- Serial Nor Type HyperBus 2- Serial Nor Type XPI 3- Serial Nor Type NoCmd
needExitNoCmdMode	0x1cd	1	Need to exit No Cmd mode before other IP command
halfClkForNonReadCmd	0x1ce	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1cf	1	Need to Restore No Cmd mode after IP command execution
blockSize	0x1d0	4	Flash Block size
flashStateCtx	0x1d4	4	Flash State Context, which is used to restore the flash to default state
reserved	0x1d8	40	Reserved

NOTE

To customize the LUT sequence for some specific device, users need to enable "lutCustomSeqEnable" and fill in the corresponding "lutCustomSeq" field specified by the command index below.

For Serial (SPI) NOR, the pre-defined LUT index is as following:

Table 94. Lookup Table index pre-assignment for FlexSPI NOR

Name	Index in lookup table	Description
Read	0	Read command Sequence
ReadStatus	1	Read Status command
ReadStatusXpi	2	Read Status command under OPI mode
WriteEnable	3	Write Enable command sequence
WriteEnableXpi	4	Write Enable command under OPI mode

Table continues on the next page...

Table 94. Lookup Table index pre-assignment for FlexSPI NOR (continued)

Name	Index in lookup table	Description
EraseSector	5	Erase Sector Command
EraseBlock	8	Erase Block Command
PageProgram	9	Page Program Command
ChipErase	11	Full Chip Erase
ExitNoCmd	15	Exit No Command Mode as needed
Reserved	6,7,10,12,13,14	All reserved indexes can be freely used for other purposes

18.6.1.3 FlexSPI NOR Configuration Option Block

The FlexSPI NOR Configuration Option Block is organized by 4-bit unit, and it is expandable, the current definition of the block is as shown in the below table.

The ROM detects FCB using read SFDP command which is supported by most flash devices those are JESD216(A/B) compliant. However, JESD216A/B only defines the dummy cycles for Quad SDR read. In order to get the dummy cycles for DDR/DTR read mode, ROM supports auto probing by writing test patterns to offset 0x200 on the external memory devices. To get optimal timing, the readSampleClkSrc set to 1 for Flash devices that do not support external provided DQS pad input. It is set to 3 for flash devices that support external provided DQS pad input such as OctalFlash /HyperFLASH. FlexSPI_DQS pad is not used for other purposes.

Offset	Field	Description							
0	Option0	TAG[31:28]	Option size [27:24]	Device Detection Type[23:20]	Query CMDPad(s)[19:16]	CMDPad(s)[15:12]	QuadEnableType[11:8]	Misc[7:4]	MaxFreq[3:0]
		0x0c	Size in bytes = (Option Size + 1)* 4	0 - QuadSPI SDR 1 - QuadSPI DDR 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - MXICOPi DDR 6 - MicronOPi DDR	0 - 1 2 - 4 3 - 8	0 - 1 2 - 4 3 - 8	1 - QE bit is bit6 in StatusReg 1 2 - QE bit is bit1 in StatusReg 2 3 - QE bit is bit7 in StatusReg 2 4 - QEbit is bit 1 in StatusReg 2, enablecommand is 0x31	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only)	Max Flash Operation speed 0 - Don't change FlexSPI clock setting

Table continues on the next page...

Offset	Field	Description							
				8 -AdestoO PIDDR			This field will be effective only if device is compliant with JESD216 only (9 longword SDFP table)	5 - Select the FlexSPI data sample source as internal loop back, more details please refer FlexSPI usage 6 - Config the FlexSPI NOR flash running at stand SPI mode Experimental feature, do not use in products, keep it as 0.	

Offset	Option	Description			
4	Option1 Optional	flash_connection	Reserved	status_override	Dummy Cycle
		[31:28]	[27:16]	[15:8]	[7:0]
		Flash connection option: 0 - Single Flash connected to port A 1 - Parallel mode 2 - Single Flash connected to Port B		Override status register value during device mode configuration	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

- Tag - Fixed as 0x0C
- Option Size - Provide scalability for future use, the option block size equals to (Option size + 1) * 4 bytes
- Device Detection type - SW defined device types used for config block autodetection
- Query Command Pad(s) - Command pads (1/4/8) for the SFDP command

- CMD pad(s) - Commands pads for the Flash device (1/4/8), for device that works under 1-1-4, 1-4-4, 1-1-8 or 1-8-8 mode, CMD pad(s) value is always 0x0, for devices that only support 4-4-4 mode for high performance, CMD pads value is 2, for devices that only support 8-8-8 mode for high performance, CMD pads value is 3
- Quad Enable Type - Specify the Quad Enable sequence, only applicable for device that only JESD216 compliant, this field is ignored if device support JESD216A or later version
- Misc - Specify miscellaneous mode for selected flash type
- Max Frequency - The maximum work frequency for specified Flash device
- Dummy Cycle - User provided dummy cycles for SDR/DDR read command
- Status override - Override status register value during device mode configuration
- Flash connection – Select the FlexSPI Port A/B

Use cases for FlexSPI NOR Configuration Block:

- QuadSPI NOR - Quad SDR Read: option0 = 0xc0000006 (120MHz)
- QuadSPI NOR - Quad DDR Read: option0 = 0xc0100003 (60MHz)
- HyperFLASH 1V8: option0 = 0xc0233007 (133MHz)
- HyperFLASH 3V0: option0 = 0xc0333006 (120MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433006(120MHz)
- Micron Octal DDR: option0=0xc0600006 (120MHz)
- Micron OPI DDR: option0=0xc0603006 (120MHz), SPI->OPI DDR
- Micron OPI DDR (DDR read enabled by default): option0=0xc0633006(120MHz)
- Adesto OPI DDR: option0=0xc0803007(133MHz)

Program Serial NOR Flash device using FlexSPI NOR Configuration Option

The boot ROM supports generating complete FCB using configure-memory command. It also supports programming the generated FCB to the start of the flash memory using a specific option "0xF000000F". Here is the example for configuring and accessing HyperFLASH (Assuming it is a blank HyperFLASH device).

```
blhost -u (PID,VID) - fill-memory 0x1c000 0x04 0xc0233007 (write option block to SRAM
address 0x1c000)
blhost -u (PID,VID) - configure-memory 0x09 0x1c000 (configure HyperFLASH using
option block)
blhost -u (PID,VID) - fill-memory 0x1d000 0x04 0xf000000f (write specific
option to SRAM address 0x1d000)
blhost -u (PID,VID) - configure-memory 0x09 0x1d000 (program FCB to the
offset 0x400 of HyperFLASH device)
blhost -u (PID,VID) - flash-erase-region (addr) (size) (erase the
HyperFLASH device from addr of size)
blhost -u (PID,VID) - write-memory (addr) image.bin (write the
image to addr)
```

NOTE

Flash Block erase is not supported for FlexSPI memory, when flash-erase-region command is issued from blhost.

18.6.1.4 Reset the NOR FLASH device to SPI mode

The ROM requires the Serial NOR FLASH to stay in the default working mode during access to the FLASH device, for example, reading the SFDP table or read Manufacturer Identifier while configuring the Serial NOR FLASH device via the ROM FLEXSPI NOR driver. Otherwise, if FLASH is in other modes, for example, in the QPI mode or OPI mode, ROM cannot correctly read the SFDP table or the manufacturer identifier. The consequence is that FLEXSPI NOR boot would fail in the auto-probe mode,

or configure-memory command may report an error in ISP mode, or the FLEXSPI NOR ROM API reports an error code during execution. Here are two typical use cases that may result in the issues described above.

Scenario 1 - the user application has configured the FLASH device to QPI/OPI mode for FLEXSPI NOR boot. The unexpected reset source, for example, WDOG reset or core lockup event, will trigger the boot core reset, and the reboot sequence occurs in such a scenario while the FLASH device remains in the QPI/OPI mode. During the reboot, ROM assumes that the FLASH device works under standard SPI mode and tries to read the data via standard SPI protocol from FLASH device. However, the read and the upcoming FLASH boot will fail because the FLASH device is not in the expected mode.

Scenario 2 - The user application configures the FLASH device to the deep power-down mode for low-power consideration, and then switches itself to low-power mode as well to save power. During wake-up, the reboot sequence takes place, and the FLASH is in deep power mode, and it cannot accept any FLASH access command until being recovered by specific wake-up sequences. Hence, the FLEXSPI NOR FLASH boot may fail in such a case.

The problems mentioned above have a significant impact on system-level reliability. The ROM addresses the impact by maintaining the FLASH state context via the FLASH context register, which logs the below information:

- The FLASH default working mode on POR reset
- The current FLASH working mode
- The sequence to reset to default working mode

Before accessing the FLASH device, ROM reads the context parameter and resets the FLASH to the default working mode via the specified reset pattern first. The FLASH device is in the default operating mode, and consequently, the SW can access FLASH as expected (including rebooting, configure-memory command, or the FLASH API call).

The following section explains the detailed information about the FLASH context register.

```
typedef union
{
    struct
    {
        uint8_t por_mode;
        uint8_t current_mode;
        uint8_t reserved;
        uint8_t restore_sequence;
    } B;
    uint32_t U;
} FLASH_run_context_t;
```

Bitfield	Description	Definition
por_mode	FLASH Device Power reset running mode	kFlashInstMode_ExtendedSpi = 0x00,
		kFlashInstMode_QPI_SDR = 0x41,
		kFlashInstMode_QPI_DDR = 0x42,
		kFlashInstMode_OPI_DDR = 0x82,
current_mode	FLASH Device current running mode	kFLASHInstMode_ExtendedSpi = 0x00
		kFLASHInstMode_QPI_SDR = 0x41
		kFLASHInstMode_QPI_DDR = 0x42
		kFLASHInstMode_OPI_DDR = 0x82
restore_sequence	FLASH reset sequence of standard SPI mode	kRestoreSequence_Send_66_99 = 6
		kRestoreSequence_Send_6699_9966 = 7
		kRestoreSequence_Send_06_FF

The POR reset context register used by ROM is in SYSCON, and the address is 0x50002380. Take IMXRT500 EVK board for example, if the user needs to reset the FLASH to SPI mode, such as waking up the FLASH from deep power-down mode, or reconfigure the FLASH use ISP command, or use the ROM auto probe boot feature (when the FLASH is in OPI DDR mode), the user needs to do below actions.

FLASH Type: MX25UM51345G:

```
FLASH_run_context_t run_ctx = {.U = 0 };
    run_ctx.B.current_mode = kFLASHInstMode_OPI_DDR; // Set the current FLASH mode
    run_ctx.B.restore_sequence = kRestoreSequence_Send_6699_9966; // Select the
FLASH reset sequences
    *(volatile uint32_t *) (0x50002380) = run_ctx.U // Update the context register
```

After this operation, the context register stores the current running mode of the NOR FLASH. When it is ISP reconfiguring or reboot, ROM uses the context value to decide the way to reset the FLASH. For the FLASH resetting sequence, please refer to the FLASH device datasheet for Send_66_99, Send_6699_9966, Send_06_FF.

18.6.2 eMMC through uSDHC

BootROM supports booting from eMMC devices, the ROM supports to flash the boot image into the eMMC devices. This section explains the usage of eMMC via ROM ISP command.

18.6.2.1 eMMC Configuration Block

eMMC must be initialized before accessing it. The eMMC configuration block is used to tell ROM how to initialize the eMMC device. To use the fast boot feature offered by BootROM, eMMC also must be pre-configured. The fast boot configuration is also included in the eMMC configuration block.

Table 95. eMMC Configuration Block Definition

Word Index	Bit Field	Name	Description
Word0	[31:28]	TAG	eMMC configuration block tag used to mark if the block is valid or not. 0xC: Valid block Others: Invalid
	27	RSV	0x0
	[26:24]	PARTITION_ACCESS	Select eMMC partition which the Flashloader write the image or data to 0: User data area 1: Boot partition 1 2: Boot partition 2 3: RPMB 4: General Purpose partition 1 5: General Purpose partition 2 6: General Purpose partition 3 7: General Purpose partition 4
	23	RSV	0x0

Table continues on the next page...

Table 95. eMMC Configuration Block Definition (continued)

Word Index	Bit Field	Name	Description
	[22:20]	BOOT_PARTITION_ENABLE	Select the boot partition used for fast boot. Only valid when BOOT_CONFIG_ENABLE is set 0: Not enabled 1: Boot partition 1 2: Boot partition 2 3-6: Reserved 7: User data area
	[19:18]	RSV	0x0
	[17:16]	BOOT_BUS_WIDTH	Select the bus width used for fast boot. 0: x1(SDR) x4(DDR) 1: x4(SDR,DDR) 2: x8(SDR,DDR) 3: Reserved
	[15:12]	TIMING_INTERFACE	Select the bus timing when Flashloader accesses eMMC memory. 0: Normal 1: HS2: HS200(Not Supported) 3: HS400(Not Supported) 4-15: Reserved
	[11:8]	BUS_WIDTH	Select the bus width when Flashloader accesses eMMC memory. 0: x1 SDR 1: x4 SDR 2: x8 SDR 3-4: Reserved 5: x4 DDR 6: x8 DDR 7-15: Reserved
	[7:6]	RSV	0x0

Table continues on the next page...

Table 95. eMMC Configuration Block Definition (continued)

Word Index	Bit Field	Name	Description
	[5:4]	BOOT_MODE	0: Normal1: HS2: DDR3: Reserved
	3	RESET_BOOT_BUS_CONDITIONS	Configure eMMC behavior after exiting fast boot 0: Reset to x1,SDR,Normal 1: Retain boot config
	2	BOOT_ACK	Configure eMMC ACK behavior at fast boot. 0: NO ACK 1: ACK
	1	RSV	0x0
	0	BOOT_CONFIG_ENABLE	Determine if write fast boot configurations into eMMC or not. 0: Boot configuration will be ignored. 1: Boot configuration will be written into device
Word1	[31:26]	RSV	0x0
	[25:24]	PWR_DOWN_TIME	eMMC power down delay time before power up the eMMC card. Only valid when PWR_CYCLE_ENABLE is enable 0: 20ms 1: 10ms 2: 5ms 3: 2.5ms
	23	PWR_POLARITY	eMMC power control polarity. Only valid when PWR_CYCLE_ENABLE is enable 0: Power down when uSDHC.RST set low. 1: Power down when uSDHC.RST set high.

Table continues on the next page...

Table 95. eMMC Configuration Block Definition (continued)

Word Index	Bit Field	Name	Description
	[22:21]	RSV	0x0
	20	PWR_UP_TIME	eMMC power up delay time to wait voltage regulator output stable Only valid when PWR_CYCLE_ENABLE is enabled 0: 5 ms 1: 2.5 ms
	19	PWR_CYCLE_ENABLE	Execute a power cycle before starting the SD initialization progress 0: Disable 1: Enable
	[18:4]	RSV	0x0
	[3:0]	INSTANCE	Choose the uSDHC different instance used for eMMC device: 0: uSDHC0 is selected 1: uSDHC1 is selected

NOTE

Fast boot configuration includes BOOT_PARTITION_ENABLE, BOOT_BUS_WIDTH, BOOT_MODE, RESET_BOOT_BUS_CONDITIONS and BOOT_ACK.

18.6.2.1.1 Example usage with ROM ISP command

Here uses the 8bit DDR mode as an example, and boot image is written to user data area. After writing the boot image, user want boot ROM to boot the image via fast boot to decrease the boot time. Fast boot also uses the same mode – 8bit DDR mode. ACK is enabled for fast boot. So the hex of the eMMC configuration block is 0xC0721625, 0x00000000

- Write the configuration block to MCU internal RAM.

```
blhost -u (PID,VID) - fill-memory 0x1c000 0x4 0xC0721625
blhost -u (PID,VID) - fill-memory 0x1c004 0x4 0x00000000
```

RAM address 0x1C000 is selected as an example. User can select any RAM position which are available to use.

- Execute the initialization progress using configure-memory command.

```
blhost -u
(PID,VID) - configure-memory 0x121 0x1c000
```

0x121 is the memory ID of eMMC card device. If the eMMC card is initialized successfully, then a “Success” will be received.

- After step 2 , eMMC is available to access. User can use get property 25 command to check the eMMC card capacity.

```
blhost -u (PID,VID) - get-property 25 0x121
```

- To program the boot image, user shall erase the eMMC card memory before program the image.

```
blhost -u (PID,VID) - flash-erase-region 0x0 0x2000 0x121
blhost -u (PID,VID)
- write-memory 0x400 C:\Image\bootImage.bin 0x121
```

The address of eMMC memory in the command line is byte address, not sector address. That means 8K bytes starting from the start address of eMMC memory will be erased, then the boot image C:\Image\bootImage.bin will be written to eMMC 1st Block.

- To check if the boot image is programmed successfully, user can read the data out.

```
blhost -u (PID,VID) - read-memory 0x200 0x2000 0x121
```

In most cases, user won't need to read the data out to verify if the boot image is written successfully or not, ROM will guarantee it when user gets a "Success" status for write-memory command. If user want to switch to other partitions of the eMMC device, user has to re-configure the eMMC devices two times.

- Select the Boot partition 1, bus width and speed timing are kept unchanged. Fast boot configuration is not necessary if user does not want to update it.

```
blhost -u
(PID,VID) - fill-memory 0x1c000 0x4 0xC1001600
blhost -u (PID,VID) -
configure-memory 0x121 0x1c000
blhost -u (PID,VID) - flash-erase-region 0x0 0x1000
0x121
blhost -u (PID,VID) - write-memory 0x400 C:\Image\bootPartitionOneImage.bin
0x121
```

18.6.3 1-bit Serial NOR flash through SPI

The boot ROM supports programming 1-bit SPI NOR FLASH devices (which supports a 3-byte address read 0x03 or 4-byte address read 0x13) via the Flash Configuration Option Block. The SPI clock frequency is set to 48Mhz by the ROM. The boot ROM supports either a manually configured option or an auto-detected option. When using the manually configured option, you must specify all the Flash information (Flash size, sector size, page size) in the option block. When using the auto-detected option, (which is only supported on devices that are JESD216 compliant) the boot ROM is able to detect the Flash information via the read SFDP (0x5A) command, where you can set all the Flash information to 0x0s. Below table shows the details required for configuring SPI NOR FLASH using either of these methods. The read Page (0x03) is used for the default read command, if the FLASH size is greater than 16MB (detected by the read SFDP command), the 0x13 command is used for the page read. If the SFDP command is not supported, ROM uses the read MID (0x9F) to detect whether there is a connected device to the boot ROM.

18.6.3.1 SPI 1 bit serial NOR boot

For 1-bit SPI NOR flash boot, it is mostly the same as FlexSPI Load-To-RAM boot. When select the SPI NOR boot, ROM tried to get the boot image from the NOR flash device connected to the SPI interface. ROM Use the primary 1-bit mode to access the NOR flash device via the SPI module. ROM copies the boot image to SRAM and then jumps to SRAM to boot the image. The boot action is the same as the 4.3.2-Recovery boot mode. The difference is that ROM can directly select the SPI 1-bit boot via the boot config fuse BOOT_CFG [0] bit [3:0] setting to b'0111. For more details, please refer to Table 12. Primary Boot Source based on PRIMARY_BOOT_SRC bits in BOOT_CFG [0]

Set the boot ROM in ISP mode, and the user can use below blhost commands to program the image into the serial NOR flash memory connected to the SPI interface.

```
blhost -p (COM XX) - fill-memory 0x1c000 0x04 0xc010000
blhost -p (COM XX) - configure-memory 0x110 0x1c000
blhost -p (COM XX) - flash-erase-region 0x0 0x10000 0x110
blhost -p (COM XX) - write-memory 0x1000 image.bin 0x110
```

The SPI instance used is chosen by fuse word 0x60, BOOT_CFG [0] bit17 to bit 19, more details, please refer fuse map. After the boot image programmed into the SPI NOR flash memory. Set The device boot mode to SPI NOR Boot, power down and up the board, After that, the device boot up from 1-bit NOR Flash via SPI.

18.6.3.2 1-bit SPI NOR Configuration Option Block

Table 96. 1-bit SPI NOR Configuration Option Block

Field	Description							
Option	TAG[31:28]	Reserved[27:24]	SPI index[23:20]	Reserved[19:16]	Memory Type[15:12]	Memory size[11:8]	Sector size[7:4]	Page size[3:0]
	0xc		SPI interface index: SPI0-SPI7		Memory configure type: 0 - Configure manually based on configure option block 2 - AutoConfigure the Nor flash via SFDP info	Memory size: 0 - 512KB 1 - 1MB 2 - 2MB 3 - 4MB 4 - 8MB 5 - 16MB 6 - 32MB 7 - 64MB 8 - 128MB 9 - 258MB 10 - 512MB 11 - 1GB 12 - 32KB 13 - 64KB 14 - 128KB 15 - 256KB	Sector size: 0 - 4KB 1 - 8KB 2 - 32KB 3 - 64KB 4 - 128KB 5 - 256KB	Page Size: 0 - 256B 1 - 512B 2 - 1MB 3 - 32B 4 - 64B 5 - 128B

- Tag - Fixed as 0xc
- SPI index - SPI interface used to access serial NOR flash
- Memory Type - Memory type used to configure and access the NOR flash
- Memory size - Memory capacity of the NOR flash device
- Sector size - Sector size of the NOR flash device
- Page size - Page size of the NOR flash device

Program 1-bit serial NOR flash device via SPI NOR Configuration Option Block:

The boot ROM supports programing the serial NOR flash device via SPI interface using the SPI NOR configuration option block. Below are the example.

```
blhost -u (PID,VID) - fill-memory 0x1c000 0x04 0xc010000 (write option block to SRAM address 0x1c000, the spi instance is 1)
```

```
blhost -u (PID,VID) - configure-memory 0x110 0x1c000 (configure serial NOR FLASH using option block)
```

```
blhost -u (PID,VID) - flash-erase-region (addr) (size) 0x110 (erase the serial NOR flash deice from addr of size)
```

NOTE

Flash Block erase is not supported for FlexSPI memory, when flash-erase-region command is issued from blhost.

18.7 RT500 ISP and IAP

All RT500 devices include In-System Programming (ISP) functions to support serial interface booting (UART, SPI) from an application processor download and USB HID support. In- Application Programming (IAP) calls are also available.

18.7.1 ISP, IAP Features

In-System Programming Supports:

- Supports UART, SPI and USB peripheral interfaces
- Automatic detection of the active peripheral
- Autobaud on UART peripheral
- Common packet-based protocol for all peripherals
- Packet error detection and retransmit
- Protection of RAM used by the boot ROM while it is running
- Command to read the properties of the device, such as FLASH and RAM size
- Programming OTP
- Programming Serial NOR Flash
- Programming eMMC device
- Key-provisioning command

In-Application Programming Supports:

- ROM APIs to program OTP
- ROM APIs to program Serial NOR FLASH
- ROM APIs to program encrypted boot image

18.7.2 Functional Description

18.7.2.1 Boot ROM

The internal ROM memory is used to store the boot code. After a reset, the ARM processor starts its code execution from this memory. The boot ROM code is executed every time the part is powered-on or is reset.

The boot ROM provides flash programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond. Host-side command line and GUI tools are available to communicate with the boot ROM. Users can utilize host tools to upload/download application code and do manufacturing via the boot ROM.

18.7.2.2 In-System Programming

In-System Programming and other related functions are supported in several different ways:

- For details of the ISP protocol, see [In-System Programming Protocol](#)
- For details of the ISP packet, see [Boot ROM packet types](#)
- For details of the ISP command set, see [The boot ROM Command Set](#)
- For details of UART In-System Programming, see [UART ISP commands](#)
- For details of SPI In-System Programming, see [SPI In-System Programming](#)

- For details of USB-HID In-System Programming, see [USB In-System Programming](#)

18.7.2.3 Memory map after any reset

The ROM boot ROM is located in the memory region starting from the address 0x1300_0000. Both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described in [ISP interrupts and SRAM use](#)

Based on specific combination of the state of ISP pins and the value of PRIMARY_BOOT_SRC OTP field, the ROM boot ROM will enter ISP mode and auto-detect activity on the UART/SPI or USB-HID interface. The auto-detect looks for activity on the UART, SPI and USB-HID interfaces and selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. UART, SPI and USB-HID ISP commutations are described in [In-System Programming Protocol](#) and [Boot ROM packet types](#).

18.7.2.4 ISP interrupts and SRAM use

18.7.2.4.1 ISP interrupts

The SysTick Interrupt is always enabled when the boot ROM is running, depending on the enabled peripherals, below interrupts may be enabled:

- GPIO interrupt is enabled during auto-baud detection if the UART is allowed in ISP mode
- UART interrupt is enabled when the UART is detected as active peripheral.
- USB interrupt is enabled when the USB-HID is allowed in ISP mode.
- Only one of above interrupts is enabled when the active peripheral is detected.
- SPI interrupt is enabled when the SPI is detected as active peripheral.

18.7.2.4.2 SRAM used by the ISP

Below regions are reserved for boot ROM use when the boot ROM is running. The heap and the BSS, RW sections need to be reserved for the ROM API use before calling the ROM APIs in user application (IAP calls). SRAM reserved regions are:

- Region0: 0x10010000-0x1001BFFF (48 KB)
- Region1: 0x10000-0x1BFFF (48 KB)
- Region2: 0x20010000-0x2001BFFF (48 KB)
- Region3: 0x30010000-0x3001BFFF (48 KB)

18.7.2.5 In-System Programming Protocol

This section explains the general protocol for the packet transfers between the host and the boot ROM. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

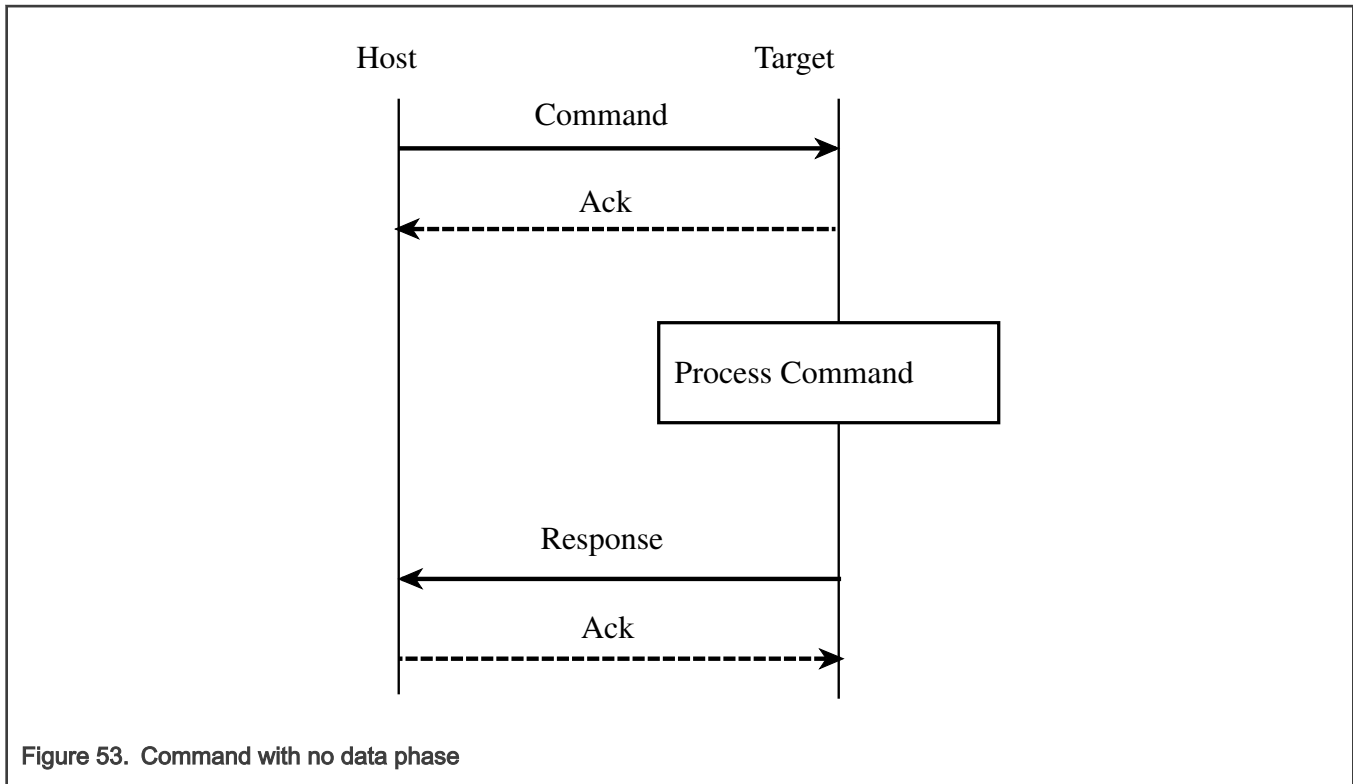
Commands may include an optional data phase.

- If the data phase is incoming (from the host to the boot ROM), it is part of the original command.
- If the data phase is outgoing (from the boot ROM to host), it is part of the response command.

18.7.2.5.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from the host)
- Generic response command packet (to host)

**NOTE**

In these diagrams, the Ack sent in response to a Command or a data packet can arrive at any time before, during, or after the Command/Data packet has processed.

18.7.2.5.2 Command with the incoming data phase

The protocol for a command with incoming data phase contains:

- Command packet (from host) (kCommandFlag_HasDataPhase set)
- Generic response command packet (to host)
- Incoming data packets (from the host)
- Generic response command packet

NOTE

- The host may not send any further packets while it is waiting for the response to a command.
- The data phase is aborted if the Generic Response packet prior to the start of the Data phase does not have a status of kStatus_Success.
- Data phases may be aborted by the receiving side by sending the final GenericResponse early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

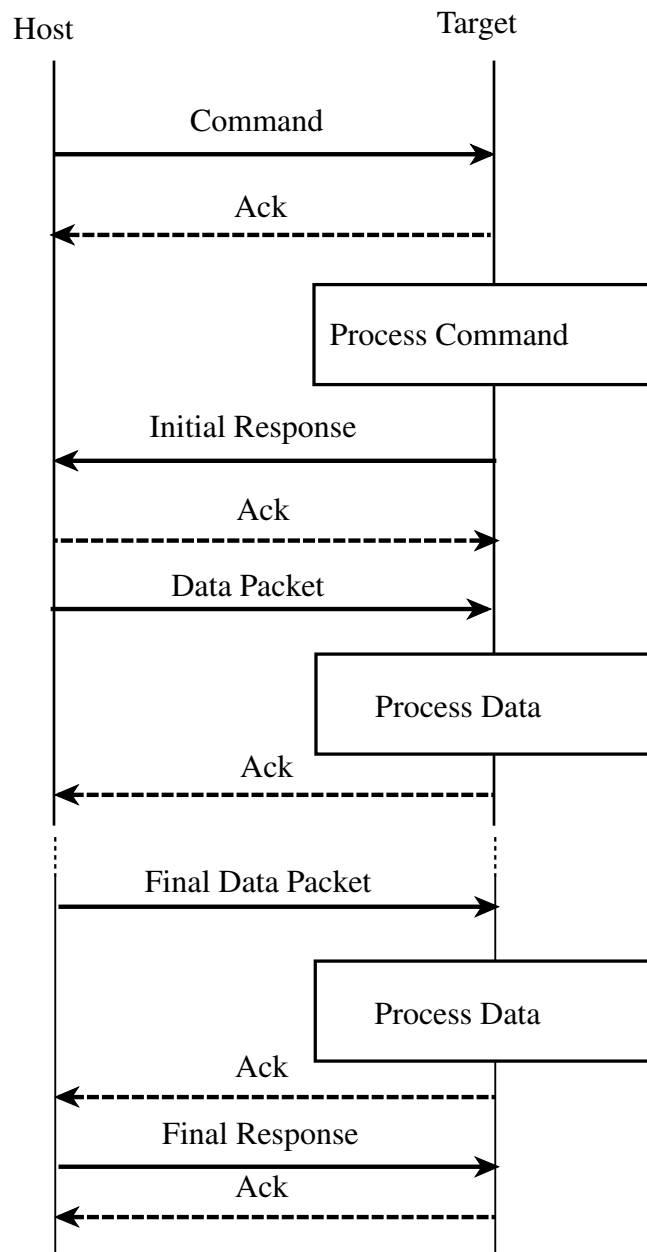


Figure 54. Command with incoming data phase

18.7.2.5.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

- Command packet (from the host)
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set)
- Outgoing data packets (to host)
- Generic response command packet (to host)

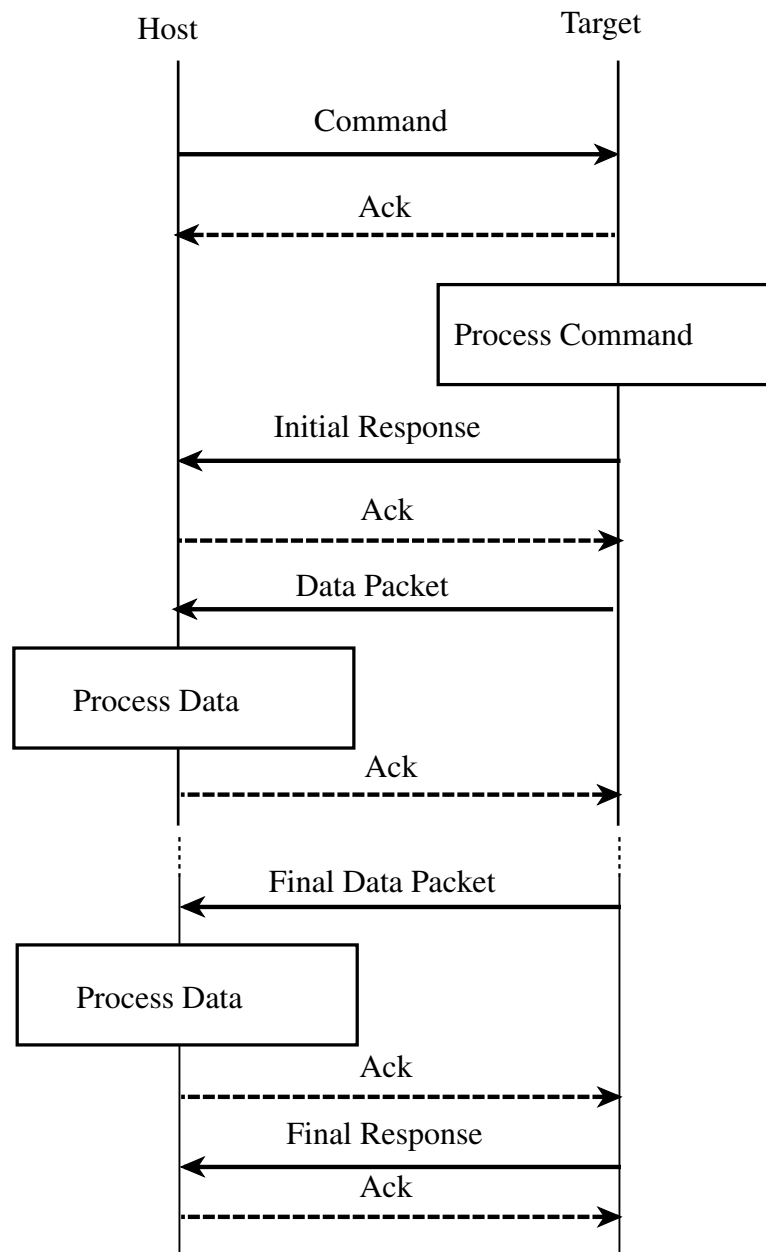


Figure 55. Command with outgoing data phase

NOTE

- The data phase is considered part of the response command for the outgoing data phase sequence.
- The host may not send any further packets while the host is waiting for the response to a command.
- The data phase is aborted if the ReadMemory Response command packet, prior to the start of the data phase, does not contain the `kCommandFlag_HasDataPhase` flag.
- Data phases may be aborted by the host sending the final Generic Response early with a status of `kStatus_AbortDataPhase`. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

18.7.2.6 Boot ROM packet types

The boot ROM device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host. The boot ROM device is the target, which receives a command or data packet. All data communication between host and target is packetized.

There are 6 types of packets used:

- Ping packet
- Ping Response Packet
- Framing Packet
- Command Packet
- Data Packet
- Response Packet

All fields in the packets are in little-endian byte order.

18.7.2.6.1 Ping Packet

The Ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral in order to run autobaud detection. The Ping packet can be sent from host to target at any time that the target is expecting a command packet. If the selected peripheral is UART, a ping packet must be sent before any other communications. For other serial peripherals, it is optional.

In response to a Ping packet, the target sends a Ping Response packet, discussed in later sections.

Table 97. Ping Packet Format

Byte #	Value	Name
0	0x5A	start byte
1	0xA6	ping

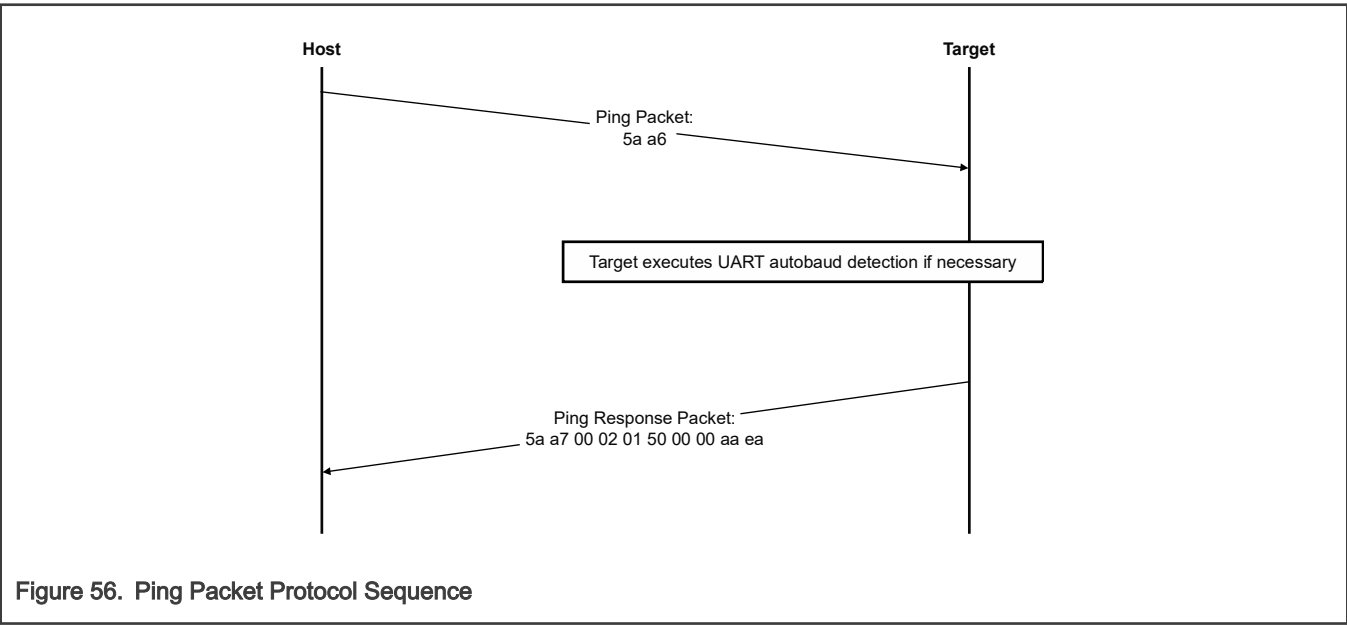


Figure 56. Ping Packet Protocol Sequence

18.7.2.6.2 Ping response packet

The target sends a Ping Response packet back to the host after receiving a Ping packet. If communication is over a UART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping Response packet. Once the Ping Response packet is received by the host, the connection is established, and the host starts sending commands to the target.

Table 98. Ping Response packet format

Byte #	Value	Parameter
0	0x5A	Start byte
1	0xA7	Ping response code
2		Protocol bugfix
3		Protocol minor
4		Protocol major
5		Protocol name = 'P' (0x50)
6		Options low
7		Options high
8		CRC16 low
9		CRC16 high

The Ping Response packet can be sent from host to target any time the target expects a command packet. For the UART peripheral, it must be sent by the host when a connection is first established, in order to run outbound. For other serial peripherals, it is optional but recommended to determine the serial protocol version. The version number is in the same format as the boot ROM version number returned by the GetProperty command.

18.7.2.6.3 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such features built-in. The framing packet structure sits between the link layer and the command layer. It wraps command and data packets as well.

Every framing packet containing data sent in one direction results in a synchronizing response framing packet in the opposite direction.

The framing packet described in this section is used for serial peripherals including the UART, I2C, and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used.

Table 99. Framing Packet Format

Byte #	Value	Parameter	
0	0x5A	Start byte	
1		packetType	
2		length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		length_high	
4		crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or
5		crc16_high	

Table continues on the next page...

Table 99. Framing Packet Format (continued)

Byte #	Value	Parameter	
			data packets, but does not include the CRC bytes.
6 . . . n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 100. Special Framing Packet Format

Byte#	Value	Parameter
0	0x5A	Start byte
1	0xA _n	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 101. packetType Field

packetType	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
0xA7	kFramingPacketType_PingResponse	A response to Ping; contains the framing protocol version number and options.

18.7.2.6.4 CRC16 algorithm

This section provides the CRC16 algorithm.

The CRC is computed over each byte in the framing packet header, excluding the crc16 field itself, plus all of the payload bytes. The CRC algorithm is the XMODEM variant of CRC-16.

The characteristics of the XMODEM variant are:

Width	16
Polynomial	0x1021

Table continues on the next page...

Table continued from the previous page...

Init value	0x0000
Reflect in	FALSE
Reflect out	FALSE
xor out	0x0000
Check result	0x31c3

The check result is computed by running the ASCII character sequence "123456789" through the algorithm.

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
        uint32_t byte = src[j];
        crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
            for (i = 0; i < 8; ++i)
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
            }
            crc = temp;
        }
    }
    return crc;
}
```

18.7.2.6.5 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 102. Command Packet Format

Command Packet Format (32 bytes)	
Command Header (4 bytes)	28 bytes for Parameters (Max 7 parameters)

Table continues on the next page...

Table 102. Command Packet Format (continued)

Command Packet Format (32 bytes)										
Tag	Flags	Rsvd	Param Count	Param1 (32-bit)	Param2 (32-bit)	Param3 (32-bit)	Param4 (32-bit)	Param5 (32-bit)	Param6 (32-bit)	Param7 (32-bit)
Byte 0	Byte 1	Byte 2	Byte 3	-	-	-	-	-	-	-

Table 103. Command Header Format

Byte #	Command Header Field	
0	Command or Response tag	The command header is 4 bytes long, with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only 7 parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 104. Command Tags

Command Tag	Name	
0x01	FlashEraseAll	The command tag specifies one of the commands supported by the boot ROM. The valid command tags for the boot ROM are listed here.
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x06	Reserved	
0x07	GetProperty	
0x08	ReceiveSbFile	
0x09	Execute	
0x0A	Call	
0x0B	Reset	
0x0C	SetProperty	
0x0D	Reserved	
0x0E	eFuseProgram	
0x0F	eFuseRead	
0x10	Reserved	
0x11	ConfigureMemory	

Table continues on the next page...

Table 104. Command Tags (continued)

Command Tag	Name	
0x12	Reserved	
0x13	Reserved	
0x14	Reserved	
0x15	KeyProvision	

Table 105. Response Tags

Response Tag	Name	
0xA0	GenericResponse	The response tag specifies one of the responses the boot ROM (target) returns to the host. The valid response tags are listed here.
0xA3	ReadMemoryResponse	
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)	
0xAF	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)	
0xB5	KeyProvisionResponse	

Flags: Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets follow the command sequence. The number of bytes that are transferred in the data phase is determined by a command specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to 7 parameters.

18.7.2.6.6 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- KeyProvisionResponse

GenericResponse: After the boot ROM has processed a command, the boot ROM sends a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 106. GenericResponse Parameters

Byte #	Parameter	Description
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target. If a command succeeds, then a kStatus_Success code is returned.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 107. GetPropertyResponse Parameters

Byte#	Value	Parameter
0-3		Status code
4-7		Property value
...		...
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to 2 for the status code and the data byte count parameters shown below.

Table 108. ReadMemoryResponse Parameters

Byte#	Parameter	Description
0-3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to 2 plus the number of words requested to be read in the FlashReadOnceCommand.

Table 109. FlashReadOnceResponse Parameters

Byte#	Value	Parameter
0-3		Status code
4-7		Byte count to read
...		...
		Can be up to 20 bytes of requested read data.

The KeyProvisionResponse packet is sent by the target in response to the host sending a KeyProvision command. The KeyProvisionResponse packet contains the framing packet data and command packet data, with the command/response tag set to a KeyProvisionResponse tag value (0xB5), and the flags field set to kCommandFlag_HasDataPhase (1).

Table 110. KeyProvisionResponse Parameters

Byte#	Value	Parameter
0-3		Status code
4-7		Data byte count

18.7.2.6.7 Host Send/Receive packet timeout

For the below packet types, each packet is bind with a timeout timer during the transfer from Host to ROM slave device(I2C, UART, SPI interface). For each packet, the Sent timeout time is the packet length(in byte) multiply by 10ms. The receive timeout timer is the packet length (in byte) multiply 20ms. That means the Host should send the packet before the sending packet timeout and receive the data from slave before the receiving packet out.

For example, for Host sending, if the packet length is 10 Bytes, the Host should send the Packet in $10 \times 10\text{ms} = 100\text{ms}$. If not, the slave device ROM regard it as a timeout.

For Host receiving, if the data packet is 10 Bytes, if the Host starts to receive, the host should finish receiving all data bytes in $10 \times 20\text{ms} = 200\text{ms}$, or the slave device ROM regards it as a timeout.

- Ping packet
- Ping Response packet
- Framing packet
- Command packet
- Data packet
- Response packet

18.7.2.6.8 Host read ACK(5a,a1) packet timing restriction

If not use the IRQ pin for the In-System Programming protocol, the host needs to follow the below data fetching timing for ACK from the device.

1. After sending out the command, the host needs to wait for at least 100us before polling the start byte 0x5A of the ACK packet sent out by the device; If fetching data is not 0x5A, delay 100us, and do polling to reread 0x5A.
2. For the ACK packet 0x5A, 0xA1, after the host receives the 0x5A, a 50us delay must add before fetching the 0xA1 from the device.

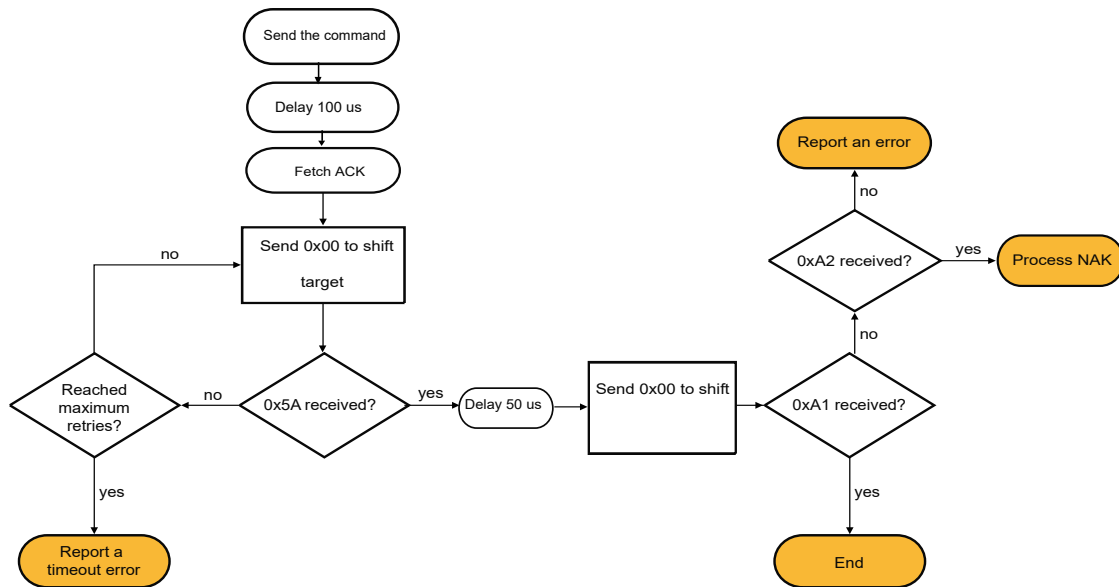


Figure 57. Host read ACK(5a,a1) packet timing restriction

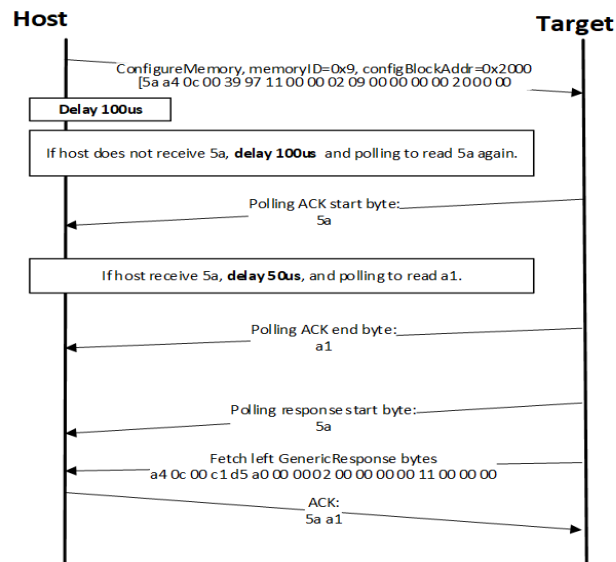


Figure 58. Host target ACK(5a,a1) packet timing restriction

18.7.2.7 The boot ROM Command Set

18.7.2.7.1 Introduction

All boot ROM commands follow the command packet format wrapped by the framing packet as explained in previous sections. See [Table 104](#) for a list of commands supported by the boot ROM.

18.7.2.7.2 GetProperty command

The GetProperty command is used to query the boot ROM about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

The 32-bit property tag is the only parameter required for GetProperty command.

Byte#	Command
0-3	Property tag
4-7	External Memory Identifier (only applies to get property for external memory)

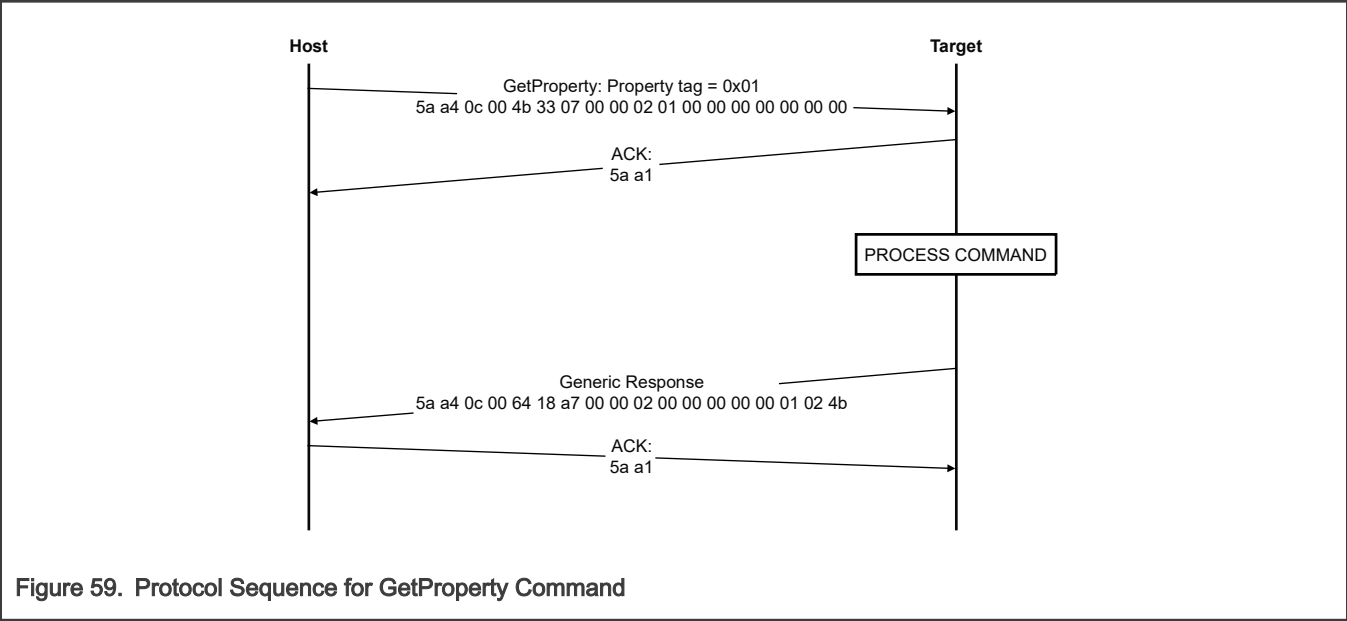


Figure 59. Protocol Sequence for GetProperty Command

Table 111. GetProperty Command Packet Format (Example)

GetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x4B 0x33
Command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	propertyTag	0x00000001 - CurrentVersion

Table continues on the next page...

Table 111. GetProperty Command Packet Format (Example) (continued)

GetProperty	Parameter	Value
	Memory ID	0x00000000 - Internal Flash (0x00000001 - FLEXSPI0 Memory)

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). The next table shows an example of a GetPropertyResponse packet.

Table 112. GetProperty Response Packet Format (Example)

GetPropertyResponse	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0c 0x00 (12 bytes)
	crc16	0x07 0x7a
Command packet	responseTag	0xA7
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	status	0x00000000
	propertyValue	0x0000014b - CurrentVersion

18.7.2.7.3 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options of the boot ROM. The command accepts the same property tags used with the GetProperty command. However, only some properties are writable--see Appendix B. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

The property tag and the new value to set are the two parameters required for the SetProperty command.

Table 113. Parameters for SetProperty Command

Byte#	Command
0-3	Property tag
4 - 7	Property value

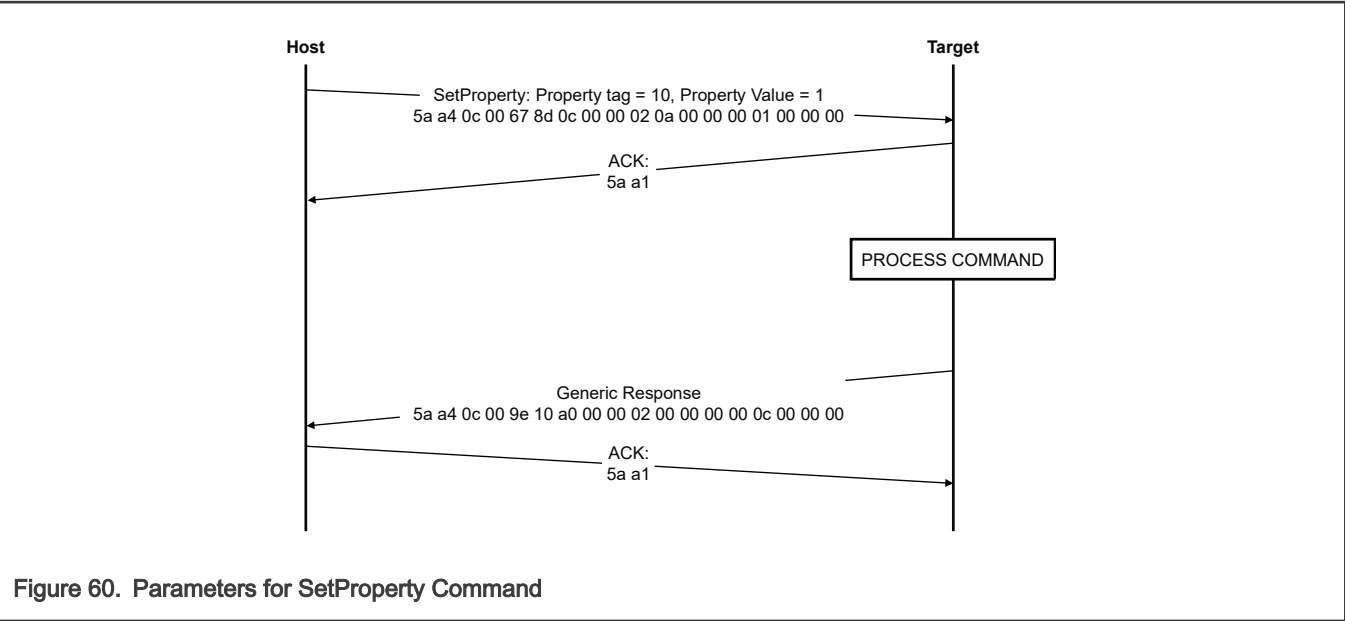


Table 114. SetProperty Command Packet Format (Example)

SetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x67 0x8D
Command packet	commandTag	0x0C – SetProperty with property tag 10
	flags	0x00
	Reserved	0x00
	parameterCount	0x02
	propertyTag	0x0000000A - VerifyWrites
	propertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target returns a GenericResponse packet with one of following status codes:

Table 115. SetProperty Response Status Codes

Status Code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

18.7.2.7.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command fails and returns an error status code. Executing the FlashEraseAll command releases flash security if it (flash security) was enabled, by setting the FTFA_FSEC register. However, the FSEC field of the flash configuration field is erased, so unless it is reprogrammed, the flash security is re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires memory ID. If memory ID is not specified, the internal flash (memory ID =0) will be selected as default.

Table 116. Parameter for FlashEraseAll Command

Byte #	Parameter	
0-3	Memory ID	
	0x000	Reserved
	0x010	Reserved
	0x001	Serial NOR through QuadSPI
	0x008	Reserved
	0x009	Serial NOR through FlexSPI
	0x100	Reserved
	0x101	Reserved
	0x110	Serial NOR/EEPROM through SPI
	0x121	eMMC through uSDHC

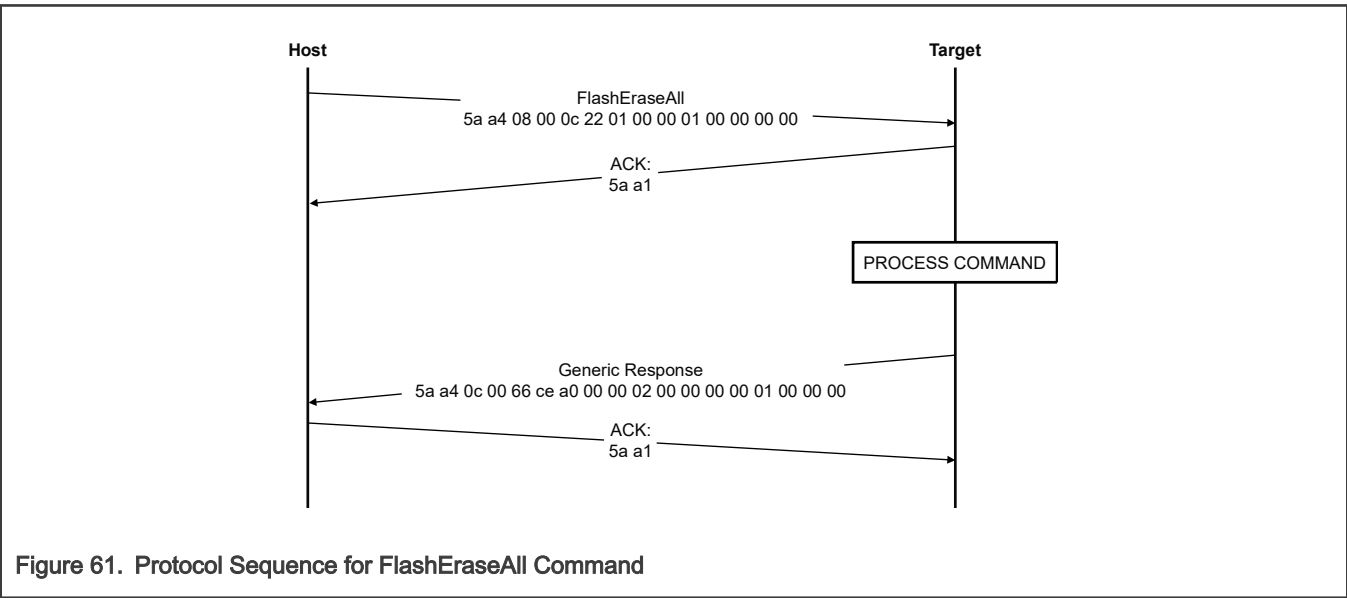


Figure 61. Protocol Sequence for FlashEraseAll Command

Table 117. FlashEraseAll Command Packet Format (Example)

FlashEraseAll	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x08 0x00
	crc16	0x0C 0x22
Command packet	commandTag	0x01 - FlashEraseAll
	flags	0x00
	reserved	0x00
	parameterCount	0x01
	Memory ID	refer to the above table

The FlashEraseAll command has no data phase.

Response: The target returns a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command or set to an appropriate error status code.

18.7.2.7.5 FlashEraseRegion command

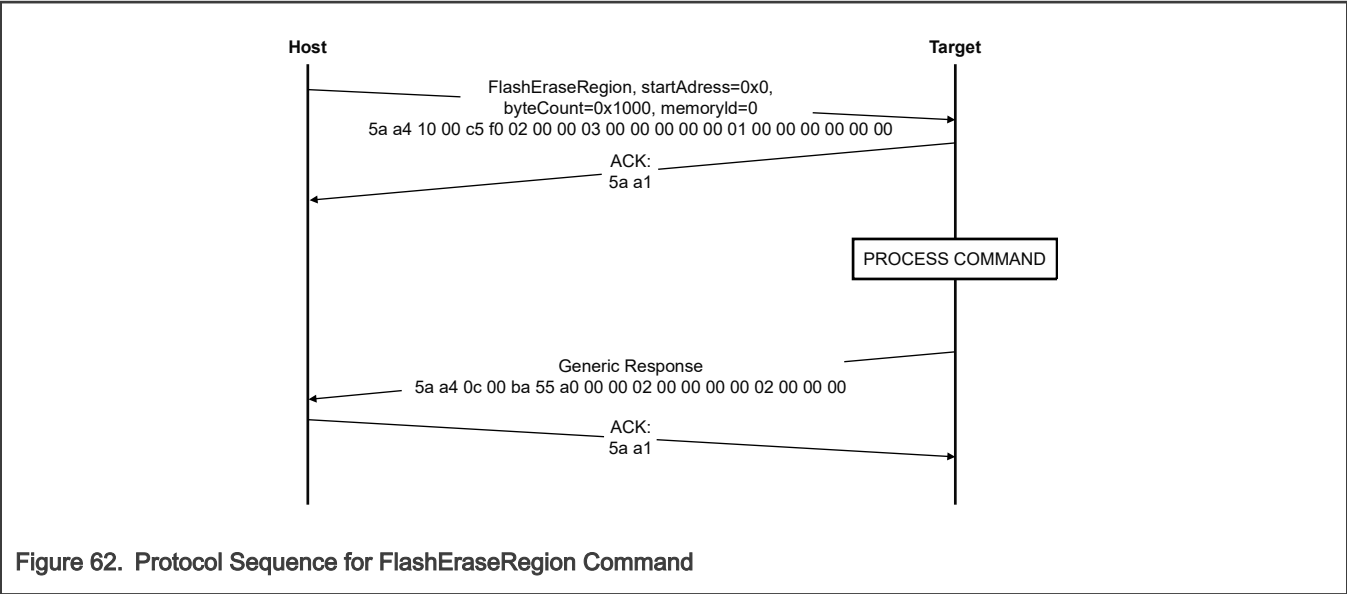
The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

The start address, and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command fails and returns kStatus_FlashAlignmentError (101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command fails and returns kStatus_FlashAddressError (102). If any part of the region specified is protected, the FlashEraseRegion command fails and returns kStatus_MemoryRangeInvalid (10200).

Table 118. Parameters for FlashEraseRegion Command

Byte#	Parameter
0-3	Start address
4-7	Byte count
8-11	Memory ID

The FlashEraseRegion command has no data phase.



Response: The target returns a GenericResponse packet with one of following error status codes.

Table 119. FlashEraseRegion Response Status Codes

Status Code
kStatus_Success (0)
kStatus_MemoryRangeInvalid (10200)
kStatus_FlashAlignmentError (101)
kStatus_FlashAddressError (102)
kStatus_FlashAccessError (103)
kStatus_FlashProtectionViolation (104)
kStatus_FlashCommandFailure (105)

18.7.2.7.6 ReadMemory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address, and number of bytes are the two parameters required for ReadMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 120. Parameters for read memory command

Byte	Parameter	Description
0-3	Start address	Start address of memory to read from
4-7	Byte count	Number of bytes to read and return to caller
8-11	Memory ID	Internal or external memory Identifier

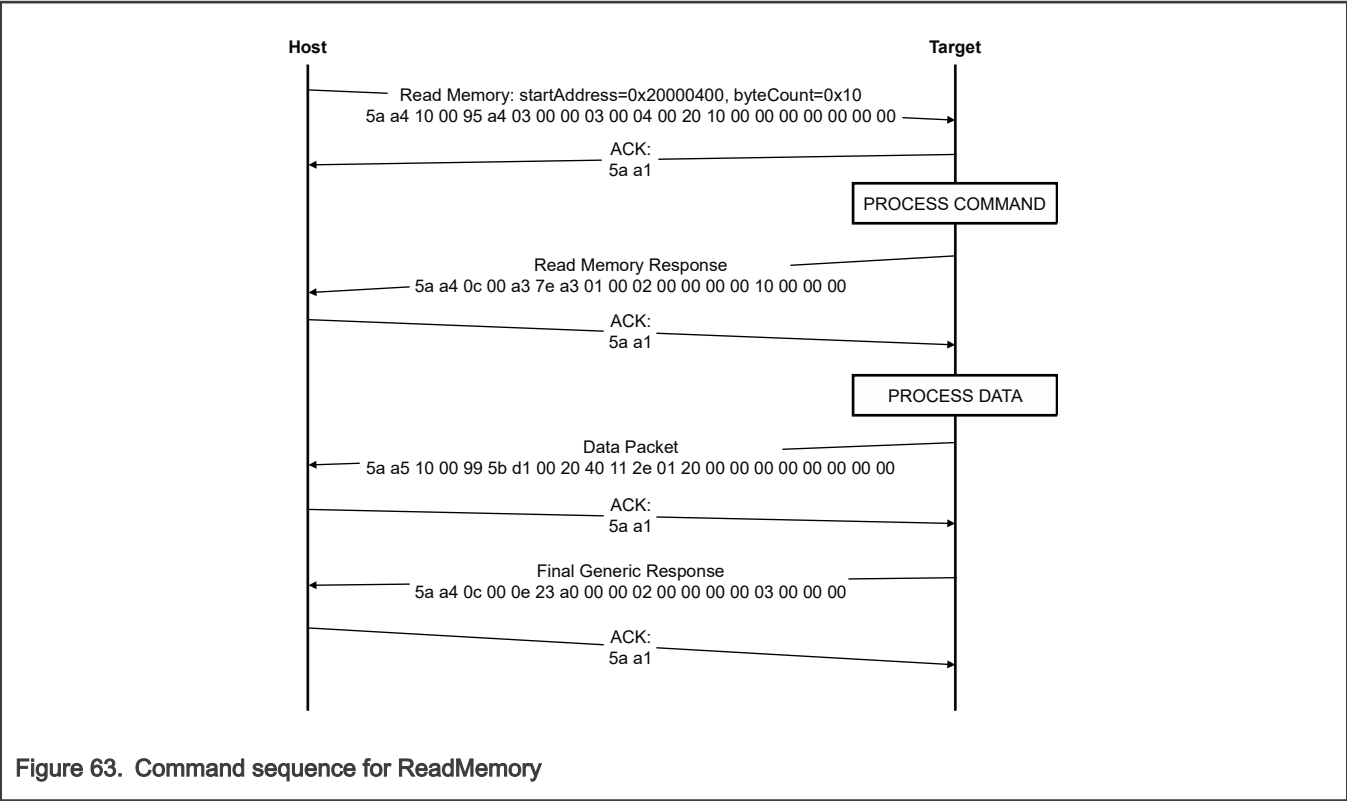


Table 121. ReadMemory Command Packet Format (Example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A0xA4,
	packetType	kFramingPacketType_Command
	length	0x10 0x00
	crc16	0xF4 0x1B
Command packet	commandTag	0x03 - readMemory
	flags	0x00
	Reserved	0x00
	parameterCount	0x03
	startAddress	0x20000400
	byteCount	0x00000064
	memoryID	0x0

Data Phase: The ReadMemory command has a data phase. Because the target works in slave mode, the host needs to pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error status code.

18.7.2.7.7 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion.
- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- The byte count is rounded up to a multiple of 4, and trailing bytes are filled with the flash erase pattern (0xff).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

The start address and number of bytes are the 2 parameters required for WriteMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 122. Parameters for WriteMemory Command

Byte#	Command
0-3	Start address
4-7	Byte count
8-11	Memory ID

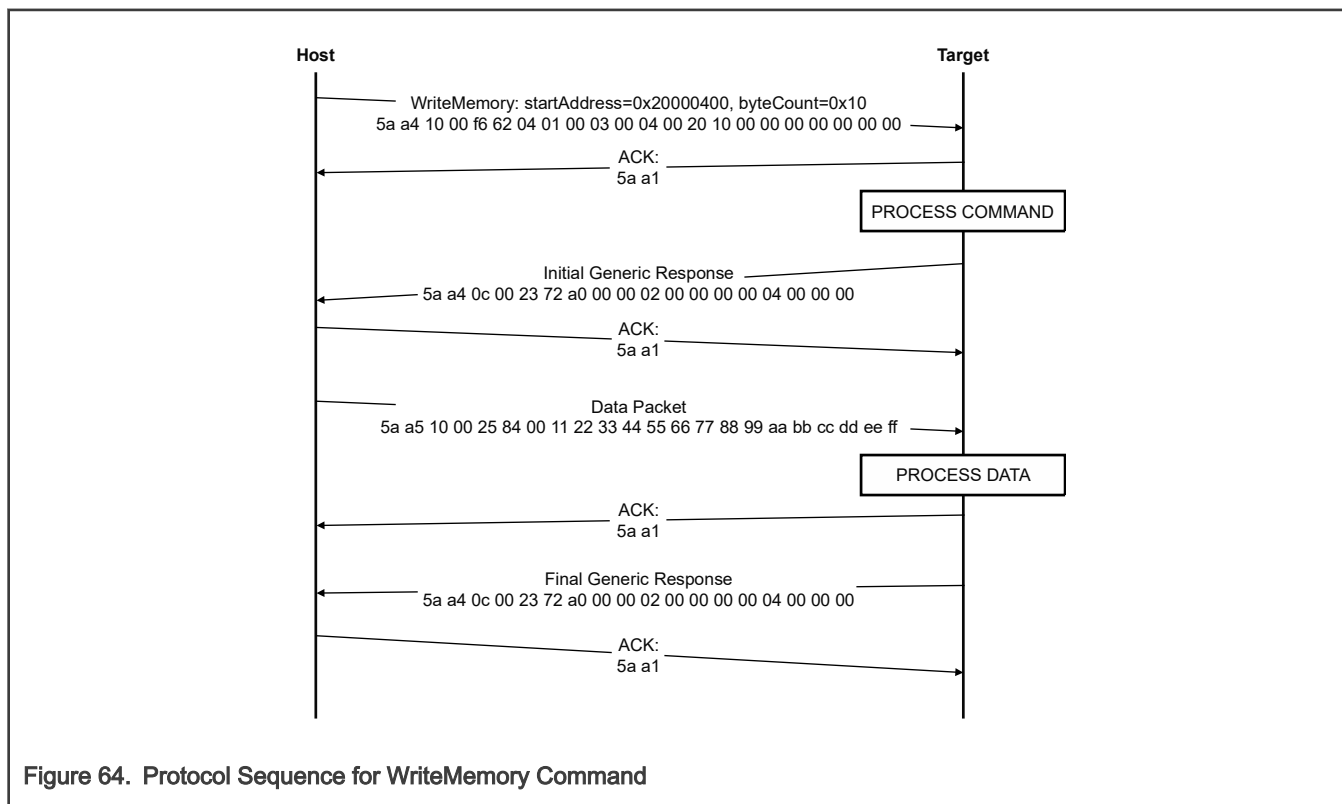


Figure 64. Protocol Sequence for WriteMemory Command

Table 123. WriteMemory Command Packet Format (Example)

WriteMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0x97 0xDD
Command packet	commandTag	0x04 - writeMemory
	flags	0x01
	Reserved	0x00
	parameterCount	0x03
	startAddress	0x20000400
	byteCount	0x00000064
	memoryID	0x0

Data Phase: The WriteMemory command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

18.7.2.7.8 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 124. Parameters for FillMemory Command

Byte#	Command
0-3	Start address of memory to fill
4-7	Number of bytes to write with the pattern - The start address should be 32-bit aligned. - The number of bytes must be evenly divisible by 4. (Note: for a part that uses FTFE flash, the start address should be 64-bit aligned, and the number of bytes must be evenly divisible by 8).
8-11	32-bit pattern

- To fill with a byte pattern (8-bit), the byte must be replicated 4 times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated 2 times in the 32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern is 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern is 0x5AFE5AFE.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, or FlashEraseRegion command.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

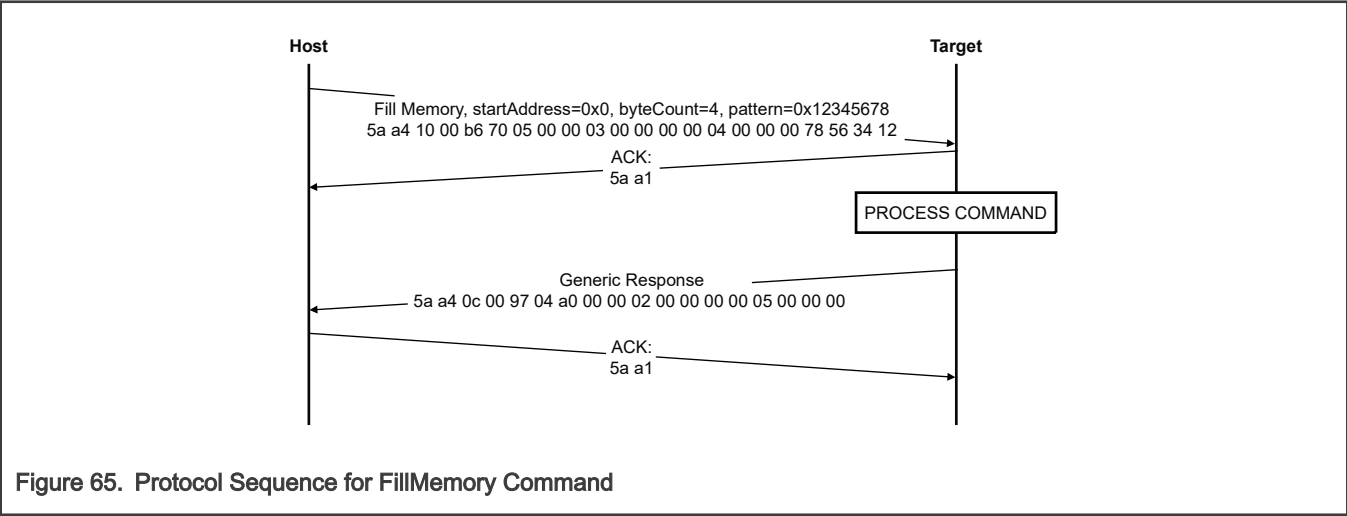


Table 125. FillMemory Command Packet Format (Example)

FillMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0xE4 0x57
Command packet	commandTag	0x05 – FillMemory
	flags	0x00
	Reserved	0x00
	parameterCount	0x03
	startAddress	0x00007000
	byteCount	0x00000800
	patternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

18.7.2.7.9 Execute command

The execute command results in the boot ROM setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command. If the stack pointer is set to zero, the called code is responsible for setting the processor stack pointer before using the stack.

Table 126. Parameters for Execute Command

Byte#	Command
0-3	Jump address
4-7	Argument word
8-11	Stack pointer address

The Execute command has no data phase.

Response: Before executing the Execute command, the target validates the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

18.7.2.7.10 Call command

The Call command executes a function that is written in memory at the address sent in the command. The address needs to be a valid memory location residing in accessible flash (internal or external) or in RAM. The command supports the passing of one 32-bit argument. Although the command supports a stack address, at this time the call still takes place using the current stack pointer. After execution of the function, a 32-bit return value is returned in the generic response message.

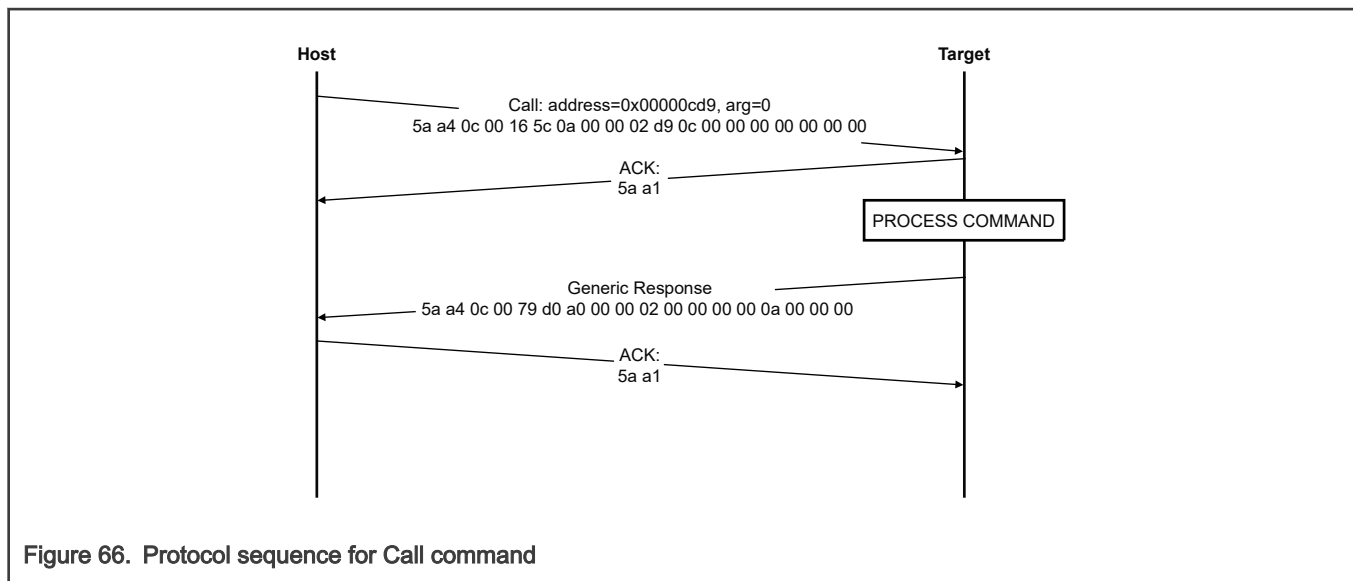


Table 127. Parameters for Call Command

Byte#	Command
0-3	Call address
4-7	Argument word
8-11	Stack pointer

Response: The target returns a GenericResponse packet with a status code either set to the return value of the function called or set to kStatus_InvalidArgument (105).

18.7.2.7.11 Reset command

The Reset command results in the boot ROM resetting the chip.

The Reset command requires no parameters.

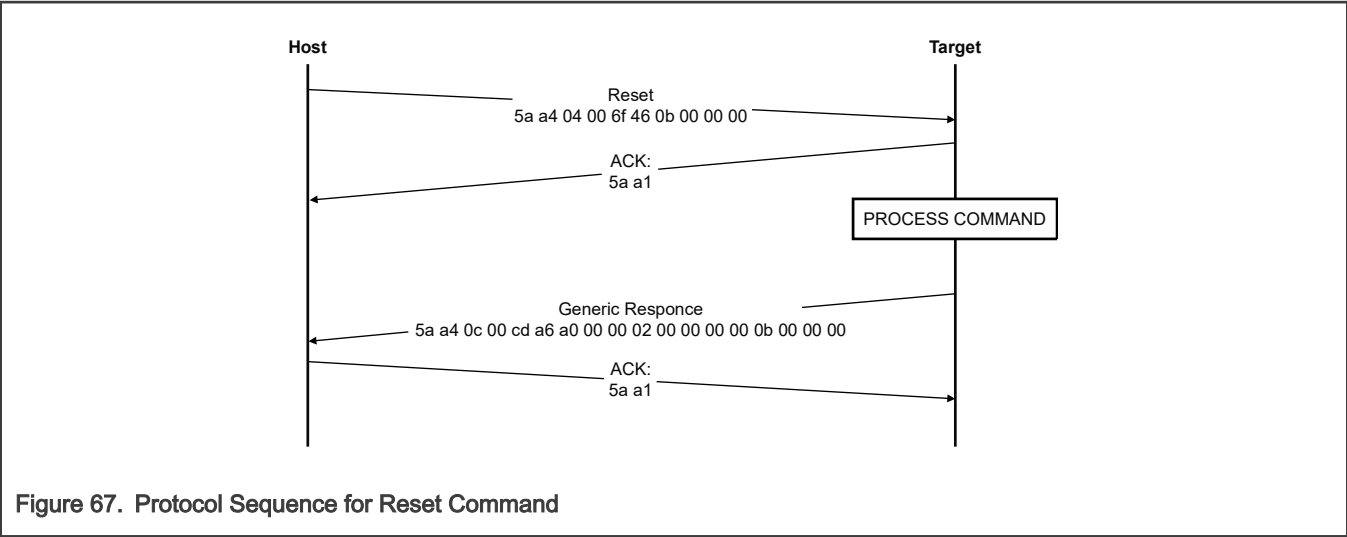


Table 128. Reset Command Packet Format (Example)

Reset	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0x6F 0x46
Command packet	commandTag	0x0B - reset
	flags	0x00
	Reserved	0x00
	parameterCount	0x00

The Reset command has no data phase.

Response: The target returns a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

The reset command can also be used to switch boot from flash after successful flash image provisioning via ROM boot ROM. After issuing the reset command, allow 5 seconds for the user application to start running from Flash.

18.7.2.7.12 eFuseProgramOnce/FlashProgramOnce command

The FlashProgramOnce command writes data (that is provided in a command packet) to a specified range of bytes in the program once field. Special care must be taken when writing to the program once field.

- The program once field only supports programming once, so any attempted to reprogram a program-once field gets an error response.
- Writing to the program once field requires the byte count to be 4-byte aligned or 8-byte aligned.

The FlashProgramOnce command uses three parameters: index 2, byteCount, data.

Table 129. Parameters for FlashProgramOnce Command

Byte#	Command
0-3	Index of program once field

Table continues on the next page...

Table 129. Parameters for FlashProgramOnce Command (continued)

Byte#	Command
4-7	Byte count (must be evenly divisible by 4)
8-11	Data

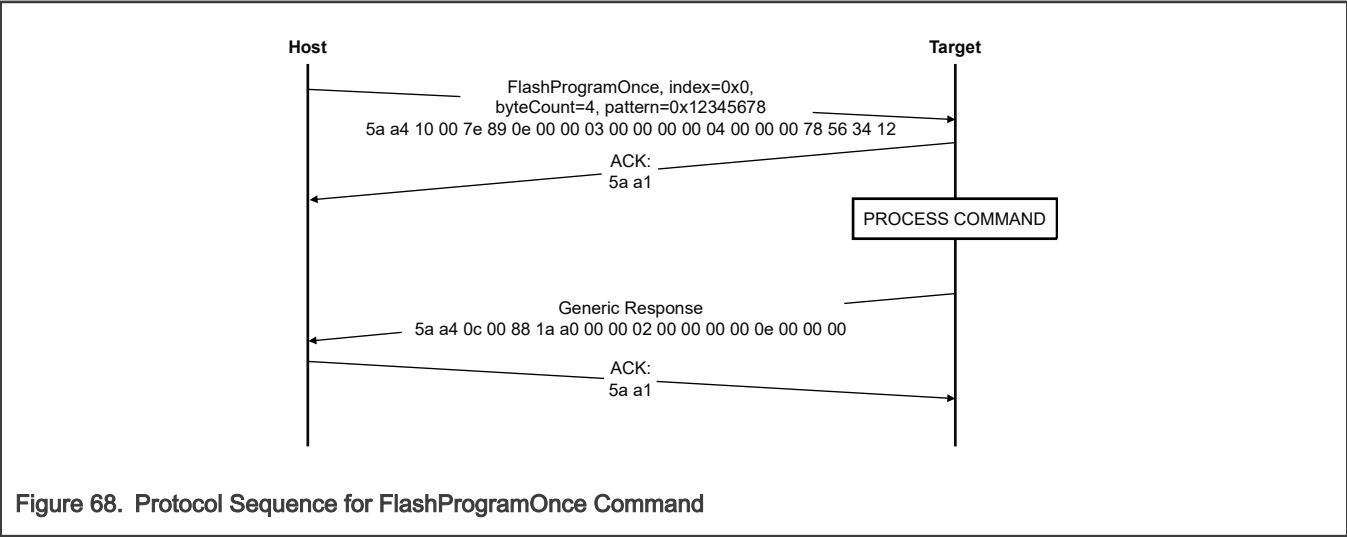


Figure 68. Protocol Sequence for FlashProgramOnce Command

Table 130. FlashProgramOnce Command Packet Format (Example)

FlashProgramOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0x7E4 0x89
Command packet	commandTag	0x0E – FlashProgramOnce
	flags	0
	Reserved	0
	parameterCount	3
	index	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

18.7.2.7.13 eFuseReadOnce / FlashReadOnce command

The FlashReadOnce command returns the contents of the program once field by given index and byte count. The FlashReadOnce command uses 2 parameters: index and byteCount.

Table 131. Parameters for FlashReadOnce Command

Byte #	Parameter	Description
0-3	Index	Index of the program once field (to read from)
4-7	byteCount	Number of bytes to read and return to the caller

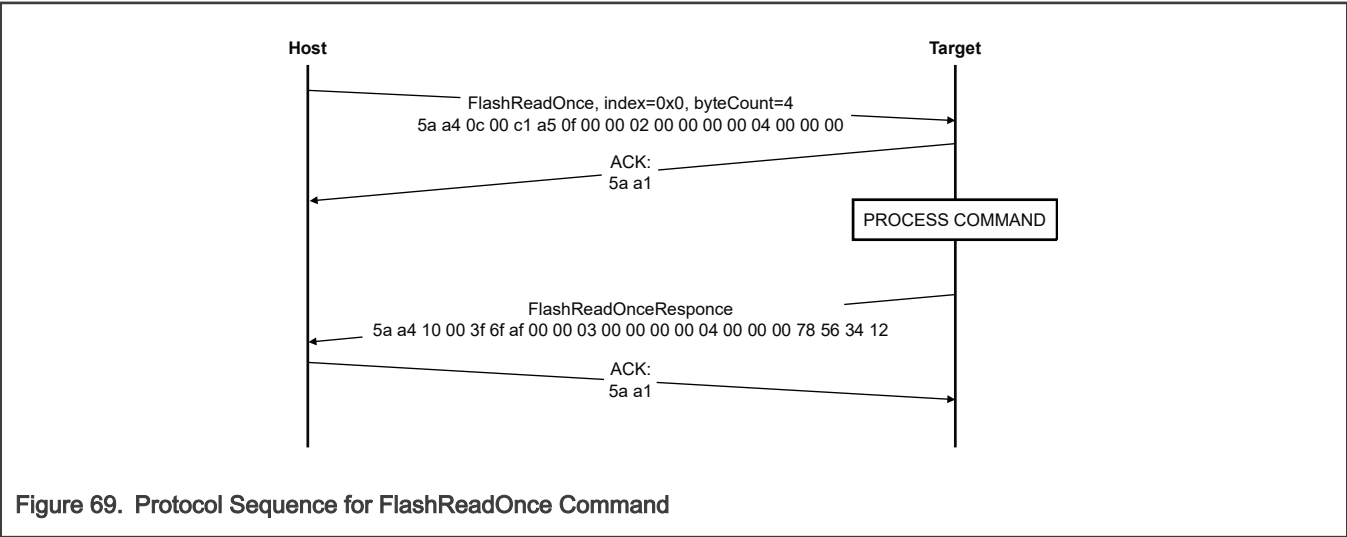


Figure 69. Protocol Sequence for FlashReadOnce Command

Table 132. FlashReadOnce Command Packet Format (Example)

FlashReadOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x0C 0x00
	Crc	0xC1 0xA5
Command packet	commandTag	0x0F – FlashReadOnce
	Flags	0x00
	Reserved	0x00
	parameterCount	0x02
	Index	0x0000_0000
	byteCount	0x0000_0004

Table 133. FlashReadOnce Response Format (Example)

FlashReadOnce Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x10 0x00

Table continues on the next page...

Table 133. FlashReadOnce Response Format (Example) (continued)

FlashReadOnce Response	Parameter	Value
Command packet	Crc	0x3F 0x6F
	commandTag	0xAF
	Flags	0x00
	Reserved	0x00
	parameterCount	0x03
	Status	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target returns a FlashReadOnceResponse packet with a status code set to kStatus_Success, a byte count and corresponding data read from Program Once Field upon successful execution of the command or returns with a status code set to an appropriate error status code and a byte count set to 0.

18.7.2.7.14 ConfigureMemory command

The ConfigureMemory command configures the internal/external memory device using a preprogrammed configuration block. The parameters passed in the command are the memory ID, and then the memory address from which the configuration data can be loaded from. Options for loading the data can be a scenario where the configuration data is written to a RAM or flash location and then this command directs the boot ROM to use the data at that location to configure the external memory devices.

Table 134. Parameters for Configure Memory Command

Byte#	Command
0-3	Memory ID
4-7	Configuration block address

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error code.

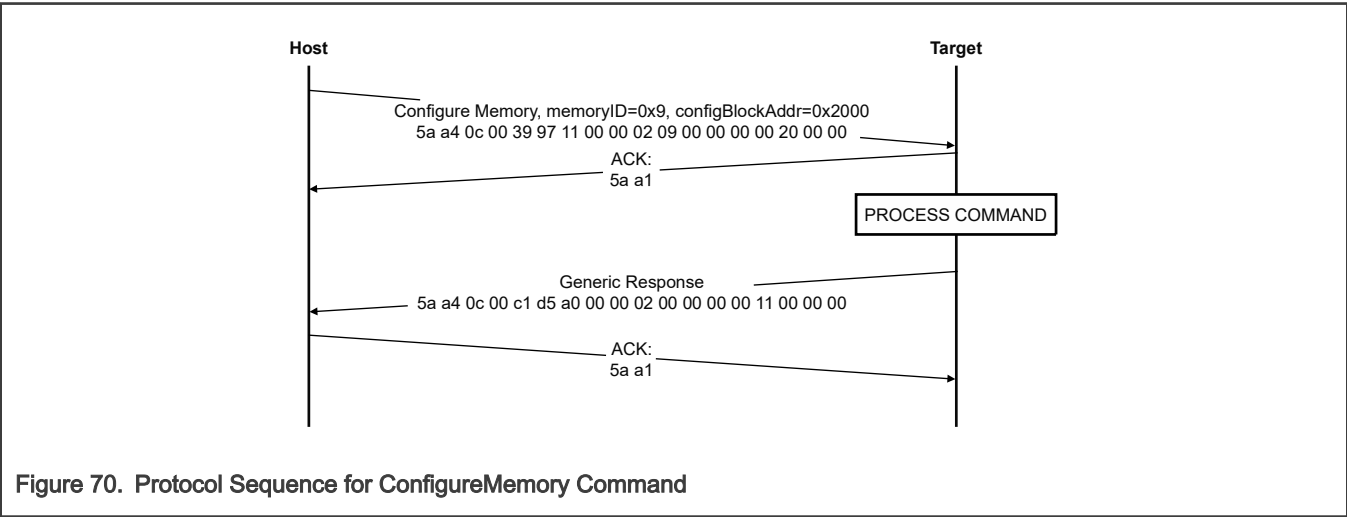


Figure 70. Protocol Sequence for ConfigureMemory Command

18.7.2.7.15 ReceiveSBFile command

The Receive SB File command (ReceiveSbFile) starts the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that is sent in the data phase. The SB file is processed as it is received by the boot ROM. See the Secure boot related sections for more details about the SB file.

Table 135. Parameters for Receive SB File Command

Byte#	Command
0-3	Byte count

Data Phase: The Receive SB file command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the Receive SB File command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to the kStatus_Success upon successful execution of the command or set to an appropriate error code.

18.7.2.7.16 KeyProvision command

The KeyProvision command is a pack of several security related commands, to install pre-shared keys, generate random keys and save them into the Protected Flash Region - Customer Key Store area.

There are 3 parameters for KeyProvision command, listed in the table below. The first parameter <Key Operation> is required to specific the KeyProvision command behavior. The other two parameters <Key Type> and <Key Size> are required for some KeyProvision operations.

Table 136. Parameters for KeyProvision Command

Byte#	Command
0-3	Key Operation
4-7	Key Type / Memory ID (optional for some Key Operations)
8-11	Key Size (optional for some Key Operations)

Table 137. KeyProvision Operation Details

Value	Operation	Details
0	Enroll	Key Provision device enrollment. Generates activation code, PUF key etc. <Key Type> and <Key Size> are not used for this operation.
1	SetUserKey	Send <Key size> bytes of the <Key Type> key to ROM from host. Incoming data Phase is required to transfer the key bytes.
2	SetIntrinsicKey	Generate <Key Size> bytes of the key specified by <Key Type> in key store
3	WriteNonVolatile	Write the key store in RAM to a nonvolatile memory specified by <Memory ID>.

Table continues on the next page...

Table 137. KeyProvision Operation Details (continued)

Value	Operation	Details
		<Key Size> is not used for this operation.
4	ReadNonVolatile	Load the key store to RAM from a nonvolatile memory specified by <Memory ID>. <Key Size> is not used.
5	WriteKeyStore	Send the key store to ROM from host. Incoming data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.
6	ReadKeyStore	Read the key store from ROM to host. Outgoing data Phase is required to transfer the key bytes. Data byte size is fixed as key store size. <Key Type> and <Key Size> are not used for this operation.

Table 138. Key Type Details

Value	Key Type
0x0	Invalid
0x1	HashCrypt SRK
0x2	OTFAD KEK
0x3	Firmware Update Key 0
0x4	Firmware Update Key 1
0x5	Firmware Update Key 2
0x6	Firmware Update Key 3
0x7	Encrypted Region Key 0
0x8	Encrypted Region Key 1
0x9	Encrypted Region Key 2
0xA	Encrypted Region Key 3
0xB	User Key
0xC	UDS

Command: KeyProvision command packet format is shown in below table.

Table 139. KeyProvision command packet format(example)

KeyProvision	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0x57, 0x32
Command packet	command tag	0x15
	flags	0x00 (no data phase, 0x01 for has data phase)
	reserved	0x00
	parameter count	0x03
	key operation	0x00000002
	key type / memory ID	0x00000000
	key size	0x00000100

Data Phase: It is determined by <Key Operation> if a KeyProvision command has Incoming or outgoing data phase or not.

For incoming packet, the host shall send data packets until the number of data bytes specified by <Key Size> or the key store size are received by the target.

For outgoing data phase, the host needs to pull data packets until the entire key store data bytes are received by host. The key store size will be send to host by KeyProvision response.

Response: The target returns a GenericResponse packet for the key operations without data phase, such as Enroll. And returns a KeyProvisionResponse packet for the other key operations, such as WriteKeyStore. The following table describes the KeyProvisionResponse packet.

Table 140. KeyProvision Response packet format (Example)

KeyProvisionResponse	Parameter	Value
Framing packet	start byte	0x5A
	packet type	0xA4, kFramingPacketType_Command
	length	0x10, 0x00
	crc16	0xXX, 0xXX
Command packet	command tag	0xB5
	flags	0x01 (has data phase)
	reserved	0x00
	parameter count	0x02
	status	0x00000000
	key size	0x00000100

18.7.2.7.17 Supported properties in GetProperty and SetProperty

Name	Writable	Tag Value	Size	Description
CurrentVersion	no	1	4	The current boot ROM version.
AvailablePeripherals	no	2	4	The set of peripherals supported on this chip.
FlashStartAddress	no	3	4	Start address of program flash.
FlashSizeInBytes	no	4	4	Program flash size in bytes.
FlashSectorSize	no	5	4	The size of one sector of program flash in bytes.
AvailableCommands	no	7	4	The set of commands supported by the boot ROM.
CRCCheckStatus	no	8	4	The status of the application CRC check.
VerifyWrites	yes	10	4	Controls whether the boot ROM verifies writes to flash. The VerifyWrites feature is enabled by default
MaxPacketSize	no	11	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	no	n	4	List of memory regions reserved by the boot ROM. Returned as value pairs (<startaddress-of-region>,<end-addressof-region>).
RAMStartAddress	no	14	4	Start address of RAM
RAMSizeInBytes	no	15	4	RAM size in bytes.
SecurityState	no	17	4	Security status
UniqueDeviceId	no	18	4	Unique device identification
TargetVersion	no	24	4	Target version
FlashPageSize	no	27	4	Flash page size in bytes

Table continues on the next page...

Table continued from the previous page...

Name	Writable	Tag Value	Size	Description
IrqNotifierPin	yes	28	4	IRQ notifier pin bit [7:0] pin, bit[15:8] port, bit[31] enable

18.7.2.8 UART ISP

18.7.2.8.1 Introduction

The boot ROM integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If UARTn is used to connect to the boot ROM, then the UARTn_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the boot ROM detects the ping packet (0x5A 0xA6) on UARTn_RX, the boot ROM firmware executes the autobaud sequence.

If the baudrate is successfully detected, then the boot ROM sends a ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The boot ROM then enters a loop, waiting for boot ROM commands via the UART peripheral.

NOTE

The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this instance, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, 115200 and 230400.

Packet transfer: After autobaud detection succeeds, boot ROM communications can take place over the UART peripheral. The following flow charts show:

- How the host detects an ACK from the target
- How the host detects a ping response from the target
- How the host detects a command response from the target

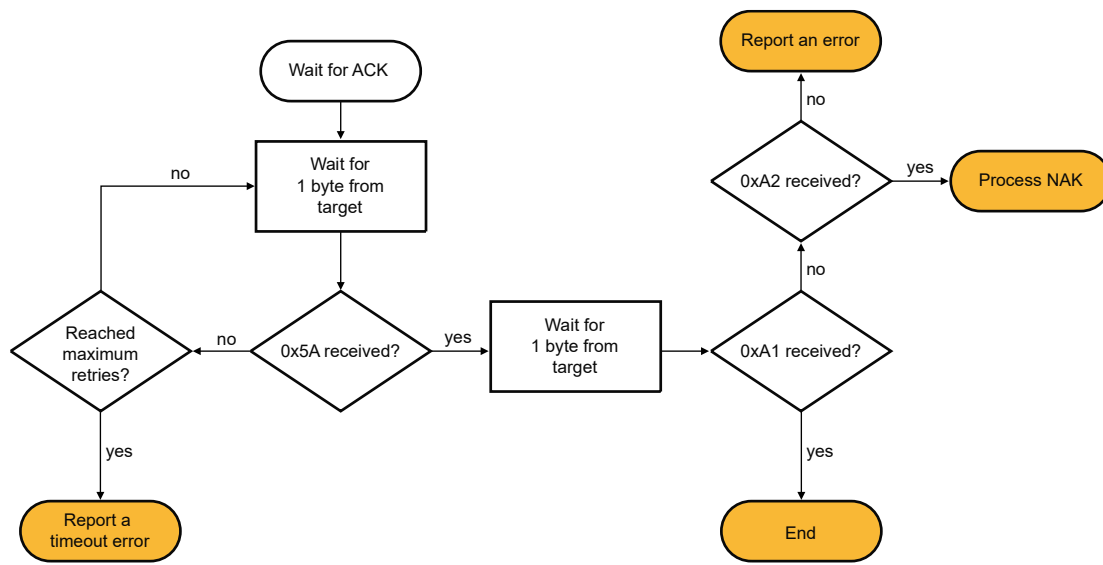


Figure 71. Host reads an ACK from target via UART

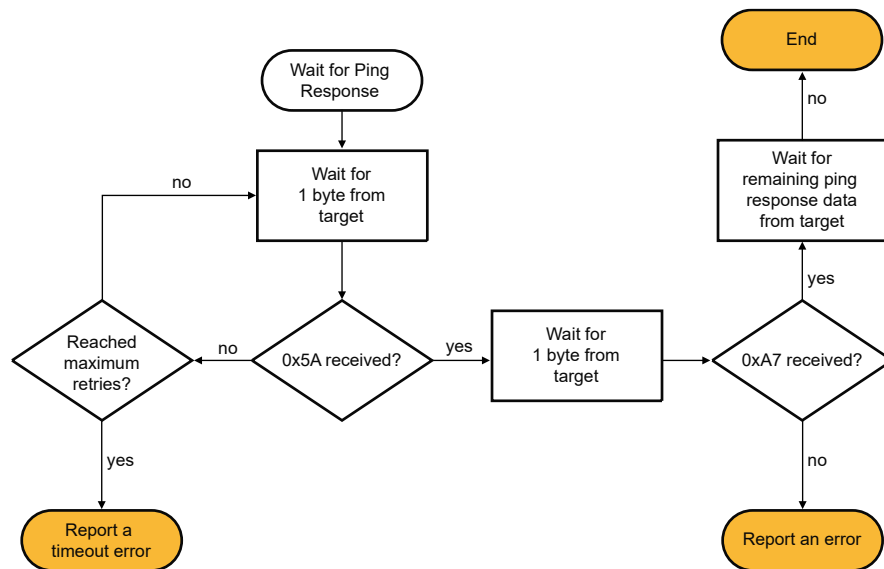


Figure 72. Host reads a ping response from target via UART

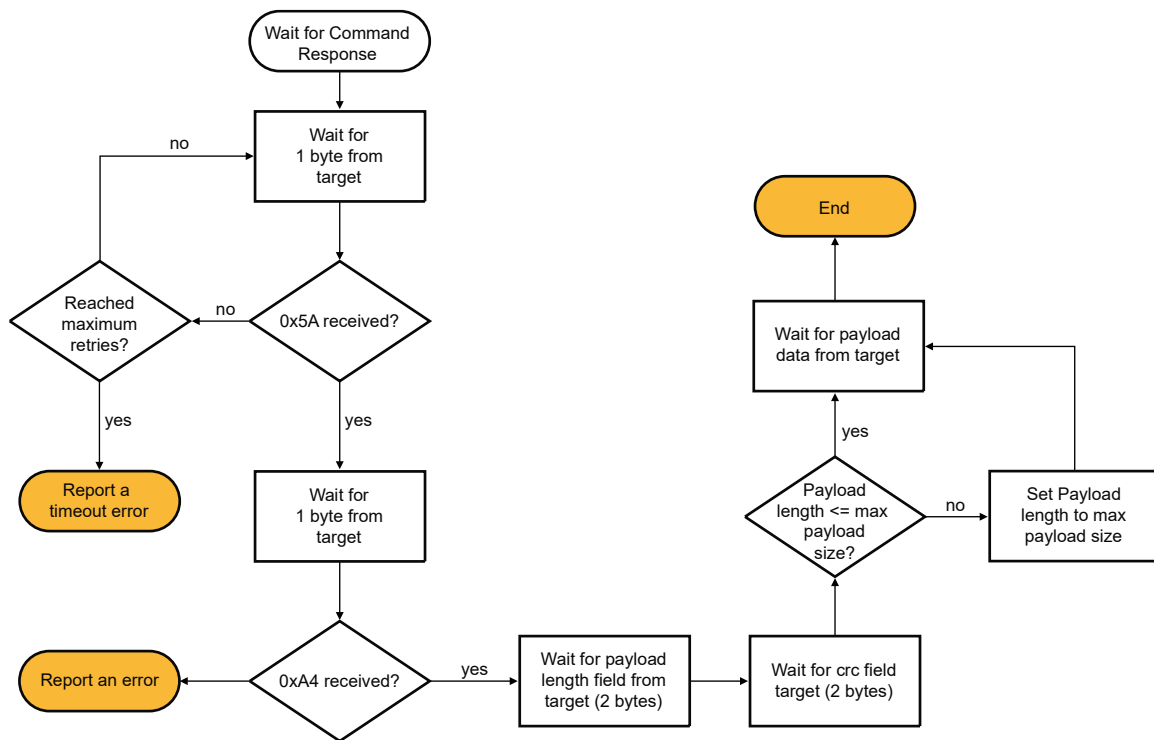


Figure 73. Host reads a command response from target via UART

18.7.2.8.2 UART ISP command format

See [Boot ROM packet types](#) for more details.

18.7.2.8.3 UART ISP response format

See [Boot ROM packet types](#) for more details.

18.7.2.8.4 UART ISP data format

See [Boot ROM packet types](#) for more details.

18.7.2.8.5 UART ISP commands

See [Boot ROM packet types](#) for more details.

18.7.2.9 SPI In-System Programming

18.7.2.9.1 Introduction

The boot ROM supports In-System Programming or serial boot via the SPI peripheral.

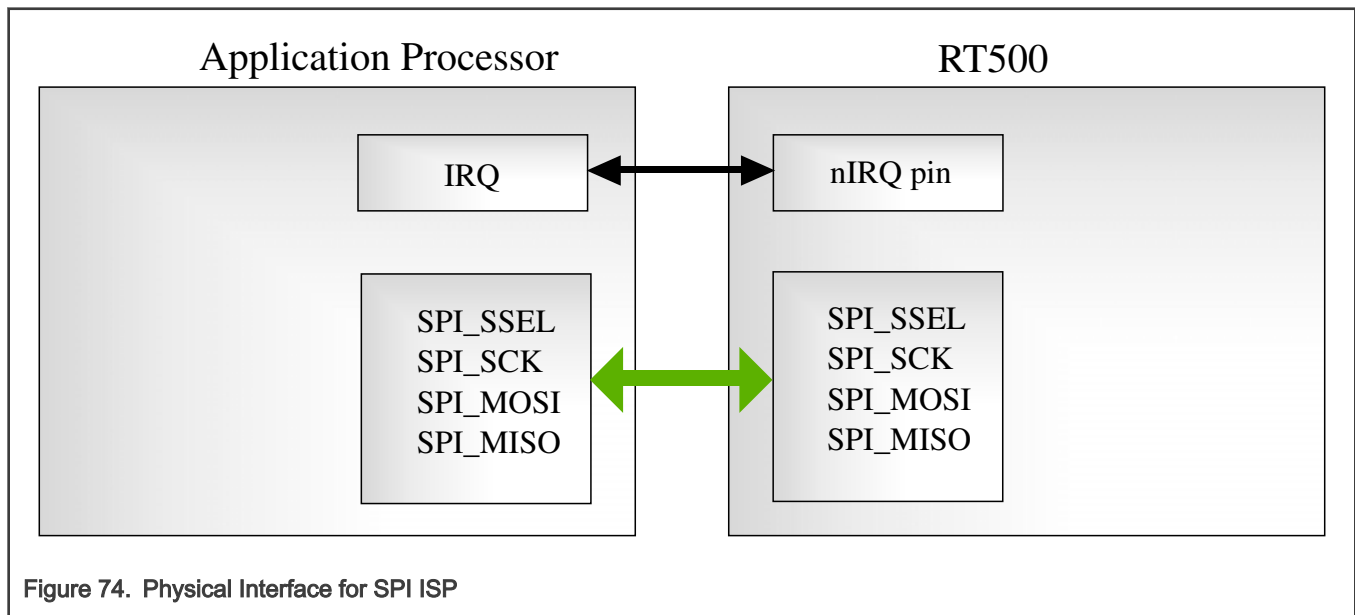
The maximum supported baud rate of the SPI depends on the core clock frequency when the boot ROM is running. The typical baud rate is 2Mbit/s with the factory settings. The actual baud rate is up to 50Mbit/s when the core is running at high-speed boot mode.

The SPI peripheral in the boot ROM, which needs to be configured to Mode 3 and 8 data bits mode, serves as an SPI slave device. Each transfer, therefore, must be started by the host, and each outgoing packet should be fetched by the host as well.

Compared to UART Peripheral, the transfer on SPI is slightly different:

- The host receives 1 byte after it sends out any byte.
- Received bytes should be ignored when the host is sending out bytes to the target
- The host starts reading bytes by sending 0x00s to target
- The byte 0x00 is sent as a response to host if the target is under the following conditions:
 - Processing incoming packet
 - Preparing outgoing data
 - Received invalid data

See the below figure for the typical physical connection between the host and the boot ROM device.



The following flowcharts show how the host reads a ping response, an ACK and a command response from target via SPI without the nIRQ pin enabled.

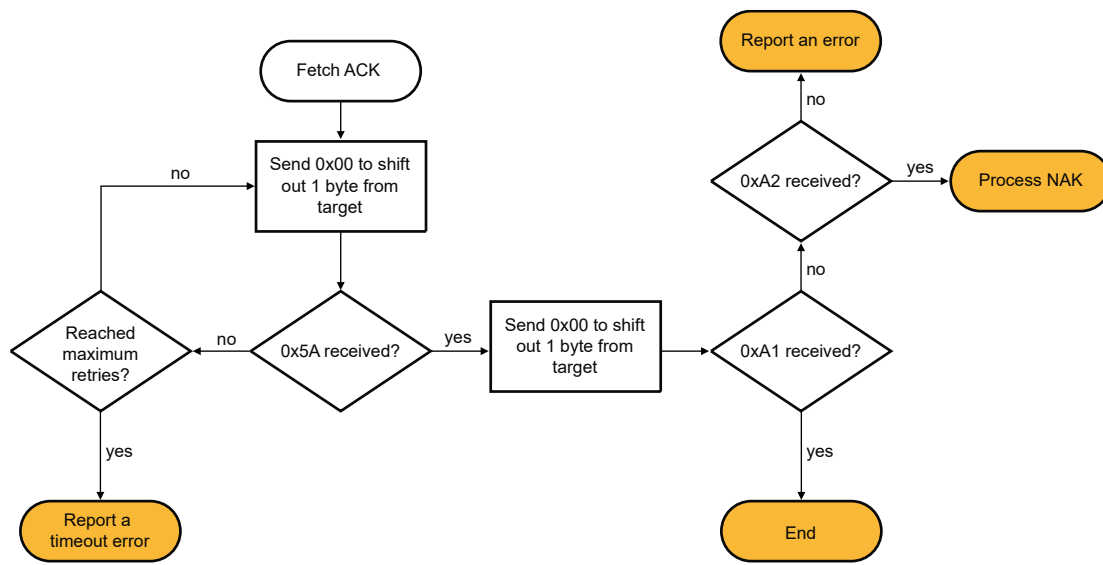


Figure 75. Host reads ACK from target via SPI

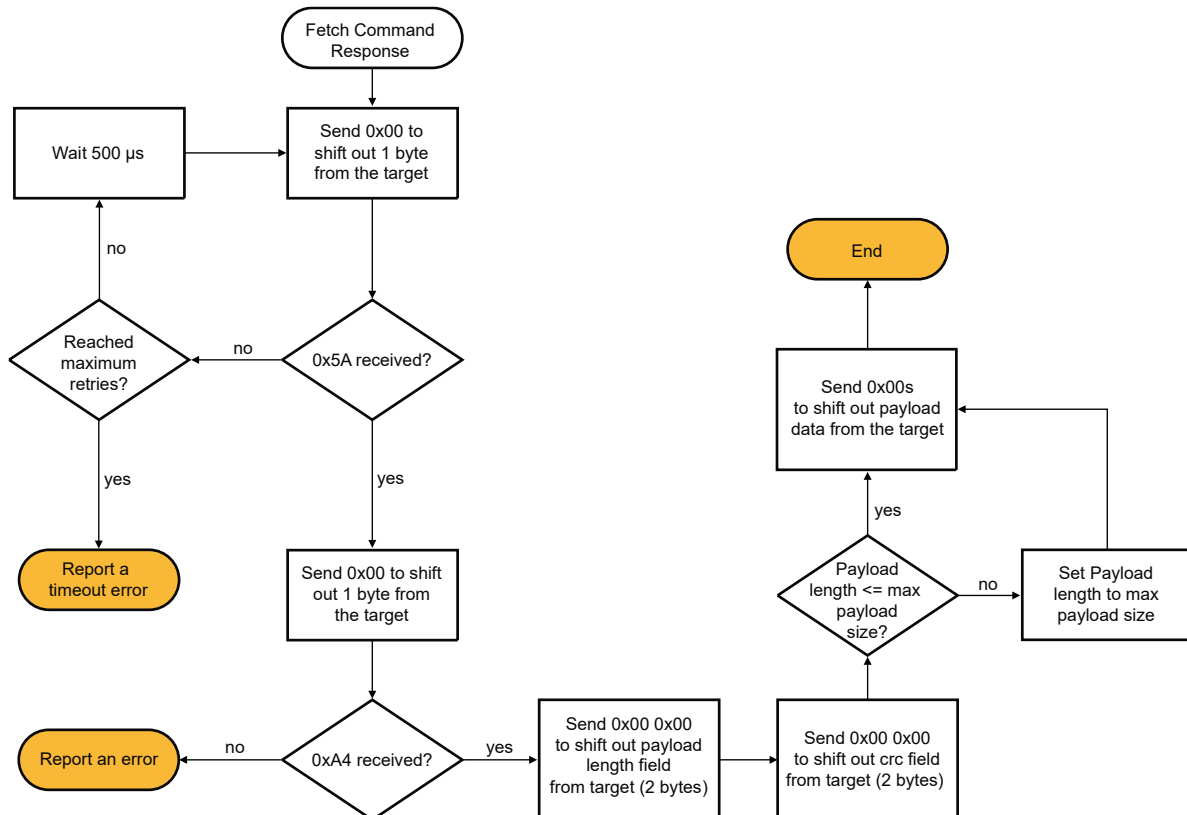


Figure 76. Host reads response from target via SPI

The application processor (namely, host) can use the feature nIRQ pin to improve the communication performance between it and the boot ROM, which can be enabled by the SetProperty <nIRQ pin tag> <nIRQ pin> command.

Once being enabled, the host waits until it catches a negative edge on the nIRQ pin before reading any data from the boot ROM, and it also needs to suspend until the nIRQ pin is HIGH before sending any commands/data to the boot ROM.

18.7.2.9.2 SPI ISP command format

See [Boot ROM packet types](#) for more details.

18.7.2.9.3 SPI ISP response format

See [Boot ROM packet types](#) for more details.

18.7.2.9.4 SPI ISP data format

See [Boot ROM packet types](#) for more details.

18.7.2.9.5 SPI ISP commands

See [Boot ROM packet types](#) for more details.

18.7.2.10 I2C In-System Programming

The bootloader supports In-System Programming or serial boot via the I2C peripheral, where the I2C peripheral serves as the I2C slave. The bootloader uses 0x10 as the 7-bit I2C slave address and supports up to 400 Kbit/s speed during transfer. The maximum supported I2C baud rate depends on the core clock frequency when the bootloader is running.

The typical baud rate is 400 Kbit/s with factory settings. The I2C peripheral serves as an I2C slave device, hence each transfer should be started by the host, and each outgoing packet should be fetched by the host as well.

- An incoming packet is sent by the host with a selected I2C slave address and the direction bit is set to write.
- An outgoing packet is read by the host with a selected I2C slave address and the direction bit is set as read.
- 0x00 is sent as the response to host if the target is busy with processing or preparing data.

The following charts show the communication flow of the host reading the ACK packet and the corresponding responses from the target.

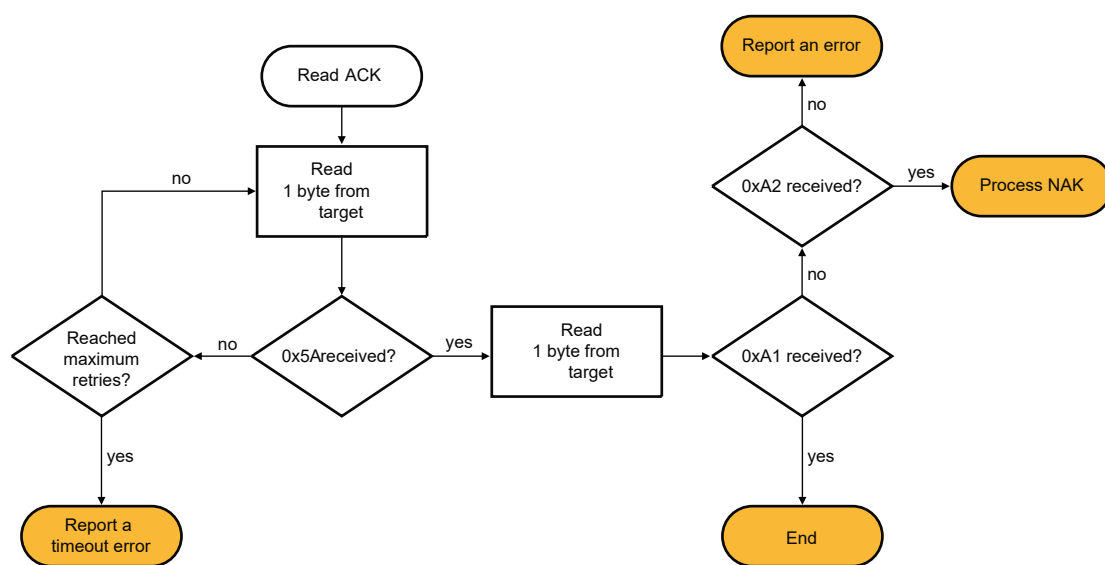
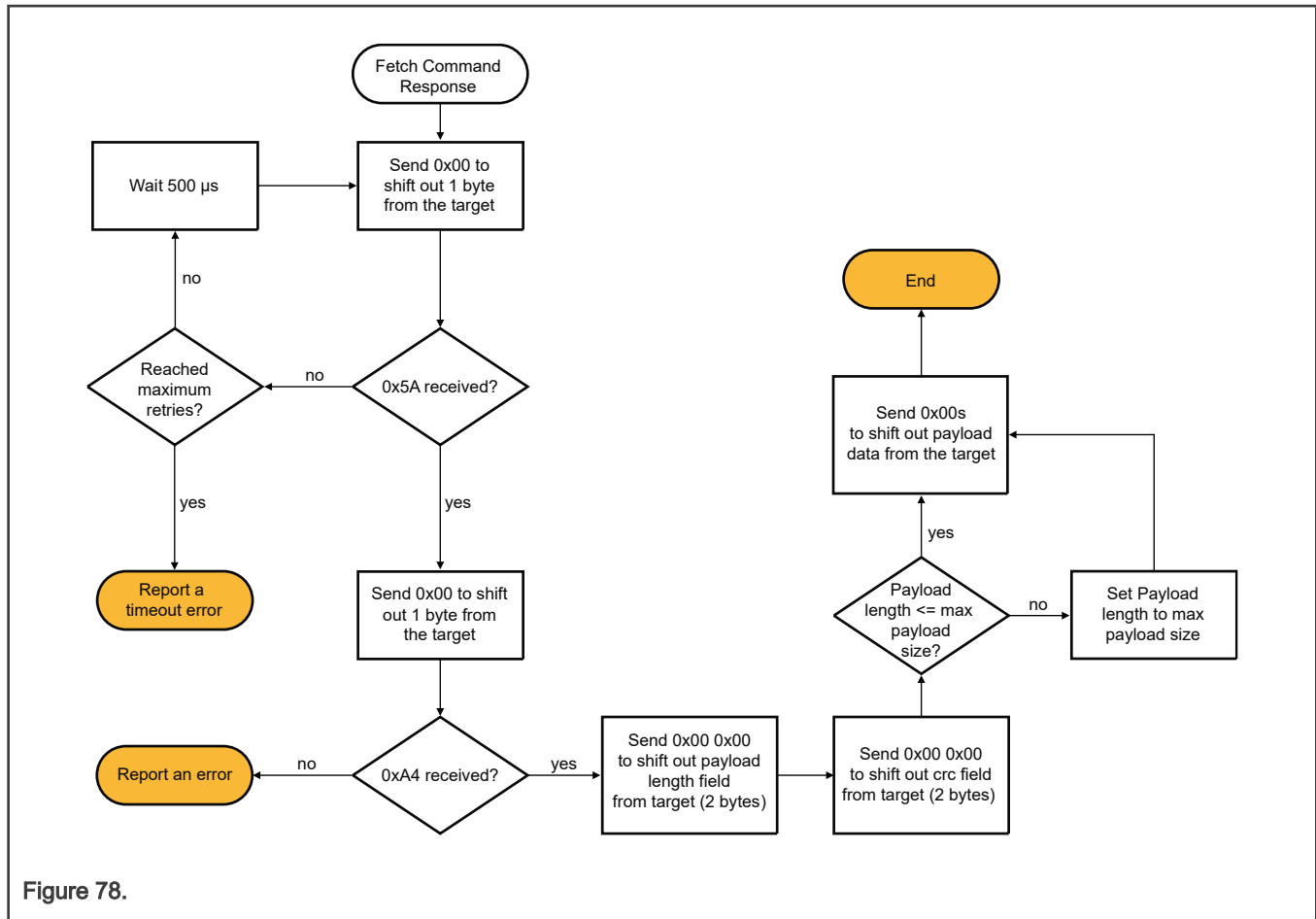


Figure 77. Host reads ACK packet from target via I2C



18.7.2.10.1 Introduction

18.7.2.10.2 I2C ISP data format

See [Boot ROM packet types](#) for more details.

18.7.2.10.3 I2C ISP command format

See [Boot ROM packet types](#) for more details.

18.7.2.10.4 I2C ISP response format

See [Boot ROM packet types](#) for more details.

18.7.2.10.5 I2C ISP commands

See [Boot ROM packet types](#) for more details.

18.7.2.11 USB In-System Programming

18.7.2.11.1 Introduction

The boot ROM supports In-System Programming using the USB peripheral. The target is implemented as USB-HID device classes.

When transfer data through USB-HID device class, USB-HID does not use framing packets. Instead, the packetization, inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control. The built-in CRC of each USB packet provides the required error detection.

NOTE

The device enumerates as a composite device with HID and DFU interfaces. But only HID interface is used for ISP mechanism and DFU interface is non-functional.

18.7.2.11.1.1 Device Descriptor

The boot ROM configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x1FC9
- PID = 0x0023

Default Strings:

- Manufacturer [1] = "NXP SEMICONDUCTOR INC."
- Product [2] = "USB COMPOSITE DEVICE"

18.7.2.11.1.2 Endpoints

The HID peripheral uses 3 endpoints:

- Control (0)
- Interrupt IN (1)
- Interrupt OUT (2)

The Interrupt OUT endpoint is optional for HID class devices, but the MCU boot ROM uses it as a pipe, where the firmware can NAK send requests from the USB host.

18.7.2.11.1.3 HID Reports

There are 4 HID reports defined and used by the boot ROM USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

Report ID	Packet Type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

For all reports, these properties apply:

Usage Min	1
Usage Max	1
Logical Min	0
	255

Table continues on the next page...

Table continued from the previous page...

Logical Max	
Report Size	8
Report Count	34

Each report has a maximum size of 34 bytes. This is derived from the minimum boot ROM packet size of 32 bytes, plus a 2-byte report header that indicates the length (in bytes) of the packet sent in the report.

NOTE

In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

0	Report ID
1	Padding
2	Packet Length LSB
3	Packet Length MSB
4	Packet [0]
5	Packet [1]
6	Packet [2]
N+4-1	...
	Packet [N-1]

This data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length, which is 56 bytes on Full-speed USB port while is 1016 bytes on High-speed USB port. The Packet Length header is written in little-endian format, and it is set to the size (in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

18.7.2.11.2 USB ISP command format

See [Command packet](#) for more details.

18.7.2.11.3 USB ISP response format

See [Command packet](#) for more details.

18.7.2.11.4 USB ISP data format

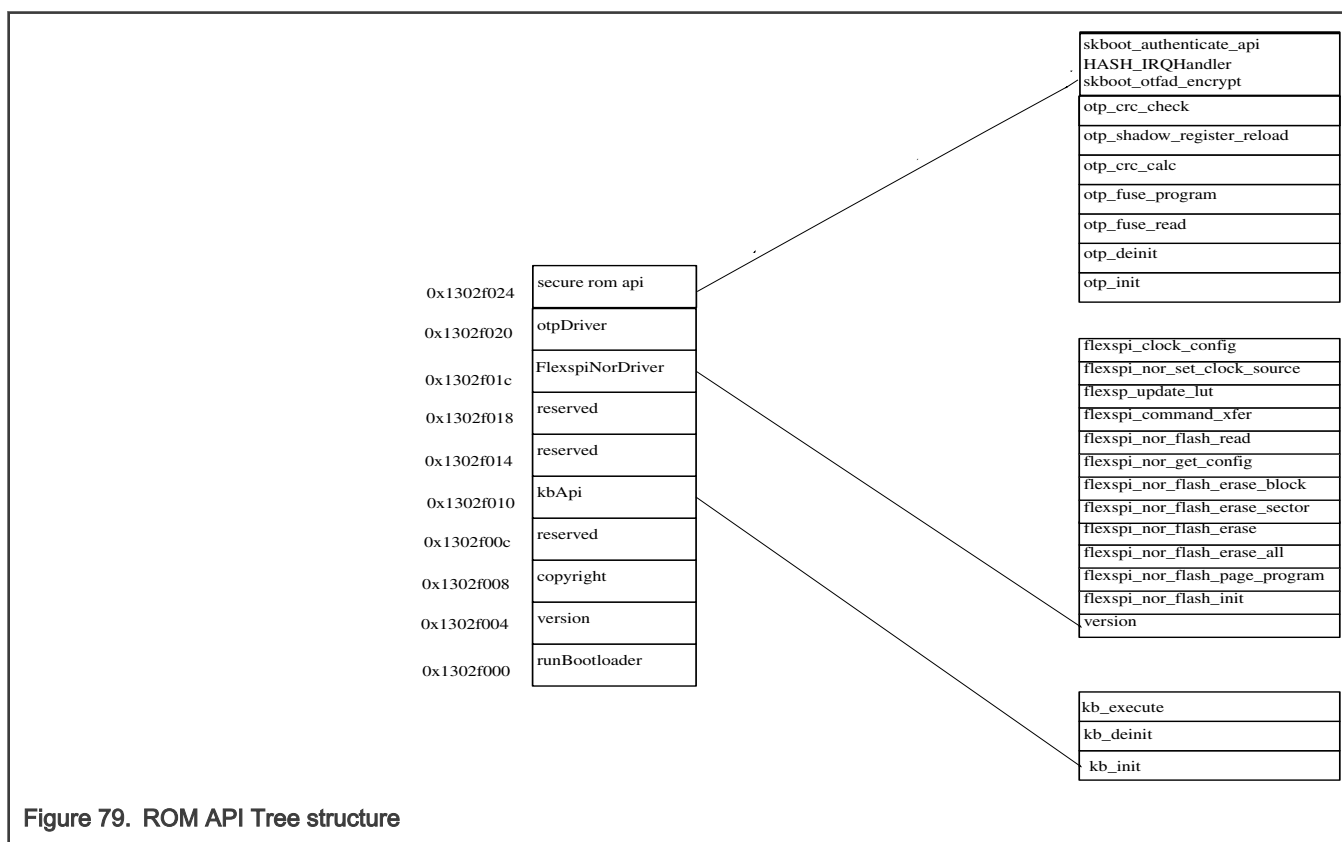
See [Command packet](#) for more details.

18.7.2.11.5 USB ISP commands

See [Command packet](#) for more details.

18.7.3 ROM API layout

Figure below shows the details of the ROM API layout. Since the application is not linked with the boot ROM library, the ROM includes look-up tables at fixed address that point to the ROM API addresses. The SDK includes examples for how to call these APIs.



18.7.3.1 API to invoke into ROM

ROM provides an API **runboot ROM** which can be used by user application to invoke into ROM with specified boot parameters as shown in table below.

Table 141. Location of runboot ROM API in ROM API

ROM offset	API
0x1302F024	secure rom api
0x1302F020	otpDriver
0x1302F01C	flexspiNorDriver
0x1302F018	Reserved
0x1302F014	Reserved
0x1302F010	kbApi
0x1302F00C	Reserved
0x1302F008	copyright
0x1302F004	version
0x1302F000	runboot ROM

Table 141. Location of runboot ROM API in ROM API

TAG	BOOT MODE	INTERFACE	INSTANCE	FLEXSPIboot image index	Reserved	COMBINATI ON	BOOT ACTION
[31:24]	[23:20]	[19:16]	[15:12]	[13:8]	[07:00]		
0xEB	0	0	X	X	X	0xEB00XXXX	Serial MASTR BOOT: USART
	0	1	X	X	X	0xEB01XXXX	Serial MASTR BOOT: I2C
	0	2	X	X	X	0xEB02XXXX	Serial MASTR BOOT: SPI
	0	3	X	X	X	0xEB03XXXX	Serial MASTR BOOT: USB HID
	0	4	X	0	X	0xEB04X0XX	MASTR BOOT: FlexSPI: boot image 0
	0	4	X	1	X	0xEB04X1XX	MASTR BOOT: FlexSPI: boot image 1
	0	5	X	X	X	0xEB05XxXX	MASTR BOOT: 1 bit SPI Flash boot
	0	8	0	X	X	0xEB080XXX	MASTR BOOT: MMC (INSTANCE 0)
	0	8	1	X	X	0xEB081XXX	MASTR BOOT: MMC (INSTANCE 1)
	1	0	X	X	X	0xEB10XXXX	ISP BOOT: USART
	1	1	X	X	X	0xEB11XXXX	ISP BOOT: I2C
	1	2	X	X	X	0xEB12XXXX	ISP BOOT: SPI
	1	3	X	X	X	0xEB13XXXX	ISP BOOT: HID

The prototype of the **runboot ROM** is shown below.

User application can pass a parameter which contains the boot mode options

*void runboot ROM (void *arg)*

The definition of user app boot mode options is below

```
typedef struct _user_app_boot_invoke_option
{
    union
    {
        struct
        {
            uint32_t reserved : 8;
            uint32_t boot_image_index : 4;
            uint32_t instance : 4;
            uint32_t boot_interface : 4;
            uint32_t mode : 4;
            uint32_t tag : 8;
        } B;
        uint32_t U;
    } option;
} user_app_boot_invoke_option_t;
```

Description of the *user_app_boot_invoke_option_t* is in the table below.

Table 142. Combination of the user boot options

WORD	OFFSET	FIELD	DESCRIPTION
	[31:24]	TAG	Must be '0xEB'
	[23:20]	Boot mode	0: Master boot mode; 1: ISP boot
	[19:16]	Boot interface	0: USART 2: SPI 3: USB HID 4:FlexSPI 7:MMC
	[15:12]	Boot instance	0 or 1; This instance is only used when boot interface is MMC, for other interfaces it is ignored
	[11:08]	FlexSPI boot image index	FlexSPI boot image index for FlexSPI NOR flash
	[07:00]	Reserved	Reserved

The detail combination of the user app boot options is shown below.

TAG	BOOT MODE	INTERFACE	INSTANCE	FLEXSPI boot image	Reserved	COMBINATI ON	BOOT ACTION
[31:24]	[23:20]	[19:16]	[15:12]		[07:00]		

Table continues on the next page...

Table continued from the previous page...

0xEB	0	0	X	X	X	0xEB00XXXX	Serial MASTR BOOT: USART
	0	1	X	X	X	0xEB01XXXX	Serial MASTR BOOT: USART
	0	2	X	X	X	0xEB02XXXX	Serial MASTR BOOT: SPI
	0	3	X	X	X	0xEB03XXXX	Serial MASTR BOOT: USB HID
	0	4	X	0	X	0xEB04X0XX	MASTR BOOT: FlexSPI: boot image 0
	0	4	X	1	X	0xEB04X1XX	MASTR BOOT: FlexSPI: boot image 1
	0	5	X	X	X	0xEB05XxXX	MASTR BOOT: 1 bit SPI Flash boot
	0	8	0	X	X	0xEB080XXX	MASTR BOOT: MMC (INSTANCE 0)
	0	8	1	X	X	0xEB081XXX	MASTR BOOT: MMC (INSTANCE 1)
	1	0	X	X	X	0xEB10XXXX	ISP BOOT: USART
	1	1	X	X	X	0xEB11XXXX	ISP BOOT: I2C
	1	2	X	X	X	0xEB12XXXX	ISP BOOT: SPI
	1	3	X	X	X	0xEB13XXXX	ISP BOOT: HID

Examples below show how to use the API to invoke into ROM by user application:

Example 1:

```

user_app_boot_invoke_option_t boot_options = {0};
boot_options.option.B.tag = 0xeb;
boot_options.option.B.mode = 0x1;
boot_options.option.B.boot_interface = 0x2;
runboot ROM((void*)&boot_options) // After call this, system will reset and run
                                   into ROM to ISP mode using SPI.

```

Example 2:

```

user_app_boot_invoke_option_t boot_options = {0};
boot_options.option.B.tag = 0xeb;
boot_options.option.B.mode = 0x0;
boot_options.option.B.boot_interface = 0x4;
boot_options.option.B.boot_image_index = 0x0;
runboot ROM((void*)&boot_options) // After call this, system will reset and run
                                   into ROM to boot the image 0 via FLEXSPI interface.

```

18.7.3.2 KB APIs Demo

This section uses the KB APIs demos to introduce the KB APIs usage and helps the user to know how to use the APIs. In addition, the section uses the practical application scenarios to introduce the convenience, flexibility, and practicality of the KB APIs.

18.7.3.2.1 Generate SB file for the KB APIs call

The Secure Binary (SB) image format is a command-based firmware update image. In the KB APIs call, version 2.1 of the SB image format will be used, and more details of the SB file introduction can be referred to in ROM firmware update using SB file in the RT500 user manual document. The following command will describe how to generate the version 2.1 of the SB file:

```

elftosb.exe -V -d -f rt5xx -k "keys\non_zero_sbkek.txt" -c "commandFile.bd"
-o ".\output\s_b_file_program.sb2" -s keys_and_certs/selfsign_privatekey_rsa2048.pem -S
keys_and_certs/selfsign_v3.der.crt -R keys_and_certs/selfsign_v3.der.crt -h "RKTH.bin"

elftosb.exe -V -d -f <chip-family> -k "<sbkek file>" -c "<SB command BD file>" -o "<output
SB file>" -s <Path to private key for signing> -S <Path to certificate files for signing> -R
<Path to root key certificate file(s) for verifying other certificates> -h "<Path to output
hash of hashes of root keys>"

```

In the process of the generating the SB file, user should prepare the files as follow:

- elftosb tool
- chip-family (set rt5xx for the RT500)
- <sbkek file> (the type of the kek file should be .txt, and the data in the file includes 64 key numbers)

Example:

```
0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
```

- <SB command BD file>(commands will be executed in the SB file received by the device, the type of the file is .bd)

Example:

```

options {
flags = 0x8; // 0x8 rom signed 0x4 //encrypted
buildNumber = 0x1;
productVersion = "1.00.00";
componentVersion = "1.00.00";
secureBinaryVersion = "2.1";
}

```

```

sources {
mainImage = " firmware.bin ";
}
section (0)
{
load 0xc0000001> 0x1C000;
enable @9 0x1C000;
erase 0x8000000..0x08200000;
load mainImage > 0x08000000;
}

```

- <output SB file> (the file is the output SB file, the type of the file is .sb)
- <Path to the private key for signing> (provided by the user)
- <Path to certificate files for signing> (If use chain certificates, the first certificate will be the self-signed root key certificate)
- <Path to root key certificate file(s) for verifying other certificates> (max of 4 root key certs are allowed and the rest are ignored, one of the certificates must match the first certificate passed with -S/--cert arg)
- <Path to output hash of hashes of root keys> (if the argument is not provided then the tool creates hash.bin in the working directory by default)

NOTE

The data in hash.bin should be loaded into the RKTH [255:0] in OTP fuse ROKTH region.

18.7.3.2.2 Generate Key-store for the KB APIs call

The SB 2.1 processing by ROM depends on the presence of a valid key store setup with the SBKEK key code and also depends on the presence of RKTH hash (RKTH) in the OTP fuse.

The example below shows generation of the key-store, which is used for KB API calls to process the SB file in the kb_execute API. Steps of generating the key-store: using the ROM bootloader key provisioning commands:

```

blhost -p comxx -- key-provisioning enroll
blhost -p comxx -- key-provisioning set_user_key 3 sbkek.bin
blhost -p comxx -- key-provisioning read_key_store key_store_user.bin

```

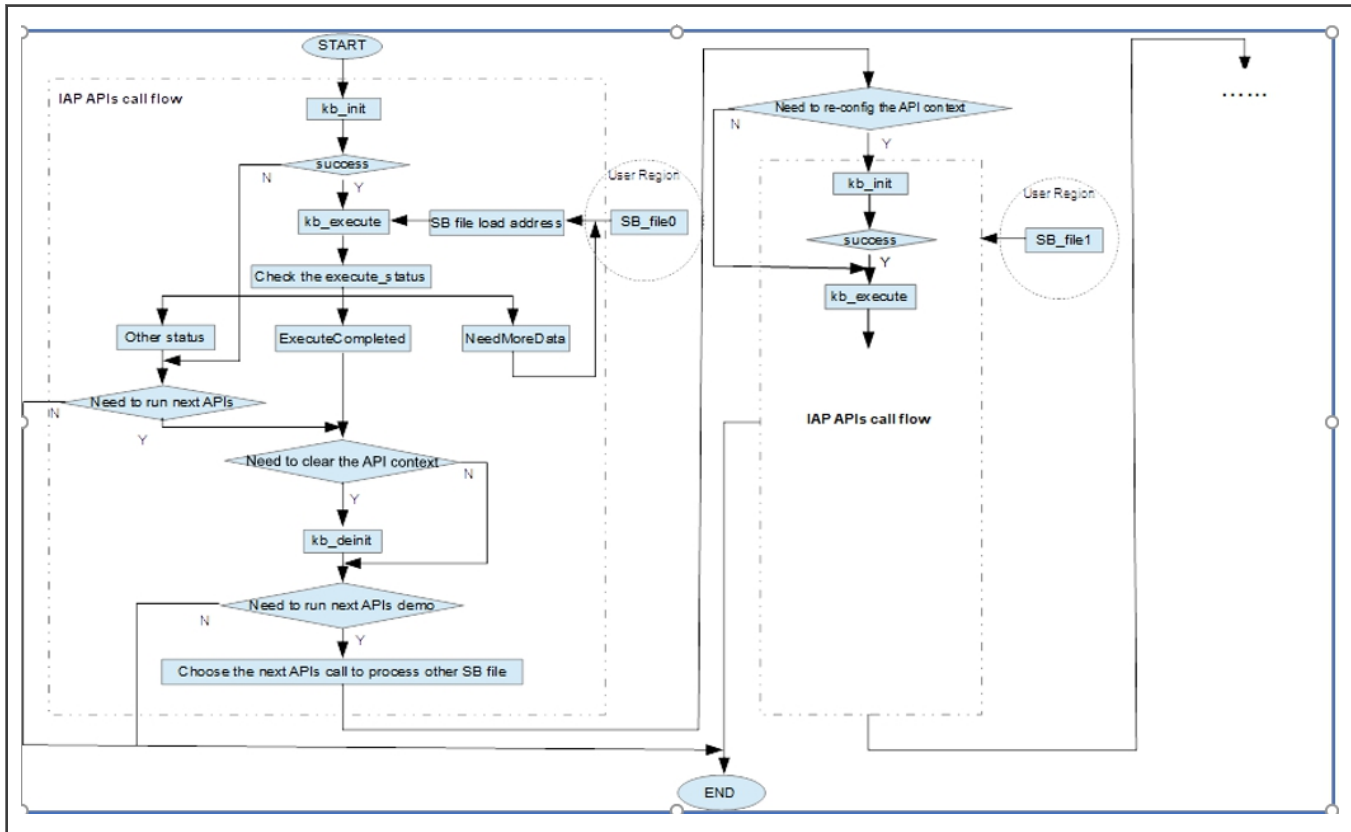
NOTE

sbkek.bin (should be inverted order to sbkek.txt, which is provided in the file of generating the SB file)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EF	CD	AB	89	67	45	23	01	EF	CD	AB	89	67	45	23	01
00000010	EF	CD	AB	89	67	45	23	01	EF	CD	AB	89	67	45	23	01

18.7.3.2.3 Create application project for the KB APIs call

This section is used for helping the user to know how to call the KB APIs in the application project. The KB APIs are defined in the `bl_kb_api.c` and `api_kb_api.h` in the ROM code. This application will call these KB APIs to transfer sequential chunks of data from the boot image (SB file) to the device. In the process of the transferring in SB file, the commands in the SB file will be executed to update the firmware of the device in real-time.



The figure above introduces the KB APIs call flow. From the flow diagram, the `kb_init` will be called first to initialize and configure the KB API context. If the `kb_init` return status is succeeded, then the `kb_execute` will be called to process the SB file, which is transferred from the user region (internal flash region, external flash region and RAM region a). In the running process of the `kb_execute` function, the return status includes three kinds of the results, which can be referred to in Table 4 The IAP bootloader API status code. If the return status is `kStatus_RomApiNeedMoreData`, it seems that more data in the `SB_file0` should be transferred to the `kb_execute` to process, and the process has not reached the end of the SB file data. The other status returned from the `kb_execute` will be thought that `kb_execute` API has not been executed successfully. In the next flow, user can decide whether or not to run the next KB APIs call for other `SB_file1`. If not, the application will run to the end, otherwise the `kb_deinit` will be executed to clear the KB APIs context parameters, which will be reconfigured based on user's decision in the next API `kb_init`. When the `kStatus_RomApiExecuteCompleted` has been received, it seems that `SB_file0` has been completely received and processed so as to update the firmware in the `SB_file0` successfully.

After the `kb_deinit` call, the APIs context will be cleared and needs to be reconfigured in the next KB APIs call. If not calling the `kb_deinit`, in the step of choosing the next APIs call to process other `SB_file0`, the user can directly continue to call the APIs to process the other SB files based on the input command option through the serials port.

When the next `SB_file1` has been chosen, the user will call the `kb_init` or not reconfigure the APIs context. The next steps will be the same as the former KB APIs call flow, and the KB APIs can be called repeatedly to finish different firmware update work. In addition, in the process of processing the SB file in the `kb_execute`, only the load address of the SB file data will be transferred to the `kb_execute`, the data of the SB file is executed with the packet chunk. In this situation, the ram size of the device is very small, just like 1k, and the needed updated SB file firmware is larger than the ram size. Users can also use the KB APIs call to finished the firmware update. Users should get and load the packet chunk of the SB file into the ram region like the following function `get_data_from_sbfile()`, and then call the `kb_execute` to process the ram data repeatedly until to the end of the SB file. In

transferring the SB file chunk to the device, the user can use different channels to receive the chunk data like UART, SPI, I2C, USB, OTP interface.

```
#define BOOTLOADER_TREE_LOCATION (0x1302f000)
#define USER_BUFFER_LENGTH 0x4000

bootloader_tree_t *romApiTree = (bootloader_tree_t *)BOOTLOADER_TREE_LOCATION;
kb_interface_t *s_romAPIInterface;
s_romAPIInterface = romApiTree->kbApi;
kb_session_ref_t *context;
uint8_t user_buf[1024];
kb_options_t options = {.version = kRomApiVersion,
.buffer = user_buf, //array defined by user
.bufferLength = USER_BUFFER_LENGTH, // defined by user
.op = kRomLoadImage, //used for IAP feature
{
.loadSB =
{
.profile = 0,
.minBuildNumber = 1,
.overrideSBBootSectionID = 1,
.userSBKEK = 0, //Set Null then IFR SBKEK will be used
.regionCount = NULL,
.regions = NULL, // If regions is NULL, all memory is accessible by the SB file
},
}
.security_options.selection = kQspiFlash_KeyStore,
.security_options.storage_config_option[0] = 0xc0000001,
.security_options.storage_config_option[1] = 0,
};
status_t status = kStatus_Fail;
status = s_romAPIInterface ->kb_init((&context), &options);
if (status != kStatus_Success)
{
return status;
}
while ((status = get_data_from_sbfile(&buffer, &size)) == kStatus_Success)
{
status = s_romAPIInterface ->kb_execute(context, &buffer, size);
if (status == kStatus_RomApiNeedMoreData)
{
debug_printf("Warning: Success to execute ROM API, but Need more data \n");
//continue to receive the data from SB file for processing
}
else if (status == kStatus_RomApiExecuteCompleted)
{

```

```

debug_printf("Completed: Success to execute ROM API erase\n");
break;    //finish the SB file firewarre update and break
}
else if (status != kStatus_Success)
{
debug_printf("Error: Failed to run API kb_execute, status is hex(%x) \n", (uint32_t)status);
return status;    //return the fail status or other actions
}
}
status = s_romAPIInterface ->kb_deinit(context);
if (status != kStatus_Success)
{
debug_printf("Error: Failed to kb_deinit\n");
return status;
}

```

In the KB API calling project, the HASHCRYPT_IRQHandler(void) must be defined to call the skboot_hashcrypt_irq_handler() API in the user application project interrupt vector table. Because in the calling the kb_execute ROM API, the authentication of the SB file will use the interrupt that skboot_hashcrypt_irq_handler() of the ROM interrupt, which should be synchronized to the application interrupt vector table.

```

skboot_authenticate_interface_t *s_authApiInterface;
s_authApiInterface = romApiTree->skbootAuthenticate;
void HASHCRYPT_IRQHandler(void)
{
s_authApiInterface->skboot_hashcrypt_irq_handler();
}

```

18.7.4 OTP Driver APIs

This ROM based OTP Driver APIs are available on all parts.

18.7.4.1 Features

- APIs to program/read the words from OTP block
- APIs to compute and check the OTP CRC checksum

18.7.4.2 Functional Description

The ROM boot ROM contains an OTP API set to simplify the OTP programming during development or production life-cycle.

18.7.4.2.1 The OTP API location

See the details of OTP API location in [Figure 79](#).

18.7.4.3 API Description

OTP APIs provide functions to program, read, reload shadow registers for the OTP block. They also support calculating and verify the CRC checksum for specified OTP region.

Table 143. OTP API calls

Function prototype	API Description
status_t otp_init(uint32_t src_clk_freq);	Initialize OTP block based on the source clock frequency
status_t otp_deinit(void);	De-initialize OTP block
status_t otp_fuse_read(uint32_t addr, uint32_t *data);	Read OTP Fuse value from specified OTP word index
status_t otp_fuse_program(uint32_t addr, uint32_t data, bool lock);	Program data to specified OTP word index and lock it if lock parameter is true
status_t otp_crc_calc(uint32_t *src, uint32_t numberOfWords, uint32_t *crcChecksum);	Calculate CRC checksum based on specified OTP region
status_t otp_shadow_register_reload(void);	Reload all the shadow registers from OTP Fuse block
status_t otp_crc_check(uint32_t start_addr, uint32_t end_addr, uint32_t crc_addr);	Perform CRC check based on provided OTP Fuse region

18.7.4.3.1 otp_init

This routine configures the OTP controller based on the specified source clock frequency.

Routine	otp_init
Prototype	status_t otp_init(uint32_t src_clk_freq);
Input Parameter	Parameter0: desired source clock frequency
Result	Error code: 0= no error
Description	Configures the OTP controller based on the source clock frequency for OTP controller

18.7.4.3.1.1 Parameter0: desired source clock frequency

This frequency is the clock rate that the CPU will be using to access the OTP controller.

18.7.4.3.1.2 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful

18.7.4.3.2 otp_deinit

This routine shutdown the OTP controller.

Routine	otp_deinit
Prototype	status_t otp_deinit(void);
Input Parameter	None
Result	Error code: 0= no error
Description	Shutdown OTP controller

18.7.4.3.2.1 Error or return codes

Return code	Error code	Description
0	kStatus_Success	The operation is successful

18.7.4.3.3 otp_fuse_read

Routine	Routine
Prototype	status_t otp_fuse_read(uint32_t addr, uint32_t *data);
Input Parameter	Parameter0: The OTP word index Parameter1: Data buffer to hold the return value from OTP controller
Result	Error code: 0= no error
Description	Read Fuse value from specified OTP word index

18.7.4.3.3.1 Parameter0: OTP word index

The OTP word index parameter specifies the OTP FUSE word to be read out.

18.7.4.3.3.2 Parameter1: Data buffer

The Data buffer parameter provides the data buffer address which will hold the return value from the OTP controller if the API call is successful.

18.7.4.3.3.3 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1 is invalid
21001	kStatus_OTP_InvalidAddress	The OTP Word index is invalid
21004	kStatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckFail	ECC Check Failed
21006	kStatus_OTP_Locked	The Fuse word is locked

18.7.4.3.4 otp_fuse_program

Routine	otp_fuse_program
Prototype	status_t opt_fuse_program(uint32_t addr, uint32_t data, bool lock);
Input Parameter	Parameter0: The OTP word index

Table continues on the next page...

Table continued from the previous page...

Routine	otp_fuse_program
	Parameter1: Data to be programmed Parameter2: Lock flag
Result	Error code: 0= no error
Description	Program data to specified OTP word index and lock it as requested

18.7.4.3.4.1 Parameter0: The OTP word index

This parameter indicates the OTP word index to be programmed.

18.7.4.3.4.2 Parameter1: Data to be programmed

The parameter specifies the data to be programmed.

18.7.4.3.4.3 Parameter2: Lock flag

The parameter determines whether the OTP word index will be locked after being programmed or not. Valid value: true (1) or false (0).

18.7.4.3.4.4 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1 is invalid
21001	kStatus_OTP_InvalidAddress	The OTP address is invalid
21002	kStatus_OTP_ProgramFail	The OTP program failed
21004	kStatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckFail	OTP ECC Check failed during operation
21006	kStatus_OTP_Locked	OTP Word is locked

18.7.4.3.5 otp_crc_calc

Routine	otp_crc_calc
Prototype	status_t otp_crc_calc(uint32_t *src, uint32_t numberOfWords, uint32_t *crcChecksum);
Input Parameter	Parameter0: The Buffer that stores the source data Parameter1: Number of Words for CRC calculation Parameter2: The buffer that will hold the checksum result

Table continues on the next page...

Table continued from the previous page...

Routine	otp_crc_calc
Result	Error code: 0= no error
Description	Calculate the checksum for a specified buffer and store the data into the output buffer. Note that otp_crc_calc() is only used for the CRC checksums of OTP configuration. It should not be used for the ImageHeader::crcChecksum. Also, otp_crc_calc() should not be used for the CRC bitfields within an OTP configuration register. For example, DCFG_CC_SOCU register has the CRC8 bitfield, which uses the CRC-8/ITU algorithm.

18.7.4.3.5.1 Parameter0: The buffer that stores the source data

This parameter provides the buffer that stores the source data for CRC checksum computing, the pointer address must be 32-bit aligned.

18.7.4.3.5.2 Parameter1: Number of words for CRC calculation

The parameter indicates the number of word for CRC calculation.

18.7.4.3.5.3 Parameter2: The buffer that will hold the checksum result

This parameter points to the buffer that will hold the CRC result if the operation is successful.

18.7.4.3.5.4 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
4	kStatus_InvalidArgument	The parameter 1 or 3 is invalid

18.7.4.3.6 otp_shadow_register_reload

Routine	otp_shadow_register_reload
Prototype	status_t otp_shadow_register_reload(void);
Input Parameter	None
Result	Error code: 0= no error
Description	Reload all the shadow registers from OTP Fuse blocks

18.7.4.3.6.1 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful

Table continues on the next page...

Table continued from the previous page...

Return code	Error Code	Description
21004	kStatus_OTP_Error	Error happened during OTP operation
21005	kStatus_OTP_EccCheckFail	OTP ECC Check failed during operation

18.7.4.3.7 otp_crc_check

Routine	otp_crc_check
Prototype	status_t otp_crc_check(uint32_t start_addr, uint32_t end_addr, uint32_t crc_addr);
Input Parameter	Parameter 0: The start OTP word index for CRC check. Parameter1: The end OTP word index for CRC check. Parameter2: The OTP word which holds the expected checksum.
Result	Error code: 0= no error
Description	Compute the CRC checksum in the specified OTP range and compare it with the specified OTP word that holds the expected checksum.

18.7.4.3.7.1 Parameter0: The start OTP word index for CRC check

This parameter specifies the start OTP word index for CRC check.

18.7.4.3.7.2 Parameter 1: The end OTP word index for CRC check

This parameter indicates the end OTP word index for CRC check.

18.7.4.3.7.3 Parameter 2: The OTP word which holds the expected Checksum

The parameter points to the OTP word which holds the expected CRC checksum but may not be visible to the otp_fuse_read API due to being locked during programming.

18.7.4.3.7.4 Error or return codes

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
21001	kStatus_OTP_InvalidAddress	The OTP address is invalid
21004	kStatus_OTP_Error	Error happened during OTP operation
21008	kStatus_OTP_CrcCheckPass	The OTP CRC check passed

18.7.5 FLEXSPI FLASH Driver APIs

The ROM based FLEXSPI Flash Driver API set consists of several separate APIs to reduce the effort on enabling the external FLASH support. This section provides an introduction to the ROM based FLEXSPI FLASH Driver APIs.

FLEXSPI FLASH Driver APIs provide functions to simplify the Flash programming work in user application.

FLEXSPI FLASH Driver API Features:

- API to recognize an external Serial FLASH device.
- APIs to program data to or read data from an external Serial FLASH device.
- APIs to partially or fully erase an external Serial Flash device.
- Support for most Serial NOR Flash devices, including but not limited to: in the market are supported
 - Serial NOR Flash devices which support JESD216 standard or later revision
 - HyperFLASH devices (HyperFLASH is a HyperBus compliant Flash device from Cypress or ISSI).
 - Micron Xcella Flash devices (Xcella is a trademark of Xcella consortium).
 - Macronix OctaFlash devices (OctaFlash is a trademark of Macronix Octal Flash).
 - Adesto EcoXiP Flash devices (EcoXiP is a trademark of Adesto Octal Flash).

Table 144. FLEXSPI FLASH Driver API List

Function prototype	API Description
<code>status_t flexspi_nor_flash_init(uint32_t instance, flexspi_nor_config_t *config);</code>	Initialize the FLEXSPI controller
<code>status_t flexspi_nor_flash_page_program(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const uint32_t *src);</code>	Program data to specified Flash Address
<code>status_t flexspi_nor_flash_erase_all(uint32_t instance, flexspi_nor_config_t *config);</code>	Erase the whole Flash
<code>status_t flexspi_nor_flash_erase(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);</code>	Erase the specified Flash region
<code>status_t flexspi_nor_flash_erase_sector(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);</code>	Erase a specified Flash Sector
<code>status_t flexspi_nor_flash_erase_block(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);</code>	Erase a specified Flash block
<code>status_t flexspi_nor_flash_read(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start, uint32_t bytes)</code>	Read data from a start address of a fixed length
<code>void flexspi_clock_config(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode)</code>	Config the flexspi clock requency and data samples mode
<code>status_t flexspi_nor_set_clock_source(uint32_t clockSource)</code>	Select the flexspi clock source
<code>status_t flexspi_nor_get_config(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t *option);</code>	Get the Flash configuration from an external FLASH

NOTE

Flash Block erase is not supported for FlexSPI memory, when flash-erase-region command is issued from blhost.

18.7.5.1 flexspi_nor_flash_init

This routine initializes the FLEXSPI controller based on the parameter specified in argument 0, argument 1.

Table 145. flexspi_nor_flash_init API

Routine	flexspi_nor_flash_init
Prototype	status_t flexspi_nor_flash_init(uint32_t instance, flexspi_nor_config_t *config)
Input Parameter	Parameter0: FLEXSPI controller instance, only support 0 Parameter1: The Flash configuration block
Result	Error code: 0 = no error
Description	Configures the FLEXSPI controller with the arguments pointed by parameter1.

18.7.5.1.1 Parameter0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 1

18.7.5.1.2 Parameter1: The Flash configuration block buffer

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.2 flexspi_nor_flash_page_program**Table 146. flexspi_nor_flash_page_program API**

Routine	flexspi_nor_flash_page_program
Prototype	status_t flexspi_nor_flash_page_program(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const uint32_t *src);
Input Parameter	Parameter 0: The FLEXSPI controller instance 0 Parameter 1: The Flash Configuration Block Parameter 2: The destination address to be programmed Parameter 3: The data to be programmed
Result	Error code: 0 = no error
Description	Program data to specified destination address

18.7.5.2.1 Parameter0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 1.

18.7.5.2.2 Parameter 1: The Flash configuration block buffer

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size.

18.7.5.2.3 Parameter 2: The destination address to be programmed

This parameter specifies the destination address to be programmed. This parameter must be page aligned.

18.7.5.2.4 Parameter 3: The data to be programmed

This parameter points to the buffer which hold the data to be programmed into the destination address specified by parameter 1.

18.7.5.3 flexspi_nor_flash_erase_all

Table 147. flexspi_nor_flash_erase_all API

Routine	flexspi_nor_flash_erase_all
Prototype	status_t flexspi_nor_flash_erase_all(uint32_t instance, flexspi_nor_config_t *config);
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0 Parameter 1: The flash configuration block
Result	Error code: 0 = no error
Description	Erase the entire external Flash

18.7.5.3.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.3.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.4 flexspi_nor_flash_erase

Table 148. flexspi_nor_flash_erase API

Routine	flexspi_nor_flash_erase
Prototype	status_t flexspi_nor_flash_erase(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The start address to be erased Parameter 3: The length to be erased
Result	Error code: 0= no error
Description	Erase the specified Flash region

18.7.5.4.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.4.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.4.3 Parameter 2: The start address to be erased

This parameter specifies the start address for the region to be erased.

18.7.5.4.4 Parameter 3 The length to be erased

This parameter indicates the length to be erased via this API.

18.7.5.5 flexspi_nor_flash_erase_sector

Table 149. flexspi_nor_flash_erase_sector API

Routine	flexspi_nor_flash_erase_sector
Prototype	status_t flexspi_nor_flash_erase_sector(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The address of the sector to be erased
Result	Error code: 0= no error
Description	Erase one specified sector

18.7.5.5.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.5.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#). details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.5.3 Parameter 2: The address of the sector to be erased

The parameter specifies the address of the sector to be erased.

18.7.5.6 flexspi_nor_flash_erase_block

Table 150. flexspi_nor_flash_erase_block API

Routine	flexspi_nor_flash_erase_block
Routine	flexspi_nor_flash_erase_block
Prototype	status_t flexspi_nor_flash_erase_block(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0 Parameter 1: The flash configuration block Parameter 2: The address of the block to be erased

Table continues on the next page...

Table 150. flexspi_nor_flash_erase_block API (continued)

Routine	flexspi_nor_flash_erase_block
Result	Error code: 0= no error
Description	Erase one specified block

18.7.5.6.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.6.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.6.3 Parameter 2: The address of the block to be erased

The parameter specifies the address of the block to be erased.

18.7.5.7 flexspi_nor_flash_read

Table 151. flexspi_nor_flash_read API

Routine	flexspi_nor_flash_read
Prototype	status_t flexspi_nor_flash_read(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start, uint32_t bytes)
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0
	Parameter 1: The Flash Configuration Block
	Parameter 2: Buffer address used to store the read data
	Parameter 3: Read address
	Parameter 4: Read size
Result	Error code: 0= no error
Description	Use to read data from the flexspi nor flash

18.7.5.7.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.7.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.7.3 Parameter 2: Buffer address used to store the read data

Buffer address which is used to copy the data read from the flexspi Nor flash.

18.7.5.7.4 Parameter 3: Read Address

Indicate the address we are reading from.

18.7.5.7.5 Parameter 3: Read Size

Indicate the read data size.

18.7.5.8 flexspi_clock_config

Table 152. flexspi_clock_config_API

Routine	flexspi_nor_flash_read
Prototype	void flexspi_clock_config(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode)
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0
	Parameter 1: FLEXSPI interface clock frequency selection
	Parameter 2: FLEXSPI controller data sample mode
Result	Error code: 0= no error
Description	Use to config the flexspi interface clock frequency and data sample mode

18.7.5.8.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.8.2 Parameter 1: FLEXSPI interface clock frequency selection

This is used to selection the flexspi spi work clock below are possible options.

- kFlexSpiSerialClk_30MHz = 1,
- kFlexSpiSerialClk_50MHz = 2,
- kFlexSpiSerialClk_60MHz = 3,
- kFlexSpiSerialClk_80MHz = 4,
- kFlexSpiSerialClk_100MHz = 5,
- kFlexSpiSerialClk_120MHz = 6,
- kFlexSpiSerialClk_133MHz = 7,
- kFlexSpiSerialClk_166MHz = 8,

18.7.5.8.3 Parameter 2: FLEXSPI controller data sample mode

This is used to selection the flexspi interface data sample mode, below are possible options.

kFlexSpiClk_SDR, //!< Clock configure for SDR mode

kFlexSpiClk_DDR, //!< Clock configurat for DDR mode

18.7.5.9 flexspi_nor_set_clock_source

Table 153. flexspi_nor_set_clock_source

Routine	flexspi_nor_get_config
Prototype	status_t flexspi_nor_set_clock_source(uint32_t clockSource)
Input Parameter	Parameter 0: Clock source for flexspi interface
Result	Error code: 0= no error
Description	Only select the clock source for the flexspi interface, do not other configuration

18.7.5.9.1 Parameter 0: Clock source selection for flexspi interface

Below are the clock selection for flexspi interface.

```
kFlexspiClockSrc_MainClk = 0,
kFlexspiClockSrc_MainPIIClk,
kFlexspiClockSrc_Aux0PIIClk,
kFlexspiClockSrc_48M_Clk,
kFlexspiClockSrc_Aux1PIIClk,
```

18.7.5.10 flexspi_nor_get_config

Table 154. flexspi_nor_get_config API

Routine	flexspi_nor_get_config
Prototype	status_t flexspi_nor_get_config(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t *option);
Input Parameter	Parameter 0: FLEXSPI controller instance, only support 0 Parameter 1: The Flash Configuration Block Parameter 2: The Flash Configuration Option block
Result	Error code: 0= no error
Description	Probe the presence of an Flash device and generate the Flash configuration block based on the parameters read from the Flash device

18.7.5.10.1 Parameter 0: FLEXSPI controller instance

This the instance number of FLEXSPI, only support 0.

18.7.5.10.2 Parameter 1: The Flash configuration block

This parameter points to the flash configuration block(FCB) which consists of arguments related to an external Flash device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#).

18.7.5.10.3 Parameter 2: The Flash Configuration Option Block

This parameter provides a simplified option for the FlexSPI driver to probe the Flash and get the Flash parameters later. See the details of the flexspi_nor_config_t in [FLEXSPI FLASH Driver APIs](#). definitions of this block as below.

```
typedef struct _serial_nor_config_option
```

```

{
    union
    {
        struct
        {
            uint32_t max_freq : 4;          //!< Maximum supported Frequency
            uint32_t misc_mode : 4;          //!< miscellaneous mode
            uint32_t quad_mode_setting : 4;  //!< Quad mode setting
            uint32_t cmd_pads : 4;           //!< Command pads
            uint32_t query_pads : 4;         //!< SFDP read pads
            uint32_t device_type : 4;        //!< Device type
            uint32_t option_size : 4;        //!< Option size, in terms of uint32_t, size =
(option_size + 1)
            uint32_t tag : 4;                //!< Tag, must be 0x0C
        } B;
        uint32_t U;
    } option0;
    union
    {
        struct
        {
            uint32_t dummy_cycles : 8;      //!< Dummy cycles before read
            uint32_t status_override : 8;    //!< Override status register value during device
mode configuration
            uint32_t pinmux_group : 4;       //!< The pinmux group selection
            uint32_t dqs_pinmux_group : 4;   //!< The DQS Pinmux Group Selection
            uint32_t drive_strength : 4;     //!< The Drive Strength of FlexSPI Pads
            uint32_t flash_connection : 4;   //!< Flash connection option: 0 - Single Flash connected
to port A, 1 - Parallel mode, 2 - Single Flash connected to Port B
        } B;
        uint32_t U;
    } option1;
} serial_nor_config_option_t;

```

The detailed info of serial_nor_config_option_t structure is shown in below table.

Table 155. . serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero See option1 definition for more details.

Table 156. . Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 - 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 2 - 4 3 - 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configure 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command Note: This field will be effective only if device is compliant with JESD216 only (9 longword SFDP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only)

Table continues on the next page...

Table 156. . Option0 definition (continued)

		Note: Experimental feature, do not use in products, keep it as 0
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting Others – See

Table 157. Option1 Definition

Field	Bits	Description
flash_connection	31:28:00	Dummy cycles for a read command
		0 - Use detected dummy cycle
		Others - dummy cycles provided in flash datasheet
drive_strength	27:24:00	Override status register value during device mode configuration
dqs_pinmux_group	23:20	The pinmux group selection
pinmux_group	19:16	The DQS Pinmux Group Selection
status_override	15:08	The Drive Strength of FlexSPI Pads
dummy_cycles	7:00	Flash connection option: 0 - Single Flash connected to port A, 1 -Parallel mode, 2 - Single Flash connected to Port B

Typical Option configurations are as below:

- QUAD NOR - Quad SDR Read: option0 = 0xc0000004 (80MHz)
- QUAD NOR - Quad DDR Read: option0 = 0xc0100003 (60MHz)
- HyperFLASH 1V8: option0 = 0xc0233004 (80MHz)
- HyperFLASH 3V0: option0 = 0xc0333004 (80MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433004(80MHz)
- Micron Octal DDR: option0=0xc0600004 (80MHz)
- Micron OPI DDR: option0=0xc0603004 (80MHz), SPI->OPI DDR
- Micron OPI DDR (DDR read enabled by default): option0 = 0xc0633004 (80MHz)
- Adesto OPI DDR: option0=0xc0803004(80MHz)

18.7.5.11 Possible Return and Error codes for Flash driver APIs

Table 158. Return and Error codes for Flash APIs

Return code	Error Code	Description
0	kStatus_Success	The operation is successful
1	kStatus_Fail	The operation fails
4	kStatus_Invalidargument	Parameter 0 or the data in the FCB pointed by parameter 0 is invalid
6000	kStatus_FLEXSPI_SequenceExecutionTimeout	The FLEXSPI Sequence Execution timeout
6001	kStatus_FLEXSPI_InvalidSequence	The FLEXSPI LUT sequence invalid
6002	kStatus_FLEXSPI_DeviceTimeout	The FLEXSPI device timeout
20100	kStatus_FLEXSPINOR_ProgramFail	Page programming failure
20101	kStatus_FLEXSPINOR_EraseSectorFail	Sector Erase failure
20102	kStatus_FLEXSPINOR_EraseAllFail	Chip Erase failure
20103	kStatus_FLEXSPINOR_WaitTimeout	The execution time out
20104	kStatus_FlexSPINOR_NotSupported	Page size overflow
20105	kStatus_FlexSPINOR_WriteAlignmentError	Address alignment error
20106	kStatus_FlexSPINOR_CommandFailure	Erase/Program Verify Error
20107	kStatus_FlexSPINOR_SFDP_NotFound	Timeout occurs during the API call
20108	kStatus_FLEXSPINOR_Unsupported_SFDP_Version	Unrecognized SFDP version
20109	kStatus_FLEXSPINOR_Flash_NotFound	Flash detection failure
20110	kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	DDR Read dummy probe failure

18.7.6 KB APIs

The APIs are used in the application to update the firmware of the device by calling the kbapis to process the SB file in the secure state. Through the KB APIs, user can update the firmware of the device including the following features:

- Erase the flash region
- Program data into the flash region
- Efuse program
- Image load into SRAM and boot

Before running the application with the KB APIs, the prepared configuration files should be as follow:

- The SB file should be generated by user to update the firmware
- The key-store/OTP master key (including the SB key code and other keycodes) should be used for processing the SB file
- The prepared files are the same as the files used for executing the receive-sb-file command by the blhost tool. These prepared configuration files are used for authenticating the signed image and the SB-file to be processed.

NOTE

The Secure Binary (SB) image format is a command-based firmware update image. One can imagine the SB2 file as a script (commands and data), for which the ROM is the interpreter. The RT500 silicon supports version 2.1 of the SB image format. Version 2.1 updates the encryption scheme to use modern algorithms, adds support for signed images and it also makes the usage of digital signature mandatory. SB 2.1 file is always digitally signed and encrypted, and the ROM verifies the digital signature before decrypting the content. The SB 2.1 file format also uses AES encryption for confidentiality and HMAC for extending trust from the signed part of the SB file to the command and data part of the SB file. These two keys (AES decrypt key and HMAC key) are wrapped in the RFC3394 key blob, for which the key wrapping key is the SBKEK key. The layout of an SB 2.1 file is shown in the elftosb tool User's Guide. The elftosb tool is the NXP image signing and SB file creating tool for Windows/Linux/MAC. In this chapter, we provide introductory description of the SB file format components, while for the exact description, one shall refer to the elftosb tool User's Guide. See "Secure ROM API" section of "Secure Boot ROM" chapter.

18.7.6.1 KB API prototypes

The bootloader KB API prototypes are as follow:

```
typedef struct _kb_interface
{
    /*!< Initialize the API. */
    status_t (*kb_init)(kb_session_ref_t **session, const kb_options_t *options);
    /*! De-initialize the API. */
    status_t (*kb_deinit)(kb_session_ref_t *session);
    /*! Execute the Bootloader API. */
    status_t (*kb_execute)(kb_session_ref_t *session, const uint8_t *data, uint32_t dataLength);
} kb_interface_t;
```

The kb_session_ref_t is the current ROM context pointer and is as follow:

```
typedef struct _kb_opaque_session_ref
{
    kb_options_t context; /*!< options context provided by the application
    bool cau3Initialized; /*!< cau3 Initialized flag,
    memory_map_entry_t *memoryMap; /*!< new memory map defined by the application based
    /*!< on the info in loadSB struct
    } kb_session_ref_t;
```

The kb_options_t is defined as follow:

```
typedef struct _kb_options
{
    uint32_t version; /*!< Should be set to #kKbootApiVersion.
    uint8_t *buffer; /*!< Caller-provided buffer used by Kboot, must be word-aligned.
    uint32_t bufferLength; /*!< Caller-provided buffer length, should be larger than size of
    /*!< kb_session_ref_t
    kb_operation_t op; /*!< The kb API purpose, should be set to # kRomLoadImage.
```

```

union
{
    //! Settings for #kKbootAuthenticate operation.
    kb_authenticate_t authenticate;
    //! Settings for #kKbootLoadSB operation.
    kb_load_sb_t loadSB;
};
kb_security_options_t security_options;
} kb_options_t;
typedef struct _kb_security_options
{
    keystore_storage_selection_t selection; // inform sbloader where is the keystore stored
    // config params used for config the NV storage interface: QSPI,MMC
    // For example: QSPI HYPERFLASH option[0]=0xc0233001 option[1] not used
    //               QSPI QUAD FLASH option[0]=0xc0000001 option[1] not used
    //               MMC CARD option[0]=0xc0000001 option[1]=0
    uint32_t storage_config_option[2];
    uint32_t reserved;
} kb_security_options_t;
The kb_load_sb_t is defined as follow:
typedef struct _kb_load_sb
{
    uint32_t profile; //!< This includes the minimum AES and RSA key sizes.
    uint32_t minBuildNumber; //!< An optional field that can be used to prevent version
    rollback.
    uint32_t overrideSBBootSectionID; //!< Lets the caller override the default section of the SB
    file.
    uint32_t *userSBKEK; //!< Lets a user provide their own AES-256 key for unwrapping keys in
    //!< an SB file.
    uint32_t regionCount; //!< The memory regions number which used for load SB file.
    const kb_region_t *regions; //!< points to an array of memory regions that the SB file
    being
    //!< loaded is allowed to access.
} kb_load_sb_t;

```

18.7.6.1.1 kb_init

This API is used for initializing the ROM bootloader API based on the options provided by the application. It must be called before calling other ROM KB APIs.

Parameter	Description
session	Pointer to kb_session_ref_t data structure in memory to store current ROM context
options	Pointer to kb_options_t data structure that is defined by the application.

Prototype

```
status_t (*kb_init)(kb_session_ref_t **session, const kb_options_t *options);
```

Example:

```
#define BOOTLOADER_TREE_LOCATION (0x1302f000)
#define USER_BUFFER_LENGTH 0x4000
bootloader_tree_t *romApiTree = (bootloader_tree_t *)BOOTLOADER_TREE_LOCATION;
kb_interface_t *s_romAPIInterface;
s_romAPIInterface = romApiTree->kbApi;
kb_session_ref_t *context;
const kb_options_t options = {
    .version = kRomApiVersion,
    .buffer = user_buf,
    .bufferLength = USER_BUFFER_LENGTH,
    .op = kRomLoadImage,
    {
        .loadSB =
        {
            .profile = 0,
            .minBuildNumber = 1,
            .overrideSBBootSectionID = 1,
            .userSBKEK = NULL,
            .regionCount = 0,
            .regions = NULL,
        },
    },
    .security_options.selection = kQspiFlash_KeyStore,
    .security_options.storage_config_option[0] = 0xc0000001,
    .security_options.storage_config_option[1] = 0,
};
status = s_romAPIInterface->kb_init(&context, &options);
```

18.7.6.1.2 kb_execute

This API is to transfer sequential chunks of data from the boot image (SB file) to be processed. The ROM will perform the selected operation on this data and return running status.

Prototype

```
status_t (*kb_execute)(kb_session_ref_t *session, const uint8_t *data, uint32_t dataLength);
```

Parameter	Description
session	Pointer to kb_session_ref_t data structure in memory to store current ROM context
data	Pointer to the data array that store the SB-file data
dataLength	The data length of the data pointer array that stores SB-file

Example:

```
get_chunk_from_SB_file(uint8_t * chunk_processing_buffer, uint8_t * SBfile_incoming_chunk,
uint32_t chunk_size); //load the SB file chunk-by-chunk into chunk_processing_buffer
status = s_romAPIInterface->kb_execute(context, & chunk_processing_buffer, chunk_size);
```

18.7.6.1.3 kb_deinit

This API is used for cleaning the ROM bootloader API context. After calling this API, the context can be reused for another operation by calling kb_init() again.

Prototype

```
status_t (*kb_deinit)(kb_session_ref_t *session);
```

Parameter	Description
session	Pointer to kb_session_ref_t data structure in memory to store current ROM context

```
status = s_romAPIInterface->kb_deinit(context);
```

Table 159. IAP bootloader API status code

Status	Code	Description
kStatus_RomApiExecuteCompleted	0	ROM successfully process the whole sb file/boot image
kStatus_RomApiNeedMoreData	10801	ROM needs more data to continue processing the boot image
kStatus_RomApiBufferSizeNotEnough	10802	The user buffer is not enough for using by Kboot during the execution of the operation
kStatus_RomApiInvalidBuffer	10803	The user buffer is not ok for IAP or authentication

Chapter 19

Secure Boot ROM

19.1 Introduction

19.1.1 Secure Boot

Secure boot provides guarantee that unauthorized code cannot be executed on a given product. It involves ROM of the device, always executing when coming out of reset. The ROM will then examine the first user executable image resident in external boot media (NOR flash, MMC) or in System RAM (downloaded to RAM by serial boot interface) to determine the authenticity of that code. If the code is authentic, then control is transferred to it. This establishes a chain of trusted code from the ROM to the user boot code. This chain can be further extended, as described below. The method used in this architecture to verify the authenticity of the boot code is to verify RSA signatures over the code. The boot code is signed with RSA private keys. The corresponding RSA public keys used for signature verification are contained in X.509 certificates that are contained in the signed image. Device supports up to four Root of Trust keys.

Device could be configured to boot plain images during development. In such case ROM does not check image to be booted, or perform only CRC32 checking, depending on configuration.

19.1.2 Secure firmware update

If firmware updates are to be performed in the field when secure boot is enabled, then a secure firmware update mechanism is preferred. Otherwise inauthentic firmware may be written to the device, causing it to not boot. In the most basic sense, secure firmware update simply performs an authentication of the new firmware prior to committing it to memory. In this case, the chain of trust is extended from the old, currently executing, code to the new code.

Another use case for secure firmware update is to hide the application binary code during transit over public media such as the web. This is accomplished by encrypting the firmware update image. As the new firmware is written into device memory, it is decrypted.

In this architecture, both cases of secure firmware update are supported. The SB file format is encrypted and digitally signed. SB file can be loaded via secure interfaces such as USB, UART etc. or can be provided to ROM API as complete binary

19.1.3 Extending the chain of trust

Once secure boot has transferred CPU control to user code, that code may need to load additional pieces of code. This establishes another link in the chain of trust. The process can continue for any many nested sub-modules are required, with each parent code module authenticating the chain. Another use case is to authenticate boot code for one or more secondary CPU cores prior to releasing them from reset.

The loader API is used from customer code to verify signatures on the additional code images. Using the API to verify signatures gives complete control to the customer code over what additional code must be signed and how that code is organized in memory..

19.1.4 Miscellaneous functions

ROM provides support for various additional security related functionality. The main are:

- Support of DICE (Device Identifier Composition Engine)
- Support for load of TrustZone-M pre-configuration during ROM secure boot
- Support of booting from encrypted external SPI memory using OTFAD peripheral module
- Debug Authentication
- Device provisioning (ROM embedded support for the initial secure provisioning of the boot keys). Enables NXP device authenticity feature.
- Initial boot state, as specified in ARM Platform Security Architecture Security Model 1.0.

19.1.5 Boot flow diagram

Booting of the device is controlled by configuration written in OTP (One Time Programmable) fuses and based on ISP pin setting.

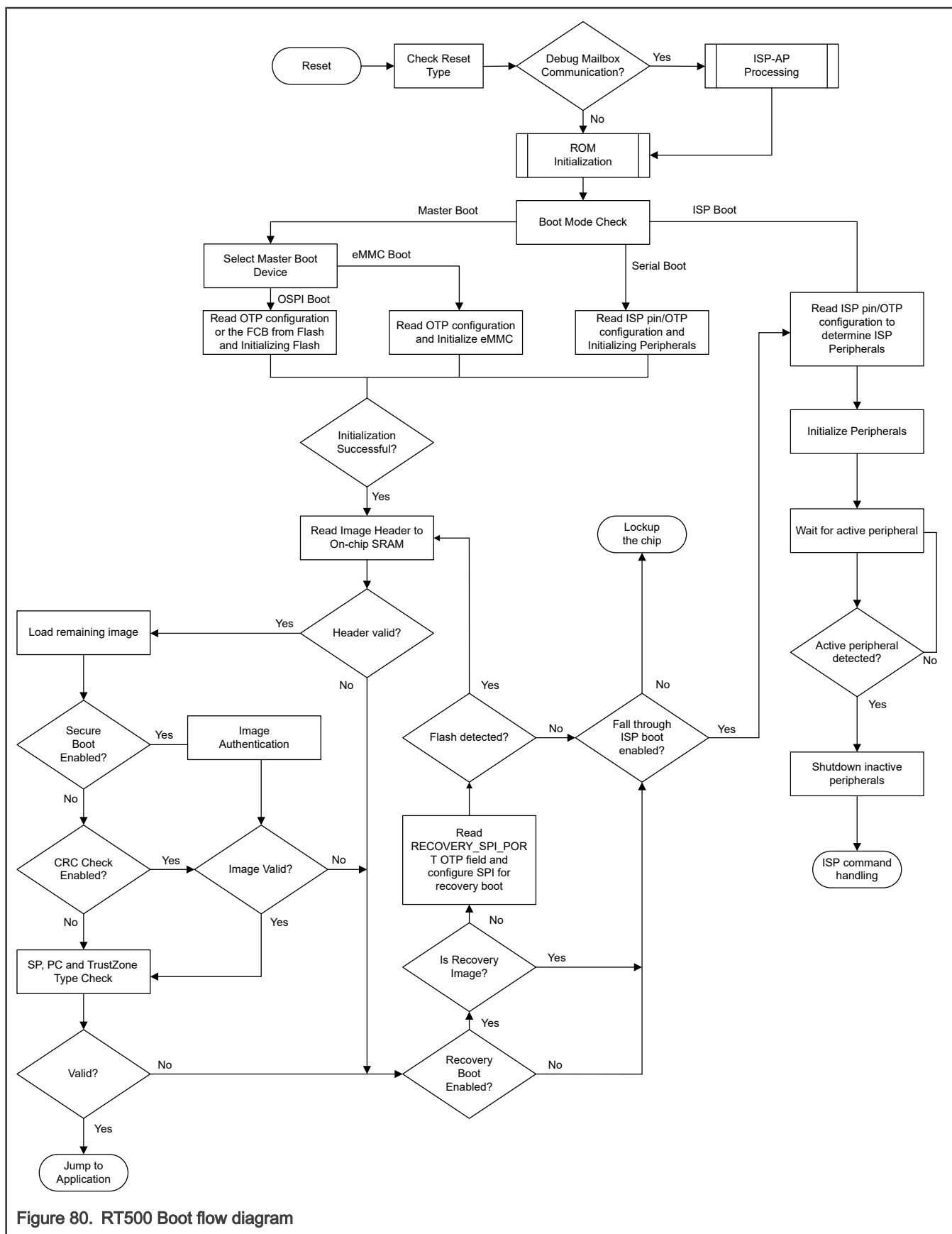


Table 160. ISP commands

Name	Description
GetProperty	Get the current value of a property
ReceiveSBFile	Receive and process an SB-format programming image
Reset	Reset the chip
SetProperty	Attempt to modify a writable property. Used for setting nHostIRQ pin
KeyProvision	PUF key provision commands - enroll, set key, set user key, read key store

19.2 Data structures

19.2.1 Overview

RT500 stores its configuration for the boot ROM in OTP (One Time Programmable) fuses. According to bootloader configuration, OTP can be also used for storing the keys.

19.2.2 OTP key storage

RT500 bootloader can be configured to store keys in OTP fuses. It is achieved when USE_PUF bit field in BOOT_CFG[5] is set to zero. RT500 has limited OTP storage and has device unique scrambler feature associated with single 256 bit key. Hence all other keys needed for other security features are derived from OTP_MASTER_KEY.

Table 161. OTP Key store summary

OTP Word	Shadow Register Offset	Size(bytes)	Name	Description
94	0x178	4	UDS_RANDOM_DATA	Unique device secret data, used to derive UDS key. Programmed by NXP.
107	0x1AC	4	KEY_SCRAMBLE_SEED	Key scramble data used as input to scramble master key stored in OTP. Please see Note 1 and Note 2 .
108	0x1B0	16	OTFAD_KEK_SEED	Used to derive OTFAD key blob unwrap key.
112	0x1C0	32	OTP_MASTER_KEY	Single 256-bit key used to derive other functional boot keys.
344	N/A	64	DEVICE_CERT_SIGNATURE	Device certificate signature (Elliptic curve P-256)

NOTE

To use key scrambler feature, the following fuse programming order shall be used:

- Program the KEY_SCRAMBLE_SEED OTP word
- Program the OTP_MASTER_KEY key words in 8 sequential OTP writes (from 112, 113. ... 119).

NOTE

During development, following features are enabled to test different boot scenarios:

- If KEY_SCRAMBLE_SEED (OTP word 107) is not programmed, then the OTP_MASTER_KEY (Words 112 to 119) key is stored and retrieved in plain format.
 - If KEY_SCRAMBLE_SEED (fuse and shadow register) and OTP_MASTER_KEY (fuse and shadow register) are left un-programmed. Then zero key is passed to AES engine as master key.
 - If KEY_SCRAMBLE_SEED (fuse and shadow register) are left blank, then shadow registers of OTP_MASTER_KEY can be written with a test key which is passed in plain format to AES engine.
- 16 OTFAD_KEK_SEED

Table 162. OTP Key store summary

OTP Word	Shadow Register Offset	Description	OTP Word	Shadow Register Offset	Description
112	0x1C0	OTP_MASTER_KEY[31:0]	116	0x1D0	OTP_MASTER_KEY[159:128]
113	0x1C4	OTP_MASTER_KEY[63:32]	117	0x1D4	OTP_MASTER_KEY[191:160]
114	0x1C8	OTP_MASTER_KEY[95:64]	118	0x1D8	OTP_MASTER_KEY[223:192]
115	0x1CC	OTP_MASTER_KEY[127:96]	119	0x1DC	OTP_MASTER_KEY[255:224]

19.2.3 Boot keys derived from OTP_MASTER_KEY

19.2.3.1 HMAC_KEY

- This key is used for image header authentication in LoadToRam images.
- AES256(OTP_MASTER_KEY, 00000000_00000000_00000000_00000000)

19.2.3.2 ENC_IMAGE_KEY

- Key used to decrypt encrypted LoadToRam images during boot.
-
- This usage key is derived from OTP_MASTER_KEY using following construct AES256(OTP_MASTER_KEY, 01000000_00000000_00000000_00000000_02000000_00000000_00000000_00000000).

19.2.3.3 SB2_KEK

- Key used as key encryption key to handle SB2 file (update capsule)
- This usage key is derived from OTP_MASTER_KEY using following construct AES256(MASTERKEY256, 03000000_00000000_00000000_00000000_04000000_00000000_00000000_00000000)

19.2.3.4 OTFAD_KEK128

- Used by ROM to unwrap key blobs for OTFAD configuration
- AES256_ENCRYPT (OTP_MASTER_KEY, OTFAD_KEK_SEED [127:0])

Table 163. OTFAD_KEK_SEED organization in OTP fuses

OTP Word	Shadow Register Offset	Description
108	0x1B0	OTFAD_KEK_SEED[127:96]
109	0x1B4	OTFAD_KEK_SEED[95:64]
110	0x1B8	OTFAD_KEK_SEED[63:32]
111	0x1BC	OTFAD_KEK_SEED [31:0]

19.2.3.5 UDS_KEY

A 32 bit value OTP word, UDS_RANDOM_DATA is defined to support UDS from OTP. The value is set during manufacturing process. Apart from the value being random a scrambler block is attached to generate final value of UDS key

19.2.4 PUF key storage

When USE_PUF bit field in BOOT_CFG[5] is set to one, bootloader will use keys stored in PUF key store, which can be part of application image or it can be stored in non-volatile memory (QSPI flash, MMC card). The key store data structure is located at address 0x800 of external non-volatile memory or at offset 0x64 of the application image if *Key Store Included* bit is set in image header for Load to Ram nonXip image type.

Table 164. PUF Key code storage area structure

Offset	Bytes	Name	Description
0x0	4	Key Store Header	Marker. A value of 0x95959595 means that Activation code is valid.
0x4	4	PUF Discharge time	Time in milliseconds to wait until PUF SRAM fully discharges. Only effective when PUF Start fails. Set to zero to use default discharge time.
0x8	1,192	Activation Code	Device specific PUF activation code generated by enroll command during key provisioning.
0x4B0	56	SB2_KEK Key Code	Key Code for wrapped SB2_KEK key
0x4E8	56	Encrypted boot image Key Code	Key Code for wrapped USERKEK key
0x520	56	UDS Key Code	Key Code for wrapped UDS key
0x558	56	OTFAD_KEK Key Code	Key Code for wrapped OTFAD_KEK

19.2.5 PUF Keys

19.2.5.1 PUF Key Code format

Table 165. PUF Key code storage area structure

Offset	Bytes	Name	Description

Table continues on the next page...

Table 165. PUF Key code storage area structure (continued)

0x0	4	Key code marker	A value of 0x59595959 means that Key code is valid.
0x4	52	Key code	Key code.Wrapped plaintext key. SRAM PUF device unique key is used as a key wrapping key.

19.2.5.2 Key descriptions

SB2_KEK – Secure Binary Key Encryption Key

- Key used as key encryption key to handle SB2 file (update capsule)
- AES-256 symmetric key
- blhost key type 3
- PUF key index 0

NOTE

Note PUF key index 0 means the key unwraps to dedicated hardware bus, directly connected to AES crypto engine

Encrypted boot image Key

- Key used to decrypt encrypted LoadToRam images during boot
- 256-bit symmetric key
- blhost key type 11
- PUF key index 0

UDS key – Universal Device Secret

- Universal Device Secret for DICE
- HMAC-SHA256 256-bit key
- blhost key type 12
- PUF key index 15

OTFAD_KEK

- Used by ROM to unwrap OTFAD key blobs
- 128-bit symmetric key
- blhost key type 2
- PUF key index 0

19.2.6 Secure boot related configuration fields in OTP fuses

19.2.6.1 BOOT_CFG[0]

Table 166. BOOT_CFG[0] bit fields

OTP Word	Shadow register Offset	Bit(s)	Name	Description
96	0x180	3:0	PRIMARY_BOOT_SRC	See the attached OTP Fuse Map spreadsheet for more details.
		6:4	DEFAULT_ISP_MODE	See the attached OTP Fuse Map spreadsheet for more details.
		7	BOOT_CLK_SPEED	See the attached OTP Fuse Map spreadsheet for more details.
		8	RSA4K_EN	Use 4096 bit RSA keys only for certificate validations. 2'b0: Use RSA2K or higher key length operations to validate certificates 2'b1: Use RSA4K operations to validate certificates
		12:9	RESERVED	Should be filled with zeros.
		14:13	TZM_IMAGE_TYPE	TrustZone-M image mode 2'b00: TZ-M image mode is taken from application image header 2'b01: TZ-M disabled image. ROM will always boot to a non-secure mode and all TZ-M features are disabled. 2'b10: TrustZone-M features are enabled. ROM will always boot to secure mode. 2'b11: TZ-M enabled image with TZ-M preset, ROM will always boot to secure mode, TZ-M pre-configured by data from application image.
		15	PSA_BSTATE_SKIP	If set, ROM skips computation of boot state defined by PSA specification. As part of boot state computation ROM includes OTP words - Shadow register values of 95 to 104 - Fuse values of words 128 to 147
		16	PSA_BSTATE_INC_KEYS	If set, boot state computation includes OTP shadow register values of words 106 to 127.
97	0x180	19:17	REDUNDANT_SPI_PORT	See the attached OTP Fuse Map spreadsheet for more details.
		21:20	SECURE_BOOT_EN	Enable secure boot. 2'b00: Secure boot is disabled. 2'b01: Secure boot is enabled. 2'b10: Secure boot is enabled. 2'b11: Secure boot is enabled.

Table continues on the next page...

Table 166. BOOT_CFG[0] bit fields (continued)

OTP Word	Shadow register Offset	Bit(s)	Name	Description
		22	DICE_INC_OTP	Include non-field updatable OTP Fields in DICE computation. OTP values in shadow registers are used in computation for words 95, 96, 98, 99, 104, 120 - 127, 160 - 335. 2'b0: Do not include OTP Fields in DICE computation 2'b1: Include OTP Fields in DICE computation
		23	SKIP_DICE	Skip computation of Composite Device Identifier (CDI) If set, ROM skips computation of CDI defined in DICE specification. ROM will continue to hide UDS source in OTP and PUF (index 15) before passing control to user code in either case. 2'b1: Skip computation of Composite Device Identifier 2'b0: Do not skip computation of Composite Device Identifier
		31:24	BOOT_FAIL_PIN	See the attached OTP Fuse Map spreadsheet for more details.

19.2.6.2 BOOT_CFG[5]

Table 167. BOOT_CFG[5] bit fields

OTP Word	Shadow register Offset	Bit(s)	Name	Description
106	0x194	3:0	REVOKE_ROOTKEY	Revoke up to 4 root keys. When a bit is set corresponding root key is revoked.
		4	FA_MODE_EN	Enable Fault Analysis mode 2'b0: FA_MODE is enabled 2'b1: FA_MODE is disabled
		6:5	ENABLE_CRC_CHECK	See the attached OTP Fuse Map spreadsheet for more details.
		7	USE_PUF	Use PUF to store AES keys and UDS. 2'b0: AES keys and UDS are read from OTP. 2'b1: AES keys and UDS are read from PUF.
		8	PUF_BLOCK_ENROLL	Block further enrollement of PUF peripheral. When this bit is set, ROM blocks generation of new activation codes.

Table continues on the next page...

Table 167. BOOT_CFG[5] bit fields (continued)

OTP Word	Shadow register Offset	Bit(s)	Name	Description
				2'b0: PUF enroll is allowed 2'b1: PUF enroll is blocked
		9	PUF_BLOCK_SET_KEY	Block PUF set key operation of PUF peripheral. When this bit is set, ROM blocks generation of new key codes. 2'b0: PUF set key operation is allowed 2'b1: PUF set key operation is blocked
		31:10	RESERVED	Should be filled with zeros.

19.2.6.2.1 REVOKE_ROOTKEY

Each of four Root Keys can be revoked. When trying to boot Images that are signed using revoked Root key, they will be rejected during the authentication process and will fail to boot if SEC_BOOT_EN is set to boot only signed images.

Table 168. REVOKE_ROOTKEY field description

REVOKE_ROOTKEY bit field	Description
0	REVOKE_ROOTKEY 0 2'b0: ROOTKEY 0 enabled 2'b1: ROOTKEY 0 is revoked
1	REVOKE_ROOTKEY 1 2'b0: ROOTKEY 1 enabled 2'b1: ROOTKEY 1 is revoked
2	REVOKE_ROOTKEY 2 2'b0: ROOTKEY 2 enabled 2'b1: ROOTKEY 2 is revoked
3	REVOKE_ROOTKEY 3 2'b0: ROOTKEY 3 enabled 2'b1: ROOTKEY 3 is revoked

19.2.6.3 BOOT_CFG[6]

Table 169. BOOT_CFG[6] bit fields

Shadow Register Offset	Bit(s)	Name	Description
0x198	15:0	REVOKE_IMG_KEY	Image keys revocation identifier.
	31:16	RESERVED	Should be filled with zeros.

19.2.6.3.1 REVOKE_IMG_KEY

This value is checked during image authentication process. The x509 serial number field in the image signing certificate is used in the following way: byte 0 shall be 0x3c, byte 1 shall be 0xc3, byte 2 and byte 3 form an unsigned 16-bit integer whose value is compared with the REVOKE_IMG_KEY value in the OTP memory (fuses). On mismatch, the image authentication process will fail. Only 17 revocation IDs are possible. (0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, 0x3F, 0x7F, 0xFF ... 0xFFFF). One bit should be set on every revocation starting from lower bit 0 to 15:

0b0 -> 0b1 -> 0b11 -> 0b111

To implement anti-rollback feature, the image key certificate should be revoked using this field. The in-field FW update process should blow the least unblown bit. During image key certificate validation this field is compared with the certificate content (upper 2 bytes of serial number field in X.509 certificate).

19.2.6.4 OTFAD_CFG

See [OTFAD](#)

19.2.6.5 KEY_SCRAMBLE_SEED

See .

19.2.6.6 OTFAD_KEK_SEED

See .

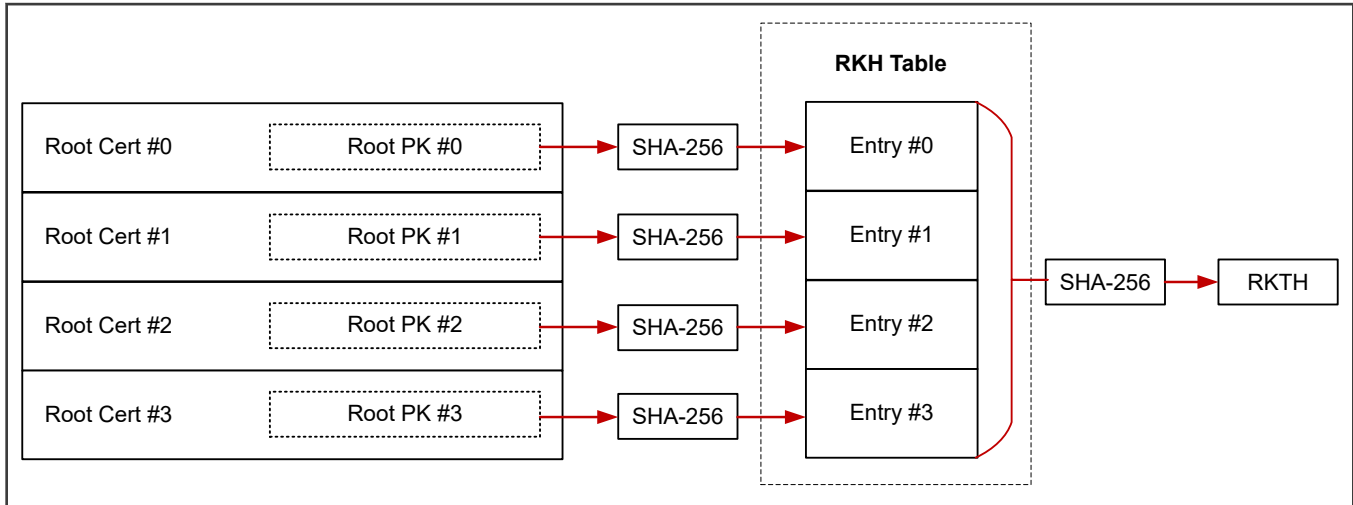
19.2.6.7 OTP_MASTER_KEY

See [OTP key store](#).

19.2.6.8 RKTH

RKTH is 32-byte SHA-256 hash of SHA-256 hashes of up to four root public keys. Multiple root public keys are supported to allow for key revocation.

The structure of this table is shown here:



Each entry in the table is a SHA-256 computed over the concatenation of an RSA public key's modulus and exponent (modulus || exponent). Both modulus and exponent must be in big endian byte order, with the minimum number of bytes required to represent the value. For instance, an exponent of 65537 would be represented by a 3-byte value of [01 00 01], while an exponent of 3 would be represented by a single byte of that value. The entire RKH table is itself hashed with SHA-256. This final hash is then stored in the RKTH field in OTP.

For i in 0..3:

Let $M_i = \text{BE}(\text{Modulus } i)$ Let $E_i = \text{BE}(\text{Exponent } i)$

Let $\text{RKH}_i = \text{SHA256}(M_i || E_i)$

Let $\text{RKTH} = \text{SHA256}(\text{RKH}_0 || \text{RKH}_1 || \text{RKH}_2 || \text{RKH}_3)$

The number of hashes of keys in the RKH table must be from at least 1 through a maximum of 4. Unused table entries must be set to all 0 bytes. When searching the RKH table for a key's hash, the loader will stop at the first entry that is all zeroes.

The extra root public keys and root certificates must be created in advance and would be held in reserve in case a public key had to be revoked. The customer is responsible for implementing the mechanism to determine whether key needs to be revoked, and then set the appropriate RKTH_REVOKE bit(s). This would usually be done through an authenticated connection with a server during a firmware update.

Note that only one of the root certificates whose keys are listed in the RKH table may be included in the certificate table at a time.

Table 170. RKTH layout in OTP

OTP Word	Shadow Register Offset	Description
120	0x1E0	RKTH [255:224]
121	0x1E4	RKTH [223:192]
122	0x1E8	RKTH [191:160]
123	0x1EC	RKTH [159:128]
124	0x1F0	RKTH [127:96]
125	0x1F4	RKTH [95:64]
126	0x1F8	RKTH [63:32]
127	0x1FC	RKTH [31:0]

19.2.6.9 FW_VERSION

Non-trusted firmware version counter (0 - 255). Optionally used during SB2 file loading. The value written in this configuration word must be always higher or equal to non-secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise if the version check command is included in the SB2 file, the SB2 file load will be rejected.

Table 171. NT_FW_VER layout in OTP

OTP Word	Description
128	NT_FW_VER [15:0]
129	NT_FW_VER [31:16]
130	NT_FW_VER [47:32]
131	NT_FW_VER [63:48]
132	NT_FW_VER [79:64]
133	NT_FW_VER [95:80]
134	NT_FW_VER [111:96]
135	NT_FW_VER [127:112]
136	NT_FW_VER [143:128]
137	NT_FW_VER [159:144]
138	NT_FW_VER [175:160]
139	NT_FW_VER [191:176]
140	NT_FW_VER [207:192]
141	NT_FW_VER [223:208]
142	NT_FW_VER [239:224]
143	NT_FW_VER [255:240]

19.2.6.10 TZ_FW_VERSION

Trusted firmware version counter (0 - 63). Optionally used during SB2 file loading. The value written in this configuration word must be always higher or equal to secure FW version specified in elftosb .bd file used for creating SB2 file. Otherwise if the check fw version command is included in the SB2 file, the SB2 file load will be rejected.

Table 172. TZ_FW_VERSION layout in OTP

OTP Word	Description
144	TZ_FW_VER [15:0]
145	TZ_FW_VER [31:16]

Table continues on the next page...

Table 172. TZ_FW_VERSION layout in OTP (continued)

OTP Word	Description
146	TZ_FW_VER [47:32]
147	TZ_FW_VER [63:48]

19.2.7 OTP shadow write lock

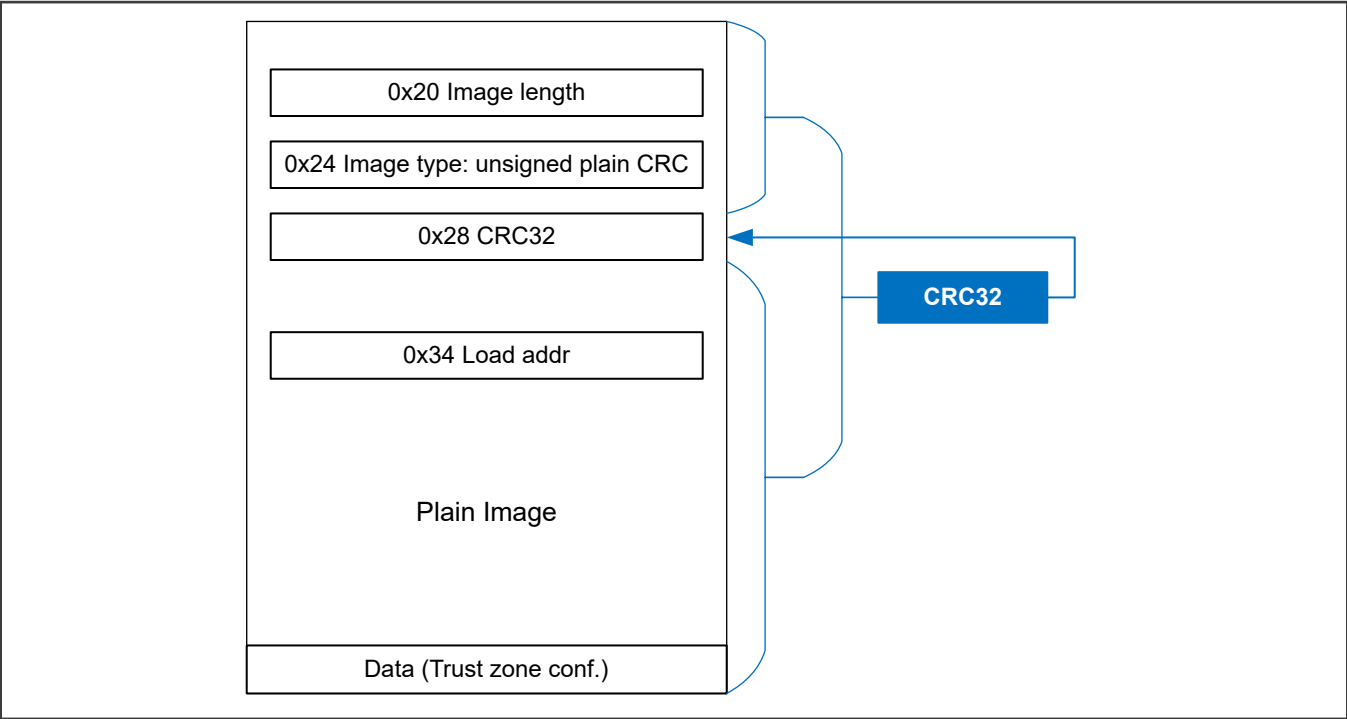
To prevent reading of sensitive information and further modification of values stored in shadow registers, the read or write access to these registers can be permanently disabled by programming appropriate bit fields in CUST_WR_RD_LOCK0 fuse. See chapter [The role of OTP shadow registers during device development lifecycle](#), for more details about shadow register usage.

19.2.7.1 CUST_WR_RD_LOCK0

19.3 Plain image structure

Unsigned Plain CRC images are supported by non-Secure versions of the chip, as well as Secure versions during development life-cycle state of S parts (RT5xxS).

The structure of unsigned CRC images is shown here:



NOTE

When Image Type is 0x0, CRC32 checking is bypassed. Such an image can be used as XiP plain image during development.

Table 173. Image Type Word

Bit	Name	Description
31-16	Reserved	

Table continues on the next page...

Table 173. Image Type Word (continued)

Bit	Name	Description
15	Reserved	0: Keep zero for compatibility with secure images.
14	TZ-M Image Type	0: TZ-M enabled image. The image uses TZ-M.
13	TZ-M Preset	0: No TZ-M peripherals preset.
12	Reserved	0: Keep zero for compatibility with secure images.
11-8	Reserved	
7-0	Image Type	0x0: plain image

19.4 Secure boot images

Table 174. Signed image - Image Type (Word at offset 0x24)

Bit(s)	Name	Description
31:16	Reserved	-
15	Key Store Included	0: The boot image does not have key store. 1: The boot image has key store. (Note - setting this flag has effect only to Load to Ram nonXip image types)
14	TZ-M Image Type	0: TZ-M enabled image. The image uses TZ-M. 1: TZ-M disabled image. The image doesn't use TZ-M.
13	TZ-M Preset	0: No TZ-M peripherals preset. 1: TZ-M peripherals preset. The TZ-M related peripherals are configured by bootloader based on data stored in extended header (after RoT Key hash table)
12	Enable HW user mode keys	0: HW bus keys are available to Secure world only 1: HW bus keys are available to Secure and Non-secure world
11:8	Reserved	-
7:0	Image Type	0x0: plain image 0x1: plain signed image (Load to RAM - nonXip) 0x2: plain image with CRC (Load to RAM - nonXip) 0x3: encrypted signed (Load to RAM - nonXip) 0x4: Xip plain signed 0x5: Xip plain with CRC

Header offset (Word at offset 0x28) - A 32-bit offset from the beginning of the signed image to the certificate block header, called `offsetToCertificateBlockInBytes`, must reside at offset 0x28 from the start of the signed image. An executable code image is expected to start with an NVIC vector table. The word at offset 0x28 is a reserved slot in the vector table. As an example, if an

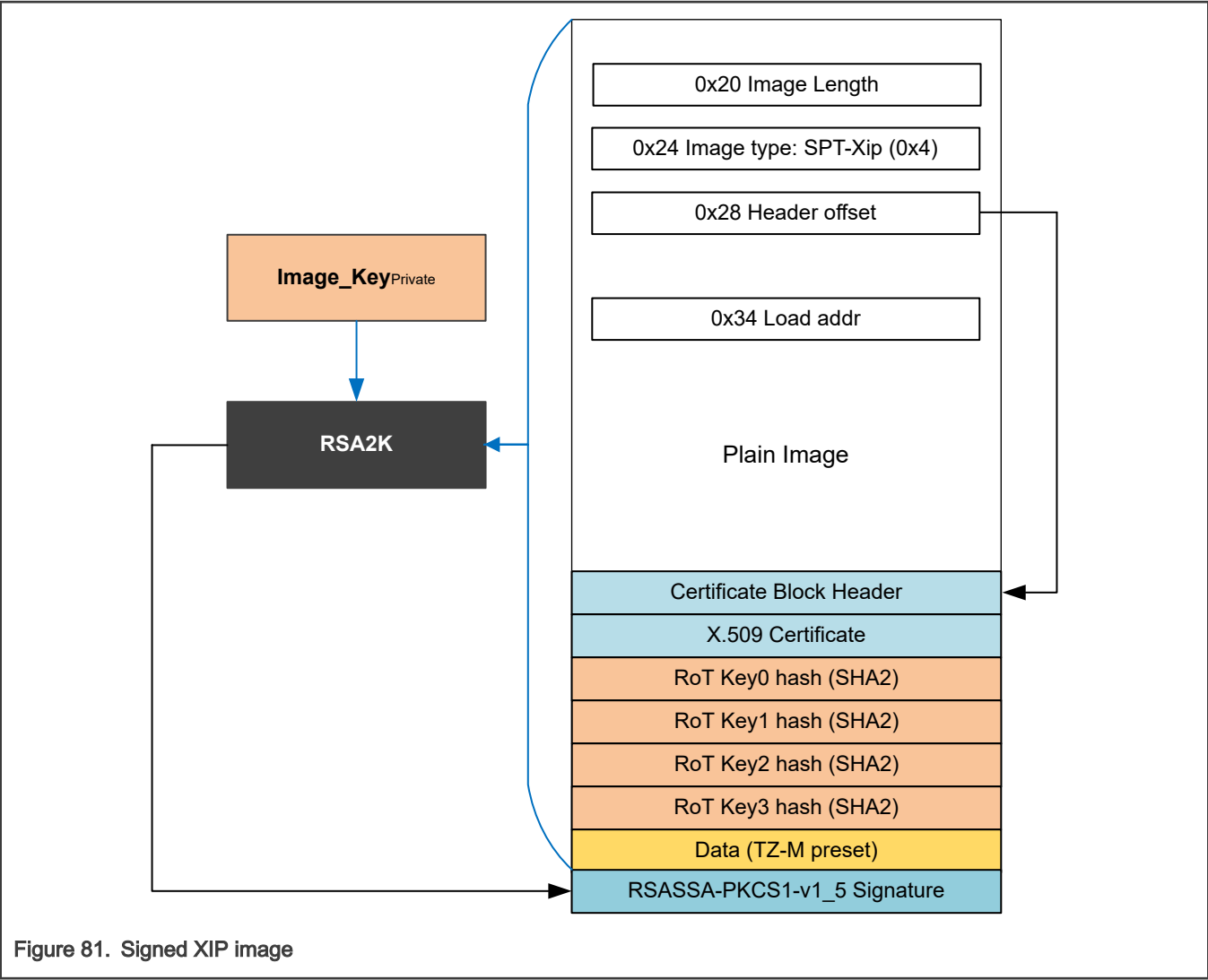
image resides in RAM at a non-zero address (say 0x8000), and its certificate block header is at address 0x24000, then the word at 0x8028 will contain the value 0x1c000.

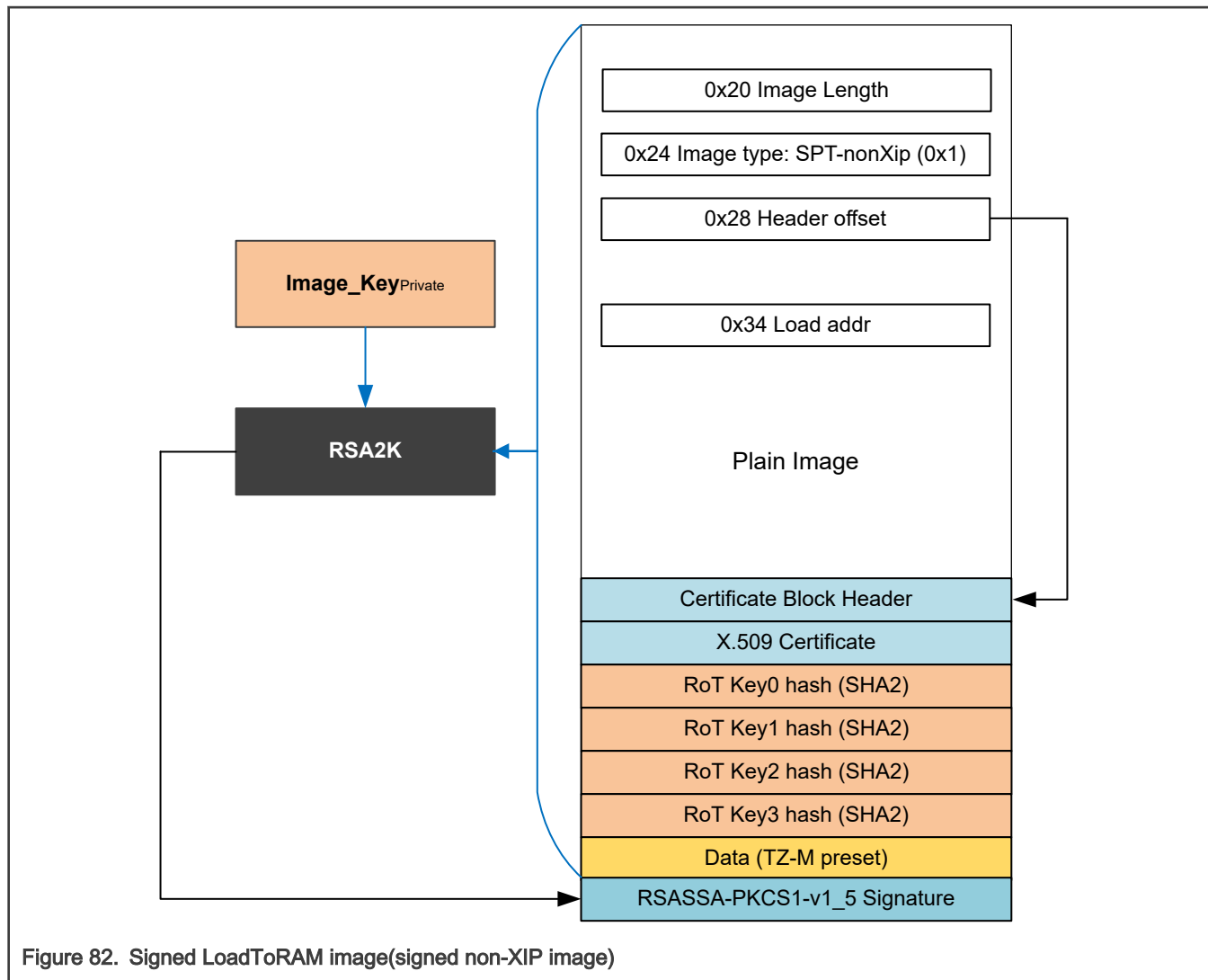
Load addr (Word at offset 0x34) – Image execution address (used for Load To Ram – nonXip images, this is the destination address in RAM for the image)

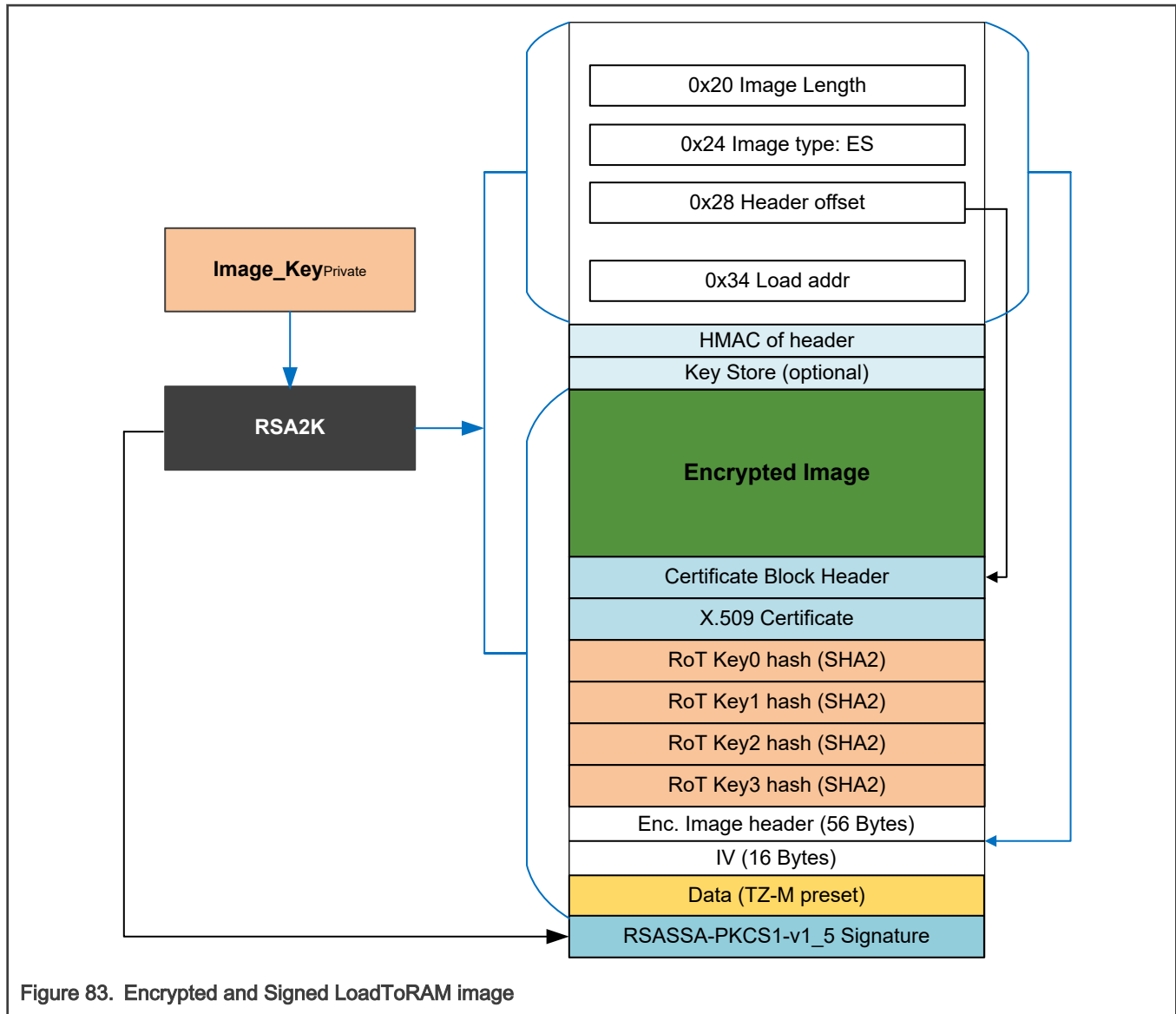
Here is a standard Cortex-M33 NVIC vector table with the Image header entries (32-bit words at offset 0x20, 0x24, 0x28 and 0x34) highlighted

Table 175. Standard Cortex_M33 NVIC vector table

Offset	Usage
0x00	Initial SP
0x04	Reset
0x08	NMI
0x0C	HardFault
0x010	MemManage
0x014	BusFault
0x018	UsageFault
0x01C	SecureFault
0x020	Image Length
0x024	Image Type
0x028	offsetToCertificateBlockInBytes
0x030	SVC
0x034	DebugMon
0x038	Load addr
0x03C	SysTick







The Load to RAM - nonXip images are the images that the ROM loader copies into the destination address in RAM, verifies digital signature, optionally decrypts in place and starts executing.

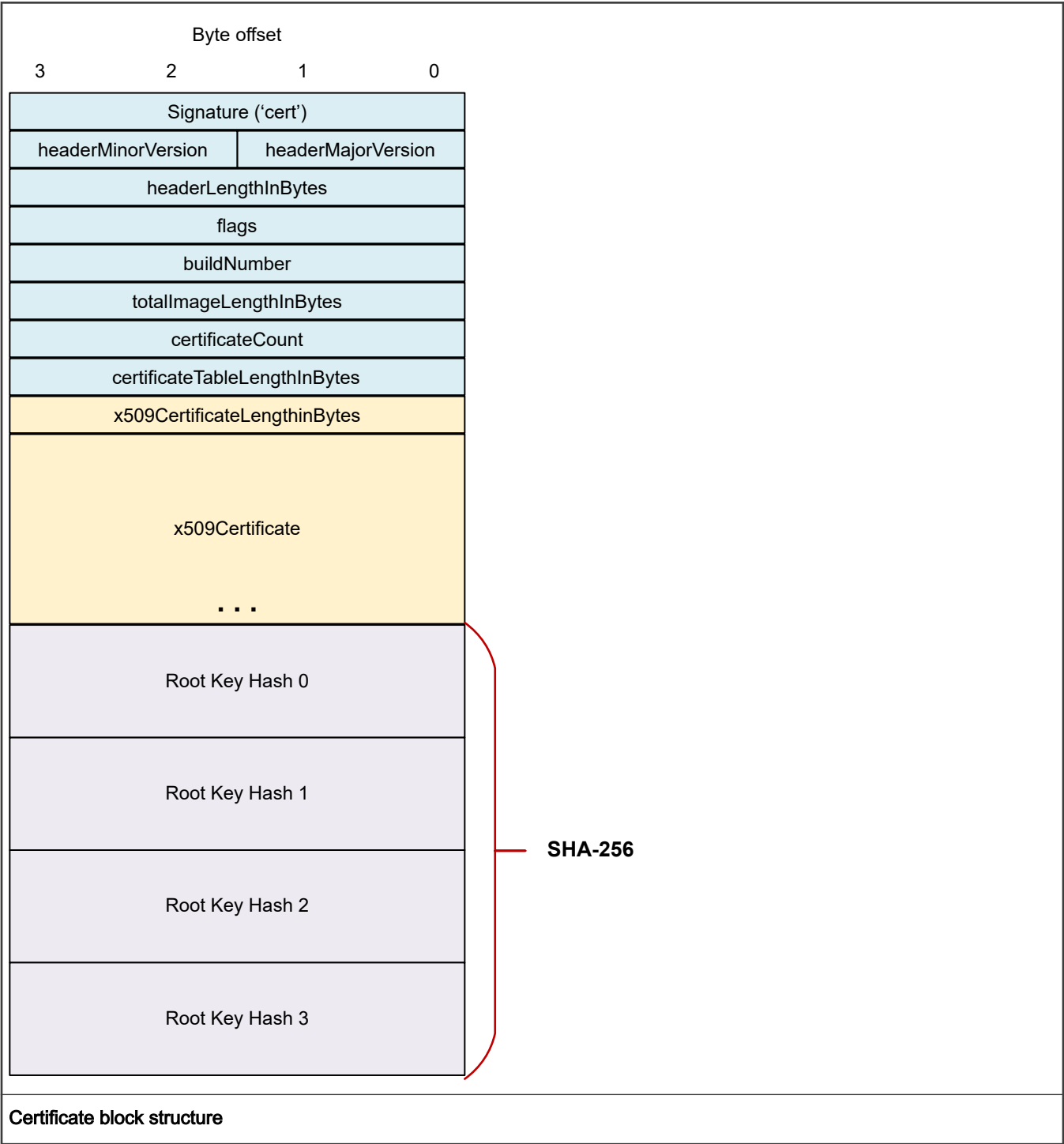
The Xip images are the images for execute-in-place, such as external SPI flash.

19.4.1 Certificate block

The certificate block consists of the certificate block header, the certificate table, and the RKH table concatenated together.

The certificate block is allowed to reside anywhere within the signed image, but must be fully contained within the signed data, such that the certificate block itself is signed. The most common constructions will have the certificate block placed at either the beginning (after the vector table) or end of the signed data.

The structure of the certificate block looks like this:



19.4.1.1 Certificate block header

The certificate block header (or just certificate header) is a structure containing information required to properly verify a signed image. As described above, it is pointed to by the offsetToCertificateBlockInBytes header offset field.

Let $O_{header} = (\text{offsetToCertificateBlockInBytes})$

The first word of the certificate block header must be 4-byte aligned.

Descriptions of the fields in the certificate block header:

Table 176. Certificate block header

Field	Description
signature	Always set to 'cert'.
headerMajorVersion	Set to 1.
headerMinorVersion	Set to 0.
headerLengthInBytes	Number of bytes long the header is, starting from the signature. Does not include the certificate table.
flags	Reserved for future use.
buildNumber	The rollback protection logic also takes into account firmware version field in OTP. The algorithm is described below.
totalImageLengthInBytes	Length in bytes of the signed data.
certificateCount	Must be greater than 0.
certificateTableLengthInBytes	Total length in bytes of the certificate table.

The key field in the certificate header is totalImageLengthInBytes. This field indicates the number of bytes of signed data, starting at offset 0 of the image. The entire certificate block **must** be contained within the signed data.

The signature field can be treated as 4-character string, without a terminating null byte, with a value of 'cert'. Represented as a 32-bit little endian constant, the value would be (('c' << 8) | ('e' << 16) | ('t' << 24)).

The buildNumber field is used for firmware version rollback protection. During boot, the ROM compares the lower 8-bits of the buildNumber field with OTP words 144 to 147 (trusted firmware version). If the value from certificate is lower than the OTP value, the image is treated as an invalid image. Similar check is done by ROM when the skboot_authenticate() ROM API function is called from applications. In addition to this check, the ROM API caller application can also set the higher 24-bits in the options parameter passed to the function call. In such case, the ROM API function also compares the upper 24-bits of the buildNumber field in certificate with the 24-bits parameter passed to ROM API call as parameter. Again, when the value from certificate is lower than the ROM API call parameter, the image is treated as an invalid image.

19.4.1.2 Certificate table

Immediately following the certificate block header is the certificate table. It consists of a complete chain of one or more X.509 certificates, each prefixed with a length word.

Let $O_{\text{cert-table}} = (\text{offsetToCertificateBlockInBytes} + \text{headerLengthInBytes})$

The x509CertificateLengthInBytes field for each certificate must be set to the length of that certificate's data in bytes, rounded up to the next word (4-byte) alignment. Thus, x509CertificateLengthInBytes must be divisible by 4. x509Certificate contains the actual certificate data and can be of variable length. There may be from 0-3 bytes of padding inserted after the certificate data. The cert_entry structure is repeated for certificateCount entries in the table. The total number of bytes occupied by the table must equal certificateTableLengthInBytes and must always be divisible by 4.

There are a number of restrictions on the certificates:

Only x509 v3 format certificates are supported. Must be DER encoded. Must use RSA-2048, RSA-3072 or RSA-4096 and SHA-256. All certificates must use RSA keys with a modulus bitlength greater than or equal the RSA bitlength specified by the security profile. This means that a 4096-bit or 3072-bit root key followed by a 2048-bit image key is allowed, if the security profile is set to 2048-bit keys. The SHA-256 hash of the public key contained in the first certificate in the table must be present in the RKH table.

The certificate table can contain one or more certificates. Certificates must be positioned in the table starting with the root certificate, followed by each subsequent certificate in the chain in order of signing. The final certificate in the table is called the image signing certificate. Using a single certificate is allowed. In this case, the sole certificate must be self-signed and must not be a CA. If multiple certificates are used, the root must be self-signed and all but the last must be CAs.

The RSA public key from the root certificate is denoted RPK, while the RSA public key from the image signing certificate is denoted IPK. The two most common configurations will be:

One self-signed certificate

Self-signed root CA certificate, followed by image signing certificate which is itself signed by the root certificate.

19.5 Firmware Update ROM support using SB2 file

The Secure Binary (SB) image format is a command-based firmware update image. It has a long history, and has been used on multiple STMP, i.MX, and Kinetis devices. One can imagine the SB2 file as a script (commands and data), for which the ROM is the interpreter. The RT500 silicon supports version 2.1 of the SB image format. Version 2.1 updates the encryption scheme to use modern algorithms, adds support for signed images and it also makes the usage of digital signature mandatory. SB 2.1 file is always digitally signed and encrypted, and the ROM verifies the digital signature before decrypting the content.

The SB 2.1 file format also uses AES encryption for confidentiality and HMAC for extending trust from the signed part of the SB file to the command and data part of the SB file. These two keys (AES decrypt key and HMAC key) are wrapped in the RFC3394 key blob, for which the key wrapping key is the SBKEK key.

The layout of an SB 2.1 file is shown in the elftosb tool User's Guide. The elftosb tool is the NXP image signing and SB file creating tool for Windows/Linux/MAC. In this chapter, we provide introductory description of the SB file format components, while for the exact description, one shall refer to the elftosb tool User's Guide.

19.5.1 Header

The header contains plaintext information about the SB file, such as version, length RSA signature of the hash computed from all the header plaintext data. The RSA Verify logic implemented in the ROM for the SB file version 2.1 is the same as the logic for the internal flash image authentication, that ROM can execute during the secure boot flow.

19.5.2 MAC of the Section MAC table

The expected MAC of the Section MAC table. During image verification, the actual MAC of the Section MAC table is computed and compared with the expected MAC. Note the expected MAC value itself is verified by the digital signature verification.

19.5.3 Key blob

The key blob wraps two 256-bit keys using the RFC3394 algorithm. It provides integrity and authenticity over the wrapped keys. The two keys in the key blob are:

1. Data Encryption Key (K DEK).
2. MAC Key (K MAC)

The K DEK is used to encrypt section data in the SB file. K MAC is used for header HMAC (if available) and section HMACs.

Both keys are uniquely generated each time an SB file is built. They are wrapped with the SBKEK that is programmed into the target device's OTP.

19.5.4 Sections

The content of SB files is divided into an arbitrary number of sections, each with a unique ID. Every section is preceded with a boot tag that acts as a header, plus an HMAC table. A section may be either a bootable section that contains boot commands, or a data section containing data not used by the loader. There must be at least 1 bootable section for an SB file to be valid. The RT5xxS ROM loader supports only a single section.

19.5.4.1 Boot tag

Boot tags prefix a section with the information about that section. They form a linked list within the SB file, allowing the loader to sequentially search for a given section. Boot tags are always encrypted using AES-CTR.

19.5.4.2 Section MAC table

Following the boot tag is a table of MACs used to verify the integrity and authenticity of section data. These MACs are computed using the HMAC-SHA256 algorithm with the K MAC key from the SB file's key blob. The number of MACs for a section is configurable by the user to trade between memory utilization and protection granularity.

19.5.4.3 Bootable section

A section that has the bootable section flag set is called a bootable section. It contains a sequence of boot commands that are processed by the loader to perform a firmware update.

The boot commands are described in the elftosb User's Guide. The RT5xx ROM loader provides the support for the following bootloader commands:

SecureFirmwareVersionCheck, NonsecureFirmwareVersionCheck, Execute

WriteMemory, FillMemory, ConfigureMemory, EfuseProgramOnce

The WriteMemory and FillMemory commands can be used to write data to RAMs. WriteMemory can be also used to program external NOR SPI flash. ConfigureMemory command can be used to configure RT5xx SPI module for the communication with an external NOR SPI flash memory. EfuseProgramOnce is used to blow OTP fuses.

The recovery boot mode using SB 2.1 file supports only four commands:

WriteMemory (RAM only) and Execute.

SecureFirmwareVersionCheck and NonsecureFirmwareVersionCheck

19.5.4.4 Data section

Any section for which the bootable section flag is cleared is a data section. The loader does not examine the contents of such sections; it simply skips over them. Data sections may optionally be unencrypted by setting the cleartext flag.

19.5.4.5 Certificate block header, certificates, RKH table

For SB 2.1, the certificate block header, certificate chain and RKH table are mandatory parts of the SB 2.1 file header.

19.5.4.6 Signature

For SB 2.1, the signature is mandatory and immediately follows the RKH table.

The SB 2.1 file header have the same structure as a signed image in the internal flash. Thus, the ROM's image authenticate function is used to verify digital signature of XiP image and SB 2.1 header. The signature in RSASSA-PKCS1-v1_5 format is appended to the tail end of the internal flash image to the tail end of the SB 2.1 header.

19.5.5 Usage of Firmware Update

SB 2.1 files are always encrypted, and the header is always signed. Users may call the loader API from their application code to authenticate an image, either signed code or an SB file.

The recommended method to perform secure firmware updates is as follows:

- User application receives an encrypted SB file containing new firmware and stores it in external SPI flash, or a similar memory.
- Use API to authenticate SB file.
- Use API to decrypt and load the SB file.
- If also using secure boot, the API can be used to authenticate the new firmware in flash before rebooting into it. If this final

authentication fails, the new firmware should be made non-executable by erasing and writing over critical regions of it such as the vector table. Even if not using secure boot, the code written to flash can still be signed to support this final authentication step.

19.5.5.1 Device setup required for SB 2.1 processing

The SB 2.1 processing by ROM depends on the presence of a valid key store with SBKEK key code and also on presence of the RKT hash (RKTH) in the OTP memory. The first choice is OTP or PUF based key store. Below is an example of a PUF based key store provisioning into the device using the ROM bootloader key provisioning commands:

PUF enroll (generate activation code into key store)

```
blhost -p comx -- key-provisioning enroll
```

install SBKEK into key store. SBKEK key type = 3.

```
blhost -p comx -- key-provisioning set_user_key 3 sbkek.bin
```

install USERKEK into key store. USERKEK key type = 11.

```
blhost -p comx -- key-provisioning set_user_key 11 userkek.bin
```

generate random UDS; UDS key type = 12.

```
blhost -p comx -- key-provisioning set_key 12 32
```

generate random OTFAD_KEK. OTFAD_KEK key type = 2

```
blhost -p comx -- key-provisioning set_key 2 16
```

save the key store to external NVM memory at offset 0x800

```
blhost -p comx -- key-provisioning write_key_nonvolatile 0
```

In case of OTP based key store, the SBKEK key is derived from the MASTERKEY256 in fuses.

19.5.5.2 The role of OTP shadow registers during device development lifecycle

One important feature that can help developers to setup device for secure boot is the OTP shadow registers. When the device is in the development lifecycle, the shadow registers are writable (through debug port or by an application) and they are POR clear only and they are re-loaded from fuses only by POR reset. Thus, it is possible to configure secure boot by writing to shadow registers, debug the secure boot flow, and only blow fuses (change device lifecycle) once the device setup is stable. However, not all fuses have shadow registers. Note that shadow registers are added to RKTH OTP words (120 to 127). Users in development stage could use these shadow registers to write the RKTH hash without programming the fuse.

19.6 OTFAD

External SPI memory can be encrypted, and the content being decrypted on-the-fly while the controller is fetching data from the external SPI memory. From the CPU perspective, the images supported are the plaintext images (plain, plain with CRC, plain with RSA signature) both Xip and Load to RAM. In the physical memory, the data is encrypted, and the controller decrypts the data on the fly.

The OTFAD image for the external flash can be prepared by the elftosb command line tool, provided by NXP.

19.6.1 OTFAD image

The RT5xxS ROM supports booting from an OTFAD enabled external NOR SPI flash. The external flash layout is shown below:

Table 177. OTFAD Key Blob layout in the external NOR flash

Offset	Description
0x00	OTFAD region 0 key blob
0x40	OTFAD region 1 key blob

Table continues on the next page...

Table 177. OTFAD Key Blob layout in the external NOR flash (continued)

Offset	Description
0x80	OTFAD region 2 key blob
0xC0	OTFAD region 3 key blob

NOTE

All blobs are present.

Table 178. Key Blob “n” content after ROM key blob unwrap

CTXn_KEY0 – CTXn_KEY3	AES Key Words. Used by ROM to configure OTFAD CTXn_KEYm registers.
CTXn_CTR0 – CTXn_CTR1	AES Counter Words. Used by ROM to configure OTFAD CTXn_CTRm registers.
STARTADDR for region n	Start address. Used by ROM to configure OTFAD CTXn_RGD_W0 register.
ENDADDR + RO + ADE + VLD for region n	End address and region configuration. Used by ROM to configure OTFAD CTXn_RGD_W1 register. ENDADDR – end address of region n RO – read only ADE – AES decryption enable VLD – Context is valid

The elftosb command line tool provides support for the OTFAD image creation. The tools encapsulate the OTFAD image into the SB2 firmware update container, which can be loaded by the ReceiveSBFile ROM bootloader command. A user provides a plain/plain CRC/plain signed image to the tool, along with the specification of the RFC3394 blob content, SBKEK key, OTFAD_KEK key and firmware updated commands (such as erase flash, configure flash, then the tool outputs SB2 file with the OTFAD image.

19.6.2 OTFAD boot

The OTFAD boot path during master boot from external SPI flash is enabled by the fuse word at offset 0x1A8, bits[13:12]. During development, it is advised to use shadow register to test the OTFAD boot path, and only blow the OTFAD boot enable fuse when the OTFAD image is verified to be functional.

19.7 ROM TrustZone support

19.7.1 TrustZone image type

From TrustZone perspective the ROM distinguishes between two image types:

- TrustZone disabled image
- TrustZone enabled image

The TrustZone Image type is defined in the vector section of image header at offset 0x24, bit 14 (TZM_IMAGE_TYPE)

Table 179. TrustZone image type

TZM_IMAGE_TYPE value (offset 0x24, bit 14)	
0	TrustZone enabled image
1	TrustZone disabled image

19.7.1.1 TrustZone disabled image

TrustZone disabled image is an image, which is supposed to be executed on devices without TrustZone (CM33 without security extension). To keep full software compatibility between CM33 with and without security extension, this software/image must be executed in normal mode. To allow easy transition between devices with and without security extension, the RT500 ROM supports direct execution of software developed for CM33 devices without security extension. If the TrustZone disabled image is executed, the ROM, before it jumps to user application, configures all device resources into normal world, lock access to all TrustZone related configuration registers, switches from secure to normal world and finally jumps to user application. This mode allows easy reuse of the software developed for Cortex-M33 without security extension. The user doesn't need to do any software modification.

NOTE

After jump into user application, the security extension (TrustZone) is still enabled. The MCU is running in normal mode, all TrustZone related configuration registers are locked, memory region 0x13000000-0x13001000 (first 4kB of ROM) is configured as secure memory. The user should avoid code execution or data read from this memory otherwise HardFault will be generated.

19.7.1.2 TrustZone enabled image

TrustZone enabled image is an image, which is supposed to be executed on devices with TrustZone (M33 with security extension) and it utilizes TrustZone. In this case the user application is split into secure and non-secure part, and after device reset, the software execution starts in secure mode. If TrustZone enabled image is executed, the ROM doesn't provide any TrustZone settings (except optional TrustZone preset data configuration) and jumps into user application in secure mode. The executed software/image is responsible for TrustZone settings and jump from secure to normal world.

19.7.2 TrustZone preset data

RT500 ROM provides support for TrustZone data configuration during boot process. The TrustZone preset data includes:

- VTOR, VTOR_NS, NVIC_ITNS0, NVIC_ITNS1 (CPU0) registers
- Secure MPU
- Non-secure MPU
- SAU
- AHB Secure Controller

If the TrustZone preset is enabled, the ROM, after image validation, configures all TrustZone related registers by data, provided at the end of the image. If any register or whole peripheral has lock feature and corresponding lock bit is set, the register is also locked, so any further register modification is not possible until next reset.

This feature increases robustness of the user application since the user application jumps into pre-configured TrustZone environment and it doesn't need to contain any TrustZone configuration code.

19.7.2.1 TrustZone preset data structure

The TrustZone preset data structure is defined by following C structure:

```
typedef struct _tzm_secure_config
{
```

```

uint32_t vtor_addr;      /*!< Secure vector table address */
uint32_t vtor_ns_addr; /*!< Non-secure vector table address */
uint32_t nvic_itns0;     /*!< Interrupt target non-secure register 0 */
uint32_t nvic_itns1;     /*!< Interrupt target non-secure register 1 */
uint32_t mpu_ctrl;       /*!< MPU Control Register.*/
uint32_t mpu_mair0;      /*!< MPU Memory Attribute Indirection Register 0 */
uint32_t mpu_mair1;      /*!< MPU Memory Attribute Indirection Register 1 */
uint32_t mpu_rbar0;      /*!< MPU Region 0 Base Address Register */
uint32_t mpu_rlar0;      /*!< MPU Region 0 Limit Address Register */
uint32_t mpu_rbar1;      /*!< MPU Region 1 Base Address Register */
uint32_t mpu_rlar1;      /*!< MPU Region 1 Limit Address Register */
uint32_t mpu_rbar2;      /*!< MPU Region 2 Base Address Register */
uint32_t mpu_rlar2;      /*!< MPU Region 2 Limit Address Register */
uint32_t mpu_rbar3;      /*!< MPU Region 3 Base Address Register */
uint32_t mpu_rlar3;      /*!< MPU Region 3 Limit Address Register */
uint32_t mpu_rbar4;      /*!< MPU Region 4 Base Address Register */
uint32_t mpu_rlar4;      /*!< MPU Region 4 Limit Address Register */
uint32_t mpu_rbar5;      /*!< MPU Region 5 Base Address Register */
uint32_t mpu_rlar5;      /*!< MPU Region 5 Limit Address Register */
uint32_t mpu_rbar6;      /*!< MPU Region 6 Base Address Register */
uint32_t mpu_rlar6;      /*!< MPU Region 6 Limit Address Register */
uint32_t mpu_rbar7;      /*!< MPU Region 7 Base Address Register */
uint32_t mpu_rlar7;      /*!< MPU Region 7 Limit Address Register */
uint32_t mpu_ctrl_ns;    /*!< Non-secure MPU Control Register.*/
uint32_t mpu_mair0_ns;   /*!< Non-secure MPU Memory Attribute Indirection Register 0 */
uint32_t mpu_mair1_ns;   /*!< Non-secure MPU Memory Attribute Indirection Register 1 */
uint32_t mpu_rbar0_ns;   /*!< Non-secure MPU Region 0 Base Address Register */
uint32_t mpu_rlar0_ns;   /*!< Non-secure MPU Region 0 Limit Address Register */
uint32_t mpu_rbar1_ns;   /*!< Non-secure MPU Region 1 Base Address Register */
uint32_t mpu_rlar1_ns;   /*!< Non-secure MPU Region 1 Limit Address Register */
uint32_t mpu_rbar2_ns;   /*!< Non-secure MPU Region 2 Base Address Register */
uint32_t mpu_rlar2_ns;   /*!< Non-secure MPU Region 2 Limit Address Register */
uint32_t mpu_rbar3_ns;   /*!< Non-secure MPU Region 3 Base Address Register */
uint32_t mpu_rlar3_ns;   /*!< Non-secure MPU Region 3 Limit Address Register */
uint32_t mpu_rbar4_ns;   /*!< Non-secure MPU Region 4 Base Address Register */
uint32_t mpu_rlar4_ns;   /*!< Non-secure MPU Region 4 Limit Address Register */
uint32_t mpu_rbar5_ns;   /*!< Non-secure MPU Region 5 Base Address Register */
uint32_t mpu_rlar5_ns;   /*!< Non-secure MPU Region 5 Limit Address Register */
uint32_t mpu_rbar6_ns;   /*!< Non-secure MPU Region 6 Base Address Register */
uint32_t mpu_rlar6_ns;   /*!< Non-secure MPU Region 6 Limit Address Register */
uint32_t mpu_rbar7_ns;   /*!< Non-secure MPU Region 7 Base Address Register */
uint32_t mpu_rlar7_ns;   /*!< Non-secure MPU Region 7 Limit Address Register */
uint32_t sau_ctrl;       /*!< SAU Control Register.*/
uint32_t sau_rbar0;      /*!< SAU Region 0 Base Address Register */

```



```

uint32_t sau_rlar0;      /*!< SAU Region 0 Limit Address Register */
uint32_t sau_rbar1;      /*!< SAU Region 1 Base Address Register */
/* uint32_t sau_rlar1;    /*!< SAU Region 1 Limit Address Register */
/* uint32_t sau_rbar2;    /*!< SAU Region 2 Base Address Register */
/* uint32_t sau_rlar2;    /*!< SAU Region 2 Limit Address Register */
/* uint32_t sau_rbar3;    /*!< SAU Region 3 Base Address Register */
/* uint32_t sau_rlar3;    /*!< SAU Region 3 Limit Address Register */
/* uint32_t sau_rbar4;    /*!< SAU Region 4 Base Address Register */
/* uint32_t sau_rlar4;    /*!< SAU Region 4 Limit Address Register */
/* uint32_t sau_rbar5;    /*!< SAU Region 5 Base Address Register */
/* uint32_t sau_rlar5;    /*!< SAU Region 5 Limit Address Register */
/* uint32_t sau_rbar6;    /*!< SAU Region 6 Base Address Register */
/* uint32_t sau_rlar6;    /*!< SAU Region 6 Limit Address Register */
/* uint32_t sau_rbar7;    /*!< SAU Region 7 Base Address Register */
/* uint32_t sau_rlar7;    /*!< SAU Region 7 Limit Address Register */
/* uint32_t bootrom0_slave_rule0; /*!< ROM Slave Rule Register 0 */
/* uint32_t bootrom0_mem_rule0; /*!< ROM Memory Rule Register 0 */
/* uint32_t bootrom0_mem_rule1; /*!< ROM Memory Rule Register 1 */
/* uint32_t bootrom0_mem_rule2; /*!< ROM Memory Rule Register 2 */
/* uint32_t bootrom0_mem_rule3; /*!< ROM Memory Rule Register 3 */
/* uint32_t qospi_slave_rule0; /*!< Quad/Octal SPI Slave Rule Register 0 */
/* uint32_t qospi0_mem_rule0; /*!< Quad/Octal SPI 0 Memory Rule Register 0 */
/* uint32_t qospi0_mem_rule1; /*!< Quad/Octal SPI 0 Memory Rule Register 1 */
/* uint32_t qospi0_mem_rule2; /*!< Quad/Octal SPI 0 Memory Rule Register 2 */
/* uint32_t qospi0_mem_rule3; /*!< Quad/Octal SPI 0 Memory Rule Register 3 */
/* uint32_t qospi1_mem_rule0; /*!< Quad/Octal SPI 1 Memory Rule Register 0 */
/* uint32_t qospi1_mem_rule1; /*!< Quad/Octal SPI 1 Memory Rule Register 1 */
/* uint32_t qospi1_mem_rule2; /*!< Quad/Octal SPI 1 Memory Rule Register 2 */
/* uint32_t qospi1_mem_rule3; /*!< Quad/Octal SPI 1 Memory Rule Register 3 */
/* uint32_t qospi2_mem_rule0; /*!< Quad/Octal SPI 2 Memory Rule Register 0 */
uint32_t qospi2_mem_rule1; /*!< Quad/Octal SPI 2 Memory Rule Register 1 */
uint32_t qospi2_mem_rule2; /*!< Quad/Octal SPI 2 Memory Rule Register 2 */
uint32_t qospi2_mem_rule3; /*!< Quad/Octal SPI 2 Memory Rule Register 3 */
uint32_t qospi3_mem_rule0; /*!< Quad/Octal SPI 3 Memory Rule Register 0 */
uint32_t qospi3_mem_rule1; /*!< Quad/Octal SPI 3 Memory Rule Register 1 */
uint32_t qospi3_mem_rule2; /*!< Quad/Octal SPI 3 Memory Rule Register 2 */
uint32_t qospi3_mem_rule3; /*!< Quad/Octal SPI 3 Memory Rule Register 3 */
uint32_t qospi4_mem_rule0; /*!< Quad/Octal SPI 4 Memory Rule Register 0 */
uint32_t qospi4_mem_rule1; /*!< Quad/Octal SPI 4 Memory Rule Register 1 */
uint32_t qospi4_mem_rule2; /*!< Quad/Octal SPI 4 Memory Rule Register 2 */
uint32_t qospi4_mem_rule3; /*!< Quad/Octal SPI 4 Memory Rule Register 3 */
uint32_t ram0_slave_rule; /*!< RAM0 Slave Rule Register */
uint32_t ram00_mem_rule0; /*!< RAM00 Memory Rule Register 0 */
uint32_t ram00_mem_rule1; /*!< RAM00 Memory Rule Register 1 */

```



```

uint32_t ram00_mem_rule2;    /*!< RAM00 Memory Rule Register 2 */
uint32_t ram00_mem_rule3;    /*!< RAM00 Memory Rule Register 3 */
uint32_t ram01_mem_rule0;    /*!< RAM01 Memory Rule Register 0 */
uint32_t ram01_mem_rule1;    /*!< RAM01 Memory Rule Register 1 */
uint32_t ram01_mem_rule2;    /*!< RAM01 Memory Rule Register 2 */
uint32_t ram01_mem_rule3;    /*!< RAM01 Memory Rule Register 3 */
uint32_t ram1_slave_rule;    /*!< RAM1 Slave Rule Register */
uint32_t ram10_mem_rule0;    /*!< RAM10 Memory Rule Register 0 */
uint32_t ram10_mem_rule1;    /*!< RAM10 Memory Rule Register 1 */
uint32_t ram10_mem_rule2;    /*!< RAM10 Memory Rule Register 2 */
uint32_t ram10_mem_rule3;    /*!< RAM10 Memory Rule Register 3 */
uint32_t ram11_mem_rule0;    /*!< RAM11 Memory Rule Register 0 */
uint32_t ram11_mem_rule1;    /*!< RAM11 Memory Rule Register 1 */
uint32_t ram11_mem_rule2;    /*!< RAM11 Memory Rule Register 2 */
uint32_t ram11_mem_rule3;    /*!< RAM11 Memory Rule Register 3 */
uint32_t ram2_slave_rule;    /*!< RAM2 Slave Rule Register */
uint32_t ram20_mem_rule0;    /*!< RAM20 Memory Rule Register 0 */
uint32_t ram20_mem_rule1;    /*!< RAM20 Memory Rule Register 1 */
uint32_t ram20_mem_rule2;    /*!< RAM20 Memory Rule Register 2 */
uint32_t ram20_mem_rule3;    /*!< RAM20 Memory Rule Register 3 */
uint32_t ram21_mem_rule0;    /*!< RAM21 Memory Rule Register 0 */
uint32_t ram21_mem_rule1;    /*!< RAM21 Memory Rule Register 1 */
uint32_t ram21_mem_rule2;    /*!< RAM21 Memory Rule Register 2 */
uint32_t ram21_mem_rule3;    /*!< RAM21 Memory Rule Register 3 */
uint32_t ram22_mem_rule0;    /*!< RAM22 Memory Rule Register 0 */
uint32_t ram22_mem_rule1;    /*!< RAM22 Memory Rule Register 1 */
uint32_t ram22_mem_rule2;    /*!< RAM22 Memory Rule Register 2 */
uint32_t ram22_mem_rule3;    /*!< RAM22 Memory Rule Register 3 */
uint32_t ram23_mem_rule0;    /*!< RAM23 Memory Rule Register 0 */
uint32_t ram23_mem_rule1;    /*!< RAM23 Memory Rule Register 1 */
uint32_t ram23_mem_rule2;    /*!< RAM23 Memory Rule Register 2 */
uint32_t ram23_mem_rule3;    /*!< RAM23 Memory Rule Register 3 */
uint32_t ram3_slave_rule;    /*!< RAM3 Slave Rule Register */
uint32_t ram30_mem_rule0;    /*!< RAM30 Memory Rule Register 0 */
uint32_t ram30_mem_rule1;    /*!< RAM30 Memory Rule Register 1 */
uint32_t ram30_mem_rule2;    /*!< RAM30 Memory Rule Register 2 */
uint32_t ram30_mem_rule3;    /*!< RAM30 Memory Rule Register 3 */
uint32_t ram31_mem_rule0;    /*!< RAM31 Memory Rule Register 0 */
uint32_t ram31_mem_rule1;    /*!< RAM31 Memory Rule Register 1 */
uint32_t ram31_mem_rule2;    /*!< RAM31 Memory Rule Register 2 */
uint32_t ram31_mem_rule3;    /*!< RAM31 Memory Rule Register 3 */
uint32_t ram32_mem_rule0;    /*!< RAM32 Memory Rule Register 0 */
uint32_t ram32_mem_rule1;    /*!< RAM32 Memory Rule Register 1 */
uint32_t ram32_mem_rule2;    /*!< RAM32 Memory Rule Register 2 */

```

```

uint32_t ram32_mem_rule3;    /*!< RAM32 Memory Rule Register 3 */
uint32_t ram33_mem_rule0;    /*!< RAM33 Memory Rule Register 0 */
uint32_t ram33_mem_rule1;    /*!< RAM33 Memory Rule Register 1 */
uint32_t ram33_mem_rule2;    /*!< RAM33 Memory Rule Register 2 */
uint32_t ram33_mem_rule3;    /*!< RAM33 Memory Rule Register 3 */
uint32_t ram4_slave_rule;    /*!< RAM4 Slave Rule Register */
uint32_t ram40_mem_rule0;    /*!< RAM40 Memory Rule Register 0 */
uint32_t ram40_mem_rule1;    /*!< RAM40 Memory Rule Register 1 */
uint32_t ram40_mem_rule2;    /*!< RAM40 Memory Rule Register 2 */
uint32_t ram40_mem_rule3;    /*!< RAM40 Memory Rule Register 3 */
uint32_t ram41_mem_rule0;    /*!< RAM41 Memory Rule Register 0 */
uint32_t ram41_mem_rule1;    /*!< RAM41 Memory Rule Register 1 */
uint32_t ram41_mem_rule2;    /*!< RAM41 Memory Rule Register 2 */
uint32_t ram41_mem_rule3;    /*!< RAM41 Memory Rule Register 3 */
uint32_t ram42_mem_rule0;    /*!< RAM42 Memory Rule Register 0 */
uint32_t ram42_mem_rule1;    /*!< RAM42 Memory Rule Register 1 */
uint32_t ram42_mem_rule2;    /*!< RAM42 Memory Rule Register 2 */
uint32_t ram42_mem_rule3;    /*!< RAM42 Memory Rule Register 3 */
uint32_t ram43_mem_rule0;    /*!< RAM43 Memory Rule Register 0 */
uint32_t ram43_mem_rule1;    /*!< RAM43 Memory Rule Register 1 */
uint32_t ram43_mem_rule2;    /*!< RAM43 Memory Rule Register 2 */
uint32_t ram43_mem_rule3;    /*!< RAM43 Memory Rule Register 3 */
uint32_t ram5_slave_rule;    /*!< RAM5 Slave Rule Register */
uint32_t ram50_mem_rule0;    /*!< RAM50 Memory Rule Register 0 */
uint32_t ram50_mem_rule1;    /*!< RAM50 Memory Rule Register 1 */
uint32_t ram50_mem_rule2;    /*!< RAM50 Memory Rule Register 2 */
uint32_t ram50_mem_rule3;    /*!< RAM50 Memory Rule Register 3 */
uint32_t ram51_mem_rule0;    /*!< RAM51 Memory Rule Register 0 */
uint32_t ram51_mem_rule1;    /*!< RAM51 Memory Rule Register 1 */
uint32_t ram51_mem_rule2;    /*!< RAM51 Memory Rule Register 2 */
uint32_t ram51_mem_rule3;    /*!< RAM51 Memory Rule Register 3 */
uint32_t ram52_mem_rule0;    /*!< RAM52 Memory Rule Register 0 */
uint32_t ram52_mem_rule1;    /*!< RAM52 Memory Rule Register 1 */
uint32_t ram52_mem_rule2;    /*!< RAM52 Memory Rule Register 2 */
uint32_t ram52_mem_rule3;    /*!< RAM52 Memory Rule Register 3 */
uint32_t ram53_mem_rule0;    /*!< RAM53 Memory Rule Register 0 */
uint32_t ram53_mem_rule1;    /*!< RAM53 Memory Rule Register 1 */
uint32_t ram53_mem_rule2;    /*!< RAM53 Memory Rule Register 2 */
uint32_t ram53_mem_rule3;    /*!< RAM53 Memory Rule Register 3 */
uint32_t ram6_slave_rule;    /*!< RAM6 Slave Rule Register */
uint32_t ram60_mem_rule0;    /*!< RAM60 Memory Rule Register 0 */
uint32_t ram60_mem_rule1;    /*!< RAM60 Memory Rule Register 1 */
uint32_t ram60_mem_rule2;    /*!< RAM60 Memory Rule Register 2 */
uint32_t ram60_mem_rule3;    /*!< RAM60 Memory Rule Register 3 */

```

```
uint32_t ram61_mem_rule0; /*!< RAM61 Memory Rule Register 0 */
uint32_t ram61_mem_rule1; /*!< RAM61 Memory Rule Register 1 */
uint32_t ram61_mem_rule2; /*!< RAM61 Memory Rule Register 2 */
uint32_t ram61_mem_rule3; /*!< RAM61 Memory Rule Register 3 */
uint32_t ram62_mem_rule0; /*!< RAM62 Memory Rule Register 0 */
uint32_t ram62_mem_rule1; /*!< RAM62 Memory Rule Register 1 */
uint32_t ram62_mem_rule2; /*!< RAM62 Memory Rule Register 2 */
uint32_t ram62_mem_rule3; /*!< RAM62 Memory Rule Register 3 */
uint32_t ram63_mem_rule0; /*!< RAM63 Memory Rule Register 0 */
uint32_t ram63_mem_rule1; /*!< RAM63 Memory Rule Register 1 */
uint32_t ram63_mem_rule2; /*!< RAM63 Memory Rule Register 2 */
uint32_t ram63_mem_rule3; /*!< RAM63 Memory Rule Register 3 */
uint32_t ram7_slave_rule; /*!< RAM7 Slave Rule Register */
uint32_t ram70_mem_rule0; /*!< RAM70 Memory Rule Register 0 */
uint32_t ram70_mem_rule1; /*!< RAM70 Memory Rule Register 1 */
uint32_t ram70_mem_rule2; /*!< RAM70 Memory Rule Register 2 */
uint32_t ram70_mem_rule3; /*!< RAM70 Memory Rule Register 3 */
uint32_t ram71_mem_rule0; /*!< RAM71 Memory Rule Register 0 */
uint32_t ram71_mem_rule1; /*!< RAM71 Memory Rule Register 1 */
uint32_t ram71_mem_rule2; /*!< RAM71 Memory Rule Register 2 */
uint32_t ram71_mem_rule3; /*!< RAM71 Memory Rule Register 3 */
uint32_t ram72_mem_rule0; /*!< RAM72 Memory Rule Register 0 */
uint32_t ram72_mem_rule1; /*!< RAM72 Memory Rule Register 1 */
uint32_t ram72_mem_rule2; /*!< RAM72 Memory Rule Register 2 */
uint32_t ram72_mem_rule3; /*!< RAM72 Memory Rule Register 3 */
uint32_t ram73_mem_rule0; /*!< RAM73 Memory Rule Register 0 */
uint32_t ram73_mem_rule1; /*!< RAM73 Memory Rule Register 1 */
uint32_t ram73_mem_rule2; /*!< RAM73 Memory Rule Register 2 */
uint32_t ram73_mem_rule3; /*!< RAM73 Memory Rule Register 3 */
uint32_t ram8_slave_rule; /*!< RAM8 Slave Rule Register */
uint32_t ram80_mem_rule0; /*!< RAM80 Memory Rule Register 0 */
uint32_t ram80_mem_rule1; /*!< RAM80 Memory Rule Register 1 */
uint32_t ram80_mem_rule2; /*!< RAM80 Memory Rule Register 2 */
uint32_t ram80_mem_rule3; /*!< RAM80 Memory Rule Register 3 */
uint32_t ram81_mem_rule0; /*!< RAM81 Memory Rule Register 0 */
uint32_t ram81_mem_rule1; /*!< RAM81 Memory Rule Register 1 */
uint32_t ram81_mem_rule2; /*!< RAM81 Memory Rule Register 2 */
uint32_t ram81_mem_rule3; /*!< RAM81 Memory Rule Register 3 */
uint32_t ram82_mem_rule0; /*!< RAM81 Memory Rule Register 0 */
uint32_t ram82_mem_rule1; /*!< RAM81 Memory Rule Register 1 */
uint32_t ram82_mem_rule2; /*!< RAM81 Memory Rule Register 2 */
uint32_t ram82_mem_rule3; /*!< RAM81 Memory Rule Register 3 */
uint32_t ram83_mem_rule0; /*!< RAM81 Memory Rule Register 0 */
uint32_t ram83_mem_rule1; /*!< RAM81 Memory Rule Register 1 */
```

```

uint32_t ram83_mem_rule2;    /*!< RAM81 Memory Rule Register 2 */
uint32_t ram83_mem_rule3;    /*!< RAM81 Memory Rule Register 3 */
uint32_t SmartDMA_ram_slave_rule;
uint32_t SmartDMA_ram0_slave_rule0;
uint32_t SmartDMA_ram0_slave_rule1;
uint32_t SmartDMA_ram0_slave_rule2;
uint32_t SmartDMA_ram0_slave_rule3;
uint32_t qospil_slave_rule0; /*!< Quad/Octal SPI Slave Rule Register 0 */
uint32_t qospil0_mem_rule0;  /*!< Quad/Octal SPI 0 Memory Rule Register 0 */
uint32_t qospil0_mem_rule1;  /*!< Quad/Octal SPI 0 Memory Rule Register 1 */
uint32_t qospil0_mem_rule2;  /*!< Quad/Octal SPI 0 Memory Rule Register 2 */
uint32_t qospil0_mem_rule3;  /*!< Quad/Octal SPI 0 Memory Rule Register 3 */
uint32_t qospil1_mem_rule0;  /*!< Quad/Octal SPI 1 Memory Rule Register 0 */
uint32_t qospil1_mem_rule1;  /*!< Quad/Octal SPI 1 Memory Rule Register 1 */
uint32_t qospil1_mem_rule2;  /*!< Quad/Octal SPI 1 Memory Rule Register 2 */
uint32_t qospil1_mem_rule3;  /*!< Quad/Octal SPI 1 Memory Rule Register 3 */
uint32_t qospil2_mem_rule0;  /*!< Quad/Octal SPI 2 Memory Rule Register 0 */
uint32_t qospil2_mem_rule1;  /*!< Quad/Octal SPI 2 Memory Rule Register 1 */
uint32_t qospil2_mem_rule2;  /*!< Quad/Octal SPI 2 Memory Rule Register 2 */
uint32_t qospil2_mem_rule3;  /*!< Quad/Octal SPI 2 Memory Rule Register 3 */
uint32_t qospil3_mem_rule0;  /*!< Quad/Octal SPI 3 Memory Rule Register 0 */
uint32_t qospil3_mem_rule1;  /*!< Quad/Octal SPI 3 Memory Rule Register 1 */
uint32_t qospil3_mem_rule2;  /*!< Quad/Octal SPI 3 Memory Rule Register 2 */
uint32_t qospil3_mem_rule3;  /*!< Quad/Octal SPI 3 Memory Rule Register 3 */
uint32_t qospil4_mem_rule0;  /*!< Quad/Octal SPI 4 Memory Rule Register 0 */
uint32_t qospil4_mem_rule1;  /*!< Quad/Octal SPI 4 Memory Rule Register 1 */
uint32_t qospil4_mem_rule2;  /*!< Quad/Octal SPI 4 Memory Rule Register 2 */
uint32_t qospil4_mem_rule3;  /*!< Quad/Octal SPI 4 Memory Rule Register 3 */
uint32_t apb_bridge_slave_rule0; /*!< APB Bridge Slave Rule Register */
uint32_t apb_grp0_mem_rule0;  /*!< APB Bridge Group 0 Memory Rule Register 0 */
uint32_t apb_grp0_mem_rule1;  /*!< APB Bridge Group 0 Memory Rule Register 1 */
uint32_t apb_grp0_mem_rule2;  /*!< APB Bridge Group 0 Memory Rule Register 2 */
uint32_t apb_grp0_mem_rule3;  /*!< APB Bridge Group 0 Memory Rule Register 3 */
uint32_t apb_grp1_mem_rule0;  /*!< APB Bridge Group 1 Memory Rule Register 0 */
uint32_t apb_grp1_mem_rule1;  /*!< APB Bridge Group 1 Memory Rule Register 1 */
uint32_t apb_grp1_mem_rule2;  /*!< APB Bridge Group 1 Memory Rule Register 2 */
uint32_t apb_grp1_mem_rule3;  /*!< APB Bridge Group 1 Memory Rule Register 3 */
uint32_t ahb_periph0_slave_rule0; /*!< AHB Peripherals 0 Slave Rule Register 0 */
uint32_t aips_bridge0_mem_rule0; /*!< AIPS bridge 0 Memory Rule Register 0 */
uint32_t aips_bridge0_mem_rule1; /*!< AIPS bridge 0 Memory Rule Register 1 */
uint32_t ahb_periph1_slave_rule0; /*!< AHB Peripherals 1 Slave Rule Register */
uint32_t ahb_periph1_slave_rule1; /*!< AHB Peripherals 1 Slave Rule Register */
uint32_t aips_bridge_slave_rule0; /*!< AIPS Bridge Slave Rule Register */
uint32_t aips_bridge1_mem_rule0; /*!< AIPS bridge 1 Memory Rule Register 0 */

```

```

uint32_t aips_bridge1_mem_rule1;    /*!< AIPS bridge 1 Memory Rule Register 1 */
uint32_t ahb_periph2_slave_rule0;   /*!< AHB Peripherals 2 Slave Rule Register 0 */
uint32_t security_ctrl_mem_rule0;   /*!< AHB Peripherals 2 Slave Rule Register 0 */
uint32_t ahb_periph3_slave_rule0;   /*!< AHB Peripherals 3 Slave Rule Register 0 */
uint32_t ahb_periph3_slave_rule1;   /*!< AHB Peripherals 3 Slave Rule Register 0 */
uint32_t sec_gp_reg0;               /*!< Secure GPIO Register 0 */
uint32_t sec_gp_reg1;               /*!< Secure GPIO Register 1 */
uint32_t sec_gp_reg2;               /*!< Secure GPIO Register 2 */
uint32_t sec_gp_reg3;               /*!< Secure GPIO Register 3 */
uint32_t sec_gp_reg4;               /*!< Secure GPIO Register 4 */
uint32_t sec_gp_reg5;               /*!< Secure GPIO Register 5 */
uint32_t sec_gp_reg6;               /*!< Secure GPIO Register 6 */
uint32_t sec_gp_reg7;               /*!< Secure GPIO Register 7 */
uint32_t sec_gp_reg8;               /*!< Secure GPIO Register 8 for DSP*/
uint32_t sec_gp_reg_lock;           /*!< Secure GPIO Lock Register */
uint32_t master_sec_reg;            /*!< Master Secure Level Register */
uint32_t master_sec_anti_pol_reg;   /*!< Master Secure Level Anti-pole Register */
uint32_t m33_lock_reg;              /*!< M33 Lock Control Register */
uint32_t misc_ctrl_dp_reg;          /*!< Secure Control Duplicate Register */
    uint32_t misc_ctrl_reg;          /*!< Secure Control Register */
    uint32_t misc_tzm_settings;      /*!< Miscellaneous TZM settings */
} tzm_secure_config_t;

```

The configuration data are copied one to one into appropriate registers except misc_tzm_settings. The configuration word misc_tzm_settings is defined in following table

Table 180. TrustZone data structure

Field	Function
31-1	Reserved
0 SECUREFAULTENA	SHCSR.SECUREFAULTENA control 0b - SECUREFAULTENA is set to 0 1b - SECUREFAULTENA is set to 1

The configuration data are attached in binary format at the end of the image, or in front of signature in the case of signed image.

The user can also use elftosb.exe tool to create TrustZone configuration data. For more information please see elftosb manual.

The information whether image file contains TrustZone configuration data or not is defined in the vector section of the image header at offset 0x24, bit 13 (TZM_PRESET)

Table 181. TrustZone configuration data

TZM_PRESET value (offset 24, bit 13)	
0	TrustZone data not present
1	TrustZone data present

Note: Since the TrustZone configuration is enabled just before jump to user application, the user's TrustZone configuration data must allow ROM code execution for successful transition from secure to normal mode and jump to user application. This means that user's TrustZone settings must include:

1. Whole ROM space (0x13000000-0x1301FFFF) must be configured as secure privilege.
2. In the case, that secure MPU is used, whole ROM space (0x13000000-0x1301FFFF) must be configured for code execution.

If these two conditions are not met, boot process will fail.

19.7.2.2 TrustZone image type restriction control during boot process

The user can restrict, which TrustZone image type is allowed for execution. For this purpose, there are two bits BOOT_CFG[0].TSM_IMAGE_TYPE[1:0] in OTP memory. This field restricts execution of the TrustZone image type as described in following table:

Table 182. TrustZone image type restriction control

Allowed TrustZone Image Type	BOOT_CFG[0].TSM_IMAGE_TYPE[1:0] Value
Any TrustZone image type is allowed. The image type is set in the Vectors section of the image (offset 0x24)	00b
TrustZone disabled image type is allowed only	01b
TrustZone enabled or TrustZone enabled with TrustZone Preset Data image type are allowed	10b
TrustZone enabled with Preset Data image type is allowed	11b

19.7.3 Boot ROM API and TrustZone

19.7.3.1 TrustZone disabled images

TrustZone disabled image is executed in normal mode and whole memory space is configured as non-secure (except first 4kB of ROM). Thus, the ROM API can be used without any limitation as on any **RT500** device without security extension.

19.7.3.2 TrustZone enabled images

The whole boot ROM is executed in secure mode. Thus, the user can fully control which ROM API will be available also in normal world. The user can expose full ROM API, limited ROM API functions set or modify/limit ROM API functionality. To enable the ROM API into normal world the user must create entry function for every ROM API function exposed to normal world. Example of entry function for kboot init function can be seen below:

```
#define BOOTLOADER_TREE_LOCATION (0x130010f0u)
#define BOOTLOADER_API_TREE_POINTER (*(bootloader_tree_t **)BOOTLOADER_TREE_LOCATION)
static const kb_interface_t *s_romAPIInterface;
s_romAPIInterface = BOOTLOADER_API_TREE_POINTER->kbApi;
__cmse_nonsecure_entry status_t kb_init_NSE (kb_session_ref_t **session, const kb_options_t *options)
{
status_t status;
/*
```

Validate all input parameters based on application requirements. If input parameters are Invalid, return error

```

*/
status = s_romAPIInterface->kb_init(session, options);
return status;
}

```

Then user can call kboot init function from normal world as:

```

static kb_session_ref_t *context;
const kb_options_t options = {
    .version = kRomApiVersion,
    .buffer = user_buf,
    .bufferLength = USER_BUFFER_LENGTH,
    .op = kRomLoadImage,
    {
        .loadSB =
        {
            .profile = 0,
            .minBuildNumber = 1,
            .overrideSBBootSectionID = 1,
            .userSBKEK = 0,
            .regionCount = NULL,
            .regions = NULL,
        },
    }
};
status = kb_init_NSE (&context, &options);

```

19.8 Secure Boot usage

The RT500 allows booting of signed images. The device boot ROM supports following types of security protected boot modes:

- Secure boot with signed image
- Secure boot with signed image from encrypted external QSPI flash

Each of these options has attributes related to manufacturability, the firmware update scheme and level of protection against attacks.

The ROM further supports public keys and image revocation i.e. the method of not allowing new updates to be applied unless they are of a specific version. This is the basis for roll back protection.

The following section describes the main steps for key provisioning, creating signed images and loading the signed images into the target.

19.8.1 Keys and certificates

Image signing process will require RSA key pair and image signing certificate. Use e.g. openssl for key and certificate generation.

ROM supports:

- Up to 4 Root of Trust (RoT) keys

- Up to 16 Image key certificates with Image revocation feature

Prior the secure image preparation Root Key Hash table needs to be written to OTP fuses.

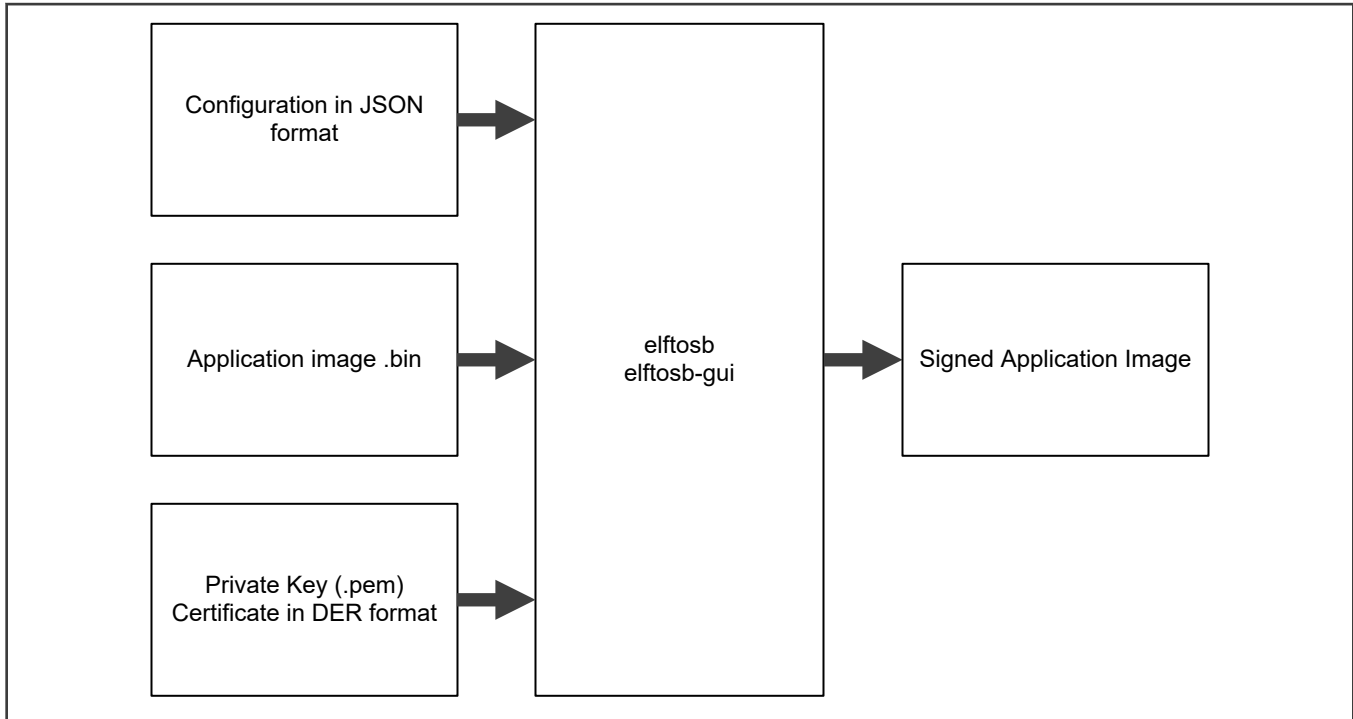
19.8.2 OTP fuses configuration preparation

Before the first use of the device certain OTP fuses must be programmed accordingly to be able to boot signed image. Here are the required steps:

- Prepare signed image using elftosb tool
- Enter ISP boot mode by asserting ISP boot pin
- Write prepared signed image into destination memory using blhost tool
- Program device configuration to fuses
- RKTH field containing root key table hash into OTP
- BOOT_CFG[0] and BOOT_CFG[5] field containing configuration of secure boot
- Power cycle the device to make new configuration to take effect
 - PRIMARY_BOOT_SRC to select the boot device
 - RSA4K field selecting minimal key length
 - SEC_BOOT_EN secure boot enable bit
 - USE_PUF to select if OTP or PUF key store will be used
 - TZM_IMAGE_TYPE to select TZ-M configuration of image
 - DICE_INC_OTP if OTP Fields should be included in DICE calculation
 - SKIP_DICE if DICE computation should be skipped

19.8.2.1 Signed image preparation

NXP provides elftosb and elftosb-gui tools which prepare signed binary, which can be loaded to target device. The input for elftosb program are plain application image in binary format, image signing certificate, associated private key and JSON format configuration structure.



Following information are needed by elftosb tool to produce Internal flash (XIP) signed image:

- Plain application binary generated for RT500 device
- Start address of application binary
- TZ-M related settings
- Certificates or chain certificates
- Private key for selected certificate (last certificate in chain)

19.8.2.2 Loading signed image

The signed image could be programmed directly into the device using various methods. The main options are:

- ROM In System Programming (ISP) using write-memory blhost command
- ROM ISP using Secure FW update container
- Programming signed image directly from target application using ROM API
- Flashing signed image through debugger

19.8.3 OTP configuration programming

19.8.3.1 RKTH

The RKTH value is generated by elftosb tool during signed image generation process. This value needs to be programmed into OTP fuses using blhost to be able to boot signed image. Here is an example how to program certain RKTH value word-by-word using blhost:

RKTH: 3619d0a5098a41b4528c1e5143fd4bd382fe4e31f47c0f9053c21b460803d2f6

```
blhost.exe -p COMx -- efuse-program-once 0x78 A5D01936
```

```
blhost.exe -p COMx -- efuse-program-once 0x79 B4418A09
```

```
blhost.exe -p COMx -- efuse-program-once 0x7A 511E8C52
```

```
blhost.exe -p COMx -- efuse-program-once 0x7B D34BFD43
blhost.exe -p COMx -- efuse-program-once 0x7C 314EFE82
blhost.exe -p COMx -- efuse-program-once 0x7D 900F7CF4
blhost.exe -p COMx -- efuse-program-once 0x7E 461BC253
blhost.exe -p COMx -- efuse-program-once 0x7F F6D20308
```

Note: The RKTH value generated by elftosb is in big-endian format. To store the value in OTP correctly, the byte order of the words supplied to efuse-program-once command needs to be byte-swapped to little-endian format.

19.8.3.2 BOOT_CFG[0] and BOOT_CFG[5]

BOOT_CFG[0] and BOOT_CFG[5] contains various fields related to device and secure boot configuration. Refer to the attached OTP Fuse Map.xlsx, [BOOT_CFG\[0\]](#) and [BOOT_CFG\[5\]](#) for bit fields description.

Example:

To program device to boot from external QSPI flash, using RSA4K only, TrustZone-M mode determined by the image header and OTP fields included in DICE calculation, write 0x00500101 into BOOT_CFG[0]. If PUF key store is used, also write 0x00000080 into BOOT_CFG[5] so that AES keys and UDS are read from PUF.

```
blhost.exe -p COMx -- efuse-program-once 0x60 00500301
blhost.exe -p COMx -- efuse-program-once 0x65 00000080
```

Note: During development, it is advised to use shadow register to test the secure boot path, and only blow the BOOT_CFG[0] fuse when the secure boot image is verified to be functional.

19.8.4 Key store setup

19.8.4.1 OTP key store

RT500 bootloader is configured to store keys in OTP fuses when USE_PUF bit field in BOOT_CFG[0] and BOOT_CFG[5] is set to zero. If OTP key store is selected, only OTP_MASTER_KEY and OTFAD_KEK_SEED values needs to be programmed into fuses. All of the keys used by bootloader are derived from these values. See [Boot keys derived from OTP_MASTER_KEY](#) for more details.

Please note prior to programming OTP_MASTER_KEY fuses, the KEY_SCRAMBLE_SEED shall be programmed.

To program OTP_MASTER_KEY use following commands:

```
blhost.exe -p COMx -- efuse-program-once 0x70 <MASTER_KEY[31:0]>
blhost.exe -p COMx -- efuse-program-once 0x71 <MASTER_KEY[63:32]>
blhost.exe -p COMx -- efuse-program-once 0x72 <MASTER_KEY[95:64]>
blhost.exe -p COMx -- efuse-program-once 0x73 <MASTER_KEY[127:96]>
blhost.exe -p COMx -- efuse-program-once 0x74 <MASTER_KEY[159:128]>
blhost.exe -p COMx -- efuse-program-once 0x75 <MASTER_KEY[191:160]>
blhost.exe -p COMx -- efuse-program-once 0x76 <MASTER_KEY[223:192]>
blhost.exe -p COMx -- efuse-program-once 0x77 <MASTER_KEY[255:224]>
```

To program OTFAD_KEK_SEED use following commands:

```
blhost.exe -p COMx -- efuse-program-once 0x6C <OTFAD_KEK_SEED[31:0]>
blhost.exe -p COMx -- efuse-program-once 0x6D <OTFAD_KEK_SEED[63:32]>
blhost.exe -p COMx -- efuse-program-once 0x6E <OTFAD_KEK_SEED[95:64]>
blhost.exe -p COMx -- efuse-program-once 0x6F <OTFAD_KEK_SEED[127:96]>
```

19.8.4.2 PUF key store

When USE_PUF bit field in BOOT_CFG[0] is set to one, bootloader will use keys stored in PUF key store, which can be part of application image or it can be stored in non-volatile memory (QSPI flash, MMC card). The key store data structure is located at address 0x800 of external non-volatile memory or at offset 0x64 of the application image if Key Store Included bit is set in image header for Load to Ram nonXip image type. Key store can be generated using elftosb gui or manually using following blhost commands:

PUF enroll (generate activation code into key store)

```
blhost -p COMx -- key-provisioning enroll
```

install SBKEK into key store. SBKEK key type = 3.

```
blhost -p COMx -- key-provisioning set_user_key 3 sbkek.bin
```

install ENC_IMAGE_KEY into key store. ENC_IMAGE_KEY key type = 11.

```
blhost -p COMx -- key-provisioning set_user_key 11 encimagekey.bin
```

generate random UDS; UDS key type = 12.

```
blhost -p COMx -- key-provisioning set_key 12 32
```

generate random OTFAD_KEK. OTFAD_KEK key type = 2

```
blhost -p COMx -- key-provisioning set_key 2 16
```

19.8.4.3 Storing PUF key store to NVM

In case that key store should be stored in external non-volatile memory use following command to save generated key store from previous chapter

```
blhost -p COMx -- key-provisioning write_key_nonvolatile <QSPI memory ID>
```

19.8.4.4 Storing PUF key store to application image

For load to RAM image types it is possible to include PUF key store into application image. It is stored at offset 0x64 and it's loaded automatically each time the image is booted. After the key store is generated by steps from [PUF key store](#), following command can be used to read it out to the host computer: `blhost -p COMx -- key-provisioning read_key_store key_store.bin`

The key store binary can be then included into application image using elftosb or elftosb-gui by selecting "Attach file" option in Key store section of the Image configuration area.

19.8.5 Upload image

Image upload process is dependent on type of destination memory. The following example will demonstrate how to configure, erase and program application image to external QSPI flash memory.

1. Configure external QSPI flash memory as explained in Non-Secure Boot chapter.
2. Erase flash before programming
 - `blhost -p COMX -t 500000 -- flash-erase-region 0x08001000 0x100000`
3. Program image into flash
 - `blhost -p COMx -- write-memory 0x08001000 signed_QSPI_xip.bin`

19.8.6 Secure ROM API

The ROM API table is located at address 0x1302f000 and contains absolute ROM API function addresses which can be called using function pointers. Secure ROM API section starts at address 0x13006094. Only secure boot related functions are described in this chapter. The main purpose of these APIs is to provide access to functions used and implemented in ROM to authenticate the application image.

Table 183. Secure ROM API

Address in Secure ROM API table	Absolute function address	Function name
0x13006094	0x1300e569	skboot_authenticate_api
0x13006098	0x13013d23	HASH_IRQHandler
0x1300609c	0x130235bd	skboot_otfad_encrypt

The secure APIs can also use the secure ROM API pointer to link to call the above three secure APIs, for example:

```
#define BOOTLOADER_TREE_LOCATION (0x1302f020)
bootloader_tree_t *romApiTree = (bootloader_tree_t *)BOOTLOADER_TREE_LOCATION;
skboot_authenticate_interface_t *s_authApiInterface;
s_authApiInterface = romApiTree->skbootAuthenticate;
void HASH_IRQHandler(void)
{
s_authApiInterface->skboot_hashcrypt_irq_handler();
}
```

For more information see ROM API layout section of Non-Secure Boot ROM chapter.

19.8.6.1 skboot_authenticate_api

Routine	skboot_authenticate_api
Prototype	skboot_status_t skboot_authenticate_api(const uint8_t *imageStartAddr, secure_bool_t *isSignVerified);
Input/Output parameter	imageStartAddr: The loaded address of the authenticated 1st boot image, or the SB 2.1 files isSignVerified: The output result of the image or SB file authenticated
Result	kStatus_SKBOOT_Success when signature verification pass kStatus_SKBOOT_Fail when parsing certificate header, certificate/certificates chain, RKH or signature verification fails kStatus_SKBOOT_InvalidArgument for unexpected value in the image
Description	This API function can be used to verify authenticity of an image. The ROM uses this function during the secure boot flow to authenticate 1st boot image, and it also uses it to verify authenticity of the SB 2.1 files.

NOTE

kb_init must be called beforehand; whereas, skboot_authneticate can be called directly instead of kb_execute.

19.8.6.1.1 RAM usage and Hashcrypt

If a user application calls skboot_authenticate() directly or indirectly from SB file processing functions kb_init/kb_execute/kb_deinit, the user HASH interrupt vector shall call the HASH_IRQHandler() function for handling of the Hash-crypt IP interrupt. This is due to the fact that the hashing is implemented as non-blocking for shorter computation time – while the Hash-crypt AHB master fetches data for hashing, the CPU and Casper co-processor work on RSA Verify. It is important to note that the skboot_authenticate() ROM API function uses global variables in RAM. Thus, the caller has to assure that it doesn't have any data

in the global variables location before the function call. The caller shall discard the data in the global variables location after the function returns. The ROM reserved space for global variables in RAM on this RT500 product is:

0x1001_2000 to 0x1001_A000

19.8.6.1.2 Memory alignment

The function requires the imageStartAddr input pointer to be 32-bit word aligned. Also, this function always use RKTH value stored in OTP fuses/shadow registers to authenticate the image. The user needs to make sure that valid RKTH value is present in OTP fuses/shadow registers before skboot_authenticate() can be executed. When image authentication is performed indirectly using kb_execute() function with "op" set to kRomAuthenticateImage, the "*userRHK" pointer is always ignored.

The status is returned in two ways - via a function return as well as by a write to the *isSignVerified pointer. This is provided for redundant protection, the caller shall verify both return values and consider authentic image only when the function returns kStatus_SKBOOT_Success AND *isSignVerified == kSECURE_TRACKER_VERIFIED.

19.8.6.1.3 Return values/error codes

On function output, it returns:

kStatus_SKBOOT_Success when signature verification pass

kStatus_SKBOOT_Fail when parsing certificate header, certificate/certificates chain, RKH or signature verification fails

kStatus_SKBOOT_InvalidArgument for unexpected value in the image

On function output, it writes

*isSignVerified = kSECURE_FALSE (0x5aa55aa5U) when signature verification fails

*isSignVerified = kSECURE_TRACKER_VERIFIED (0x55aacc33U) when signature verification pass

19.8.6.2 HASH_IRQHandler

Table 184. HASH_IRQHandler

Routine	HASH_IRQHandler
Prototype	void HASH_IRQHandler(void);
Description	This function shall be called from user Hash-crypt interrupt handler, if the skboot_authenticate() or kb_process() ROM API function is called in user applications. The hashing in ROM is implemented as non-blocking and so the interrupt handler function is required to assist with fetching data for Hash-crypt hardware module.

19.8.6.3 skboot_otfad_encrypt

Routine	skboot_otfad_encrypt
Prototype	skboot_status_t skboot_status_t skboot_otfad_encrypt(skboot_otfad_config_t *otfadConfig, uint32_t writeAddress, uint32_t *bufferIn, uint32_t *bufferOut, size_t bufferSize);
Input/Output parameter	otfadConfig: Input configuration of the OTFAD.

Table continues on the next page...

Table continued from the previous page...

Routine	skboot_otfad_encrypt
	writeAddress: Destination flash address. bufferIn: Input buffer with plaintext data bufferOut: Output buffer with ciphertext data bufferSize: bufferIn/bufferOut byte length, shall match one or multiple of flash program page size.
Result	kStatus_SKBOOT_Success when signature verification pass kStatus_SKBOOT_Fail when parsing certificate header, certificate/certificates chain, RKH or signature verification fails kStatus_SKBOOT_InvalidArgument for unexpected value in the image
Description	Caller has to supply otfadConfig, it has to be known. The function encrypts bufferIn[] plaintext data to bufferOut[] ciphertext data. Currently the function can only work if the OTFAD region is erased.

skboot_otfad_config_t is defined as follows:

```
typedef struct _skboot_otfad_config
{
    struct
    {
uint32_t startaddr;
uint32_t endaddr;
    } otfadBlob[4];
    } skboot_otfad_config_t;
```

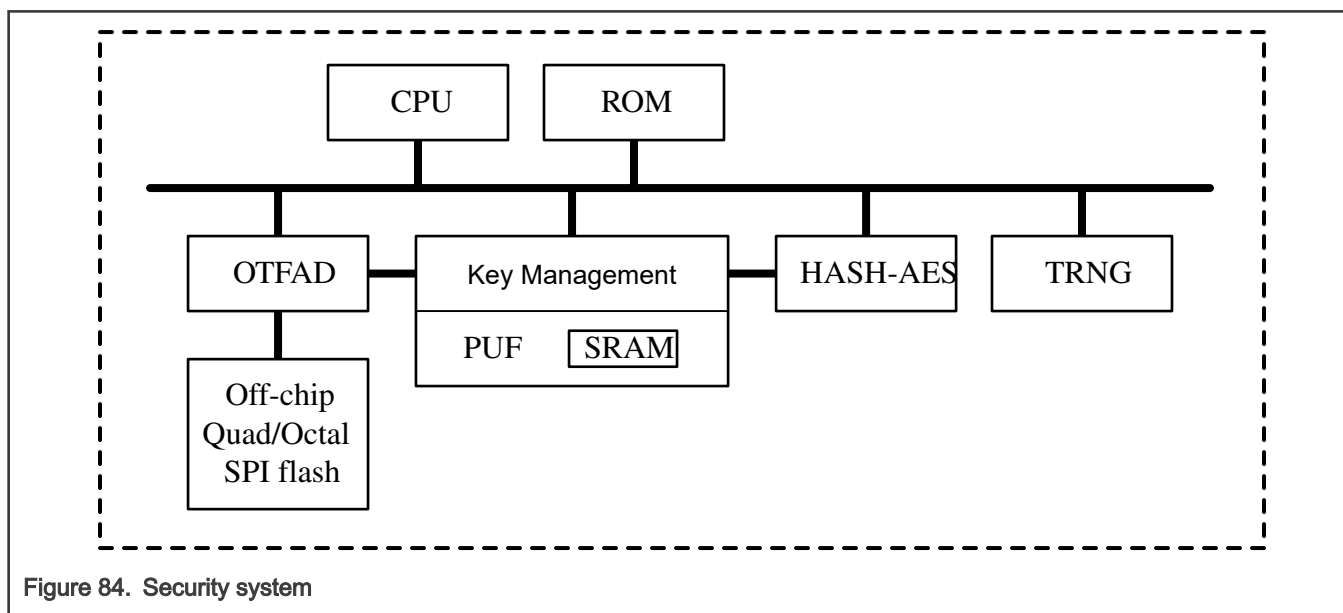
The skboot_otfad_encrypt() function verifies validity of supplied otfadConfig. This function will perform the encryption if flash destination address (writeAddress) and data size (bufferSize) fits within one of the four OTFAD regions.

Chapter 20

Security

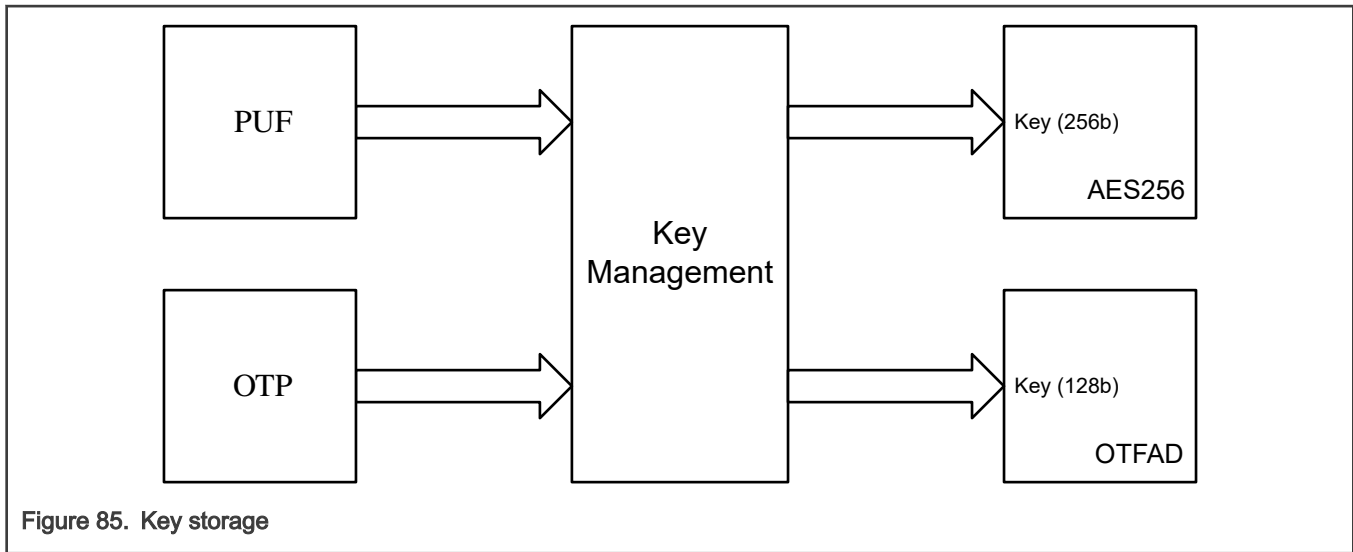
20.1 Introduction

The security system has a set of hardware blocks and ROM code to implement the security features of the device. The hardware consists of an AES engine, a Secure Hash Algorithm (SHA) engine (Hash-AES block), a Random Number Generator (RNG), and a key storage block that keys from an SRAM-based PUF (Physically Unclonable Function). The Security system diagram shows an overview of the security system. All components of the system can be accessed by the processor or the DMA engine to encrypt or decrypt data and for hashing. The ROM is responsible for secure boot in addition to providing support for various security functions.



20.1.1 Key storage/management

A critical feature of any security system is how keys are stored and managed. Keys can be used for boot loading and handling of critical user data. This device offers SRAM PUF where the PUF provides a unique key per device and exists in that device based on the unique characteristics of PUF SRAM. The Key storage figure shows the block diagram of how keys are used by the AES engine. PUF keys have a dedicated path to the AES engine and OTFAD where only the intended target can make use of its key. There is no other mechanism by which keys can be observed. KEY0 feeds into AES, KEY1/2/3 from Key Management block feed into OTFAD.



20.1.1.1 PUF keys

The PUF controller provides secure key storage without storing the key. This is done by using the digital fingerprint of a device derived from SRAM. Instead of storing the key, a Key Code is generated, which in combination with the digital fingerprint is used to reconstruct keys that are routed to the AES engine or for use by software. The PUF controller provides generation and secure storage for keys.

20.2 AES engine

The device provides an on-chip hardware AES encryption and decryption engine to protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin. Data can be encrypted or decrypted by the AES engine using a key from the PUF or a software supplied key.

ICB-AES mode exists to provide a counter measure to Side Channel Analysis.

20.3 SHA

The device provides on-chip Hash support for SHA-1 and SHA-2 with 256-bit digest (SHA-256). Hashing is a way to reduce arbitrarily large messages or code images to a relatively small fixed size "unique" number called a digest. The SHA-1 Hash produces a 160 bit digest (five words), and the SHA-256 hash produces a 256 bit digest (eight words).

For the SHA hardware:

- Even a small change to the input message causes a major change in the digest output. Therefore, for a given input message or image there is only one digest.
- There is no predictable way to modify one input to result in a specific digest. A message cannot be added, inserted, or modified to get the same Hash in any direct way.

These two properties make it useful for verifying that a message is valid, or that it has been corrupted intentionally or unintentionally.

Hashing is used for four primary purposes:

- Core of a digital signature model, including certificates, for example and for secure update.
- Support a challenge/response or to validate a message when used with a Hash-based Message Authentication Code (HMAC).
- In a secure boot model, which verifies code integrity.
- Verify that external memory has not been compromised.

20.4 Digital signatures

A digital signature combines public/private keys such as RSA or ECC with SHA Hashing. A signature is formed in the following way:

- The message or image is hashed using SHA1, SHA2, or other function.
- The digest is formed into a buffer along with a header and padding. The header indicates what hashing was used (for example, SHA1). The padding fits the buffer to the size of the public key, for example, 256 bits, 1024 bits, and 2048 bits.
- The buffer is signed (encrypted) using the private key. Note that signing is a reverse of the normal public or private key where anyone can encrypt using the public key and therefore, only the private key holder can decrypt. In this case, the model is reversed so that only a private key holder can sign, and anyone can verify the signature if it is from the private key holder.

To verify the signature, the following steps are used:

- The message or image is hashed using SHA1, SHA2. The output of the hash must match the signature.
- The signature is decrypted using the public key.
- The hash digest is compared against the stored digest in the buffer after decryption.

The advantage of the signature model is that the public key and the signature can be public. Therefore, the signature can be stored in an external flash file along with the image and then verified using a stored copy of the public key.

20.5 Hash-based Message Authentication Code (HMAC)

An HMAC can be achieved using a pre-shared key and hashing. HMAC is a way of verifying a message (encrypted or not) and can also be used for challenge or response. Both sides must have a pre-shared key that is just a shared secret value and not an encryption key.

The HMAC is formed using three steps:

- Hashing the message or image
- Taking the resultant digest and combining it with the key and some padding
- Hashing the combination of digest, key, and padding. The resultant digest is sent.

On the other side, the same procedure is followed to verify and get the same digest. Only those parties that know the key can get the correct digest and therefore, can trust the data that was hashed.

HMACs are significantly faster than signatures, but work only with pre-shared keys, which must not be leaked or lost (unlike a public key). The HMAC key can be shared dynamically using trust models like Diffie-Hellman or maybe a board-unique key shared by two devices.

20.6 RNG

Random Number Generators (RNG) are used for cryptographic, modeling, and simulation applications, which employ keys that must be generated in a random fashion.

20.7 UUID

The device stores a 128-bit IETF RFC4122 compliant non-sequential Universally Unique Identifier (UUID), which can be read from the System Controller registers (SYSCTL0), UUIDn.

20.8 DICE

The device supports Device Identifier Composition Engine (DICE) to provide Composite Device Identifier (CDI). A CDI value would be available in the System Controller registers (SYSCTL0), DICEHWREGn for consumption after boot completion. It is recommended to overwrite these registers once ephemeral key-pairs are generated using this value.

20.9 On-The-Fly AES Decryption (OTFAD)

The OTFAD function provides AES-128 Counter Mode On-the-Fly Decryption of external data located on the Quad/Octal SPI flash interface (QuadSPI) interface.

20.10 RT500 Life cycle state

20.10.1 Introduction

This chip supports a security life cycle state model. The current life cycle state determines the device functionality, debug and test port availability, and asset accessibility. The life cycle state is implicitly derived from the values programmed in different OTP Fuse fields. Because fuses control the life cycle state, moving to a more advanced state is an irreversible and permanent process. The life cycle can only be advanced and cannot return to a previous state.

The Boot ROM is responsible for checking the life cycle state. Based on the life cycle state, the ROM determines what boot flow is used, including if control is passed to application code or not. The ROM also handles the opening of test and debug ports based on the life cycle state.

20.10.2 Life cycle state

Customers will initially use the RT6xx/RT5xx device in the “NXP closed” lifecycle state. Depending on application security targets, customers can ship the RT6xx/RT5xx in this “NXP closed” state, or, move it to the “OEM closed” state, where the user programmable domain will require an authenticated image to boot.

The “Returned” state (a.k.a. FA mode) is used for retiring the RT6xx/RT5xx, not before erasing various secrets that were provisioned, by NXP and Tier1/2/OEM. This state will allow running FA testing on customer returns. The PUF output is completely blocked and the part doesn't boot any images. It only provides debug access.

The details on each of the states (e.g. OTP fields that needs to be programmed, impact on each subsystem) and transitioning between states (OTP fields that need to be programmed) are provided in the table below:

Life cycle state	Description	OTP Fields to provision	Transitions		Transition number
			Fields to Write	Next state	
NXP Closed Or OEM Factory	Initial customer state: - Test / Debug ports opened by ROM. But debug access is disabled during ROM execution. - If debug authentication fields are programmed then they are used to determine debug access. - ISP interface supports all commands. - In this mode SW could write to shadow registers of the customer programmable	Customer programmable OTP fields : Boot Configuration: - BOOT_CFG[0] - Quad SPI settings (BOOT_CFG[1]) - SD/MMC boot settings (BOOT_CFG[2])	Except for Debug Authentication (DCFG_CC_S OCU, DCFG_CC_SO CU_AP, DCFG_CC_SO CU_NS), if any other fuses are programmed.	Tier 1 Dev	Transition 1

Table continues on the next page...

Table continued from the previous page...

Life cycle state	Description	OTP Fields to provision	Transitions		Transition number
			Fields to Write	Next state	
	fuses to try different boot options. If debug authentication fields (CC_SOCU_XXX) fields are not in default state then limited ISP commands are allowed	- Pad and remap settings (BOOT_CFG[3]) Secure boot configuration: - ROTKH[255:0] - SECURE_BOOT_CFG OTFAD: - OTFAD - configuration - OTFAD_KEK Debug Authentication: - DCFG_CC_SOCU - DCFG_CC_SOCU_AP - DCFG_CC_SOCU_NS PUF Keys store: - Activation code - OTFAD KEK key - UDS key - SBKEK - User KEK Optional: - USB_ID			
Tier 1 Dev Or OEM development	Tier 1 software development state:	Customer factory programmed fields: Secure boot:	DCFG_CC_SOCU and DCFG_CC_SOCU_AP are programmed	Tier 2 Dev	Transition 2

Table continues on the next page...

Table continued from the previous page...

Life cycle state	Description	OTP Fields to provision	Transitions		Transition number
			Fields to Write	Next state	
	• Test / Debug ports opened by ROM. But debug access is disabled during ROM execution. • If debug authentication fields (CC_SOCU_xxx) are programmed then they are used to determine debug access. • SECURE_BOOT_CFG field determines whether secure boot flow is enabled or not. • If debug authentication fields (CC_SOCU_xxx) fields are not in default state then limited ISP commands are allowed	• ROTKH[255:0] • BOOT_CFG • SECURE_BOOT_CFG PRINCE: • PRINCE_BASE_ADDR • PRINCE_SR_0 • PRINCE_SR_1 • PRINCE_SR_2 Debug Authentication: • CC_SOCU_PIN • CC_SOCU_DFLT • VENDOR_USAGE PUF Keys store: • Activation code • PRINCE0 key • PRINCE1 key • PRINCE2 key • UDS key • SBKEK • User KEK Optional: • USB_ID • Customer defined fields	but DCFG_CC_SOCU_NS is left blank.		

Table continues on the next page...

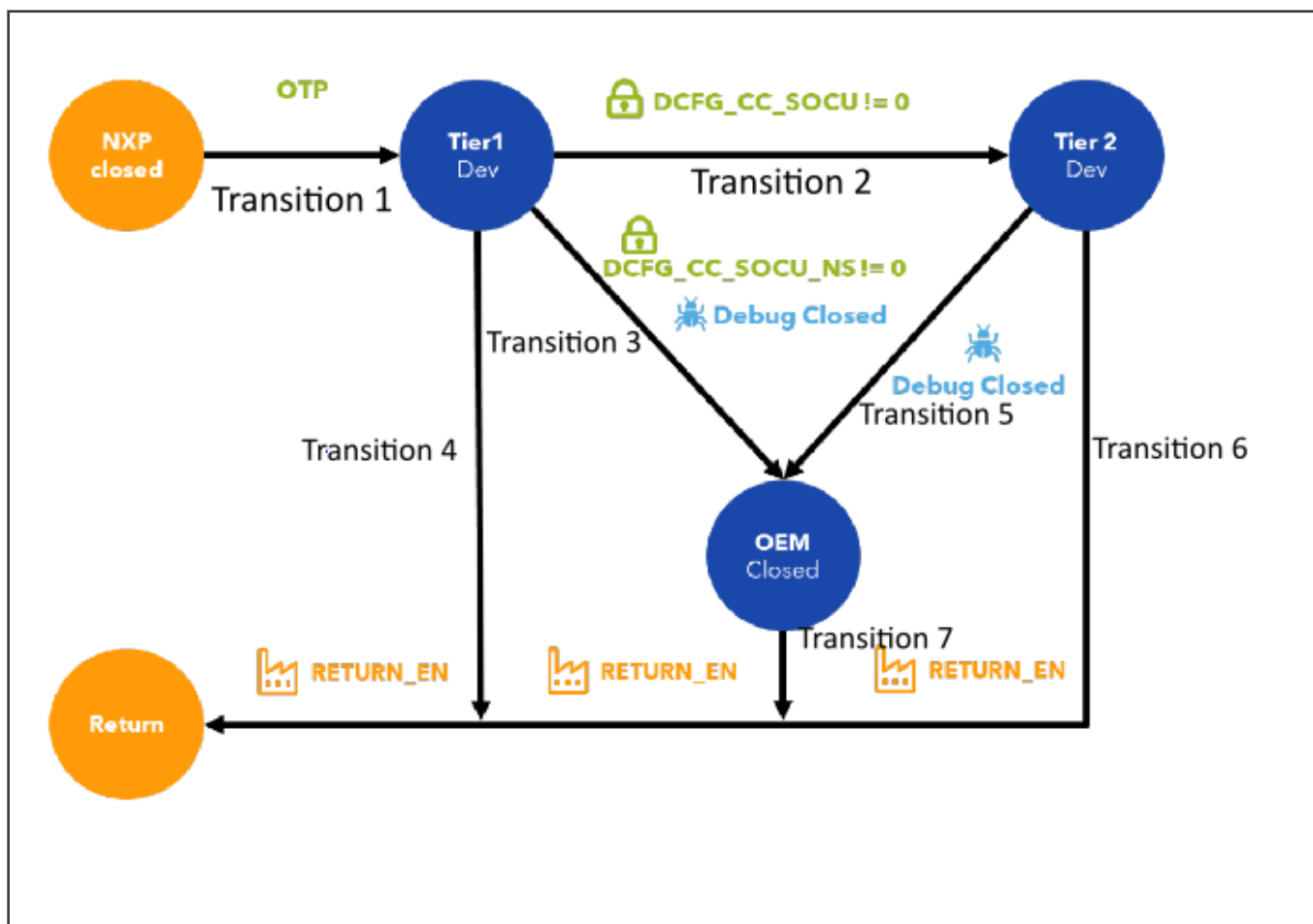
Table continued from the previous page...

Life cycle state	Description	OTP Fields to provision	Transitions		Transition number
			Fields to Write	Next state	
	CRC_BANK, CUST_WR_RD_LOCK0 and CUST_WR_RD_LOCK1 should be programmed appropriately before advancing the state to "OEM closed".	Same as above.	DCFG_CC_SO CU, DCFG_CC_SO CU_AP and DCFG_CC_SO CU_NS are programmed.	OEM Closed	Transition 3
		Same as above.	FA_MODE_EN bit is set in SEC_BOOT_CFG[5].	Returned	Transition 4
OEM Tier 2 Dev	Tier 2 software development state: • Test/Secure Debug ports closed (unless authenticated) or always closed • Non-secure Debug enabled • TZ-M is enabled by primary image. • limited ISP commands are allowed	DCFG_CC_SOCU_NS	DCFG_CC_SO CU_NS is programmed	OEM Closed	Transition 5
			FA_MODE_EN bit is set in SEC_BOOT_CFG[5].	Returned	Transition 6
OEM Closed Or Deployed	In-field Application State: • Test/Debug ports closed (unless authenticated) or always closed • TZ-M optional • Limited ISP commands are allowed unless disabled in OTP	SECURITY_EPOCH fields are programmable	FA_MODE_EN bit is set in SEC_BOOT_CFG[5].	Returned	Transition 7
Returned Or Decommission Or FA mode	Returned (CQI): • ROM checks key store is empty and blocks PUF key unwrapping functionality	N/A			

Table continues on the next page...

Table continued from the previous page...

Life cycle state	Description	OTP Fields to provision	Transitions		Transition number
			Fields to Write	Next state	
	before enabling Test/Debug ports open. • ROM doesn't boot images in flash but stays in while(1) loop. • Part can only be used to run test patterns through JTAG or load code to RAM and run using SWD interface. • ISP command interface is also disabled.				



Chapter 21

TrustZone Support

21.1 Overview

TrustZone is a security feature on ARM Cortex series that enables execution separation of trusted (secure) software, and enables access control isolation of trusted resources from non-trusted (non-secure) software and resources, while running on the same CPU. Memory arrays and peripherals are segmented into either Secure (S) or Non-Secure (NS). TrustZone is optimized for energy-efficient, embedded applications that require real-time responsiveness.

More details on TrustZone can be found in “TrustZone® technology for ARM®v8-M architecture version 1.0” document at arm.com.

NOTE

"S" = Secure; "NS" = Non-Secure.

If TrustZone functionality is enabled, then the following restrictions apply to the cores:

- The CM33 CPU in a Secure state (CPU-S) can execute instructions from Secure memory (S-memory) only; the CPU cannot execute instructions from Non-Secure memory (NS-memory).
- CPU-S can access data in both S-memory and NS-memory, which means that CPU-S can execute data reads from both S- and NS-memory, and also execute data writes to S or NS-memory.
- CPU-NS can execute instructions only from NS-memory, and cannot execute instructions from S-memory.
- CPU-NS can access data only in NS-memory, which means that CPU-NS can execute data reads from NS-memory only, and can execute data writes to NS-memory only. CPU-NS cannot access data from S-memory.

In summary:

- NS application code “trusts” that secure code will not corrupt or modify NS code (or data) inadvertently (or on purpose) to create malfunctions or hazards.
- S application code does not “trust” NS application code and disallows access to a CPU-NS.

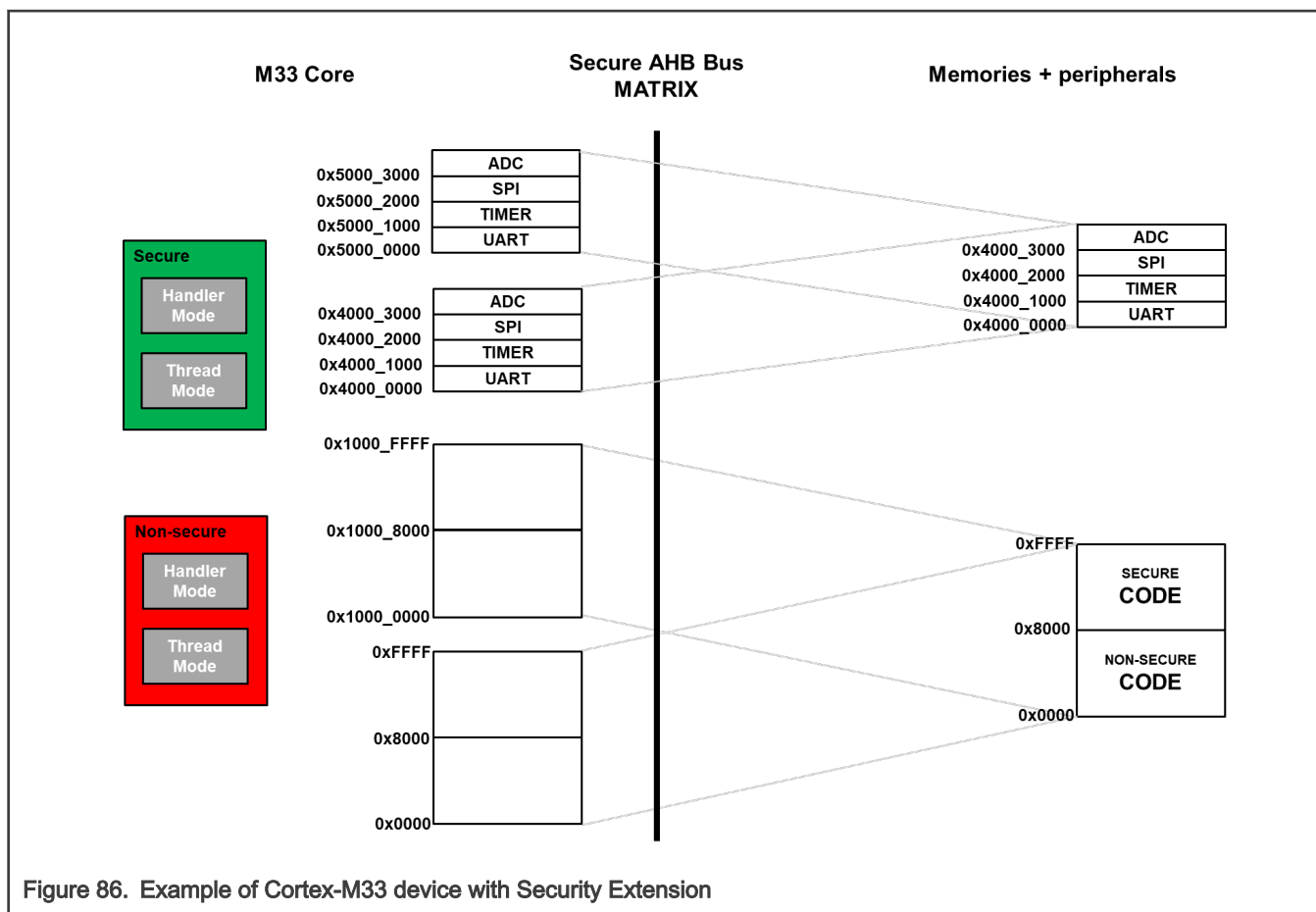
Secure state Cortex-M33 architecture extends to include secure MPU, secure NVIC, secure SysTick, and secure stack pointer with stack-threshold check.

21.2 Example Configuration

There are two layers of TrustZone protection implemented. The first layer consists of two attribution units on the CM33 core:

- SAU (Security Attribution Unit)
- IDAU (Implementation Defined Attribution Unit)

The second layer consist of secure AHB bus and AHB secure controller implemented on the system level. The proper configuration of both layers is essential for secure trusted environment execution. The figure below shows an example of TrustZone configuration. The simplified device consists of CM33 core with security extension running in secure or normal mode, 64KB of code memory, 64KB data memory and four peripherals. The peripherals, code and data memory are connected to the core via secure AHB bus. Due to memory address aliasing (using address bit A28) the CM33 core sees all memories and peripherals twice in the memory map. For example, the 64KB code memory can be seen either as 0x0000 – 0xFFFF or 0x1000_0000 – 0x1000_FFFF. To keep figures simple, the data memory (0x2000_0000 – 0x2000_FFFF, 0x3000_0000 – 0x3000_FFFF) is not shown on all figures.



21.2.1 Memory Layout after TrustZone configuration

Before TrustZone configuration, the memories and peripherals that will be assigned to secure and non-secure environment need to be defined. In the above example the trusted execution environment will be defined as:

- The first 32KB of code memory (0x0000 – 0x7FFF) as non-secure memory
- The second 32KB of code memory (0x8000 – 0xFFFF) as secure memory
- The 48KB of data memory (0x2000_0000 – 0x2000_BFFF) as non-secure
- The 16KB of data memory (0x2000_C000 – 0x2000_FFFF) as secure memory
- SPI and TIMER as secure peripherals
- UART and ADC as non-secure peripherals

The rest of address space remains secure.

After proper TrustZone configuration the MCU resources will be available on following addresses:

Table 185. Example memory layout after TrustZone configuration

MCU resource	Address range	Security attribute
Non-secure code memory	0x0000_0000 – 0x0000_7FFF	Non-secure
Secure code memory	0x1000_8000 – 0x1000_FFFF	Secure
Non-secure data memory	0x2000_0000 – 0x2000_BFFF	Non-secure

Table continues on the next page...

Table 185. Example memory layout after TrustZone configuration (continued)

MCU resource	Address range	Security attribute
Secure data memory	0x3000_C000 – 0x3000_FFFF	Secure
UART	0x4000_0000 – 0x4000_0FFF	Non-secure
TIMER	0x5000_1000 – 0x5000_1FFF	Secure
SPI	0x5000_2000 – 0x5000_2FFF	Secure
ADC	0x4000_3000 – 0x4000_3FFF	Non-secure

21.2.2 Basic SAU Configuration

The TrustZone configuration starts with secure attribution map definition, which splits memories and peripherals between secure and non-secure domain. The security attribution map is defined by IDAU and SAU. The default security attribution map is defined by IDAU and it can be modified by SAU configuration. If the SAU is disabled or empty (none of SAU region is configured and enabled) the whole address space is secure. To assign some address space into non-secure domain, this address space must be configured in the SAU and concurrently the same address space must be marked as non-secure in IDAU. It means that address space marked as secure in IDAU can never be assigned to non-secure domain.

The simplest way to set up SAU is to configure every non-secure contiguous address space as one region in SAU. Using this approach, the SAU will be configured as follows:

Table 186. Basic SAU Configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (UART)	0x4000_0000	0x4000_0FFF	0	0
Region 4 (TIMER)	0x4000_3000	0x4000_3FFF	0	0

The security attribution map after SAU configuration can be seen in the figure below. The SAU configuration also includes 1KB of secure, non-secure callable (NSC) memory placed on the top of secure memory. This region is used for veneer table. The veneer table contains all secure functions/services, which are callable from non-secure world. The NSC region is not shown on all figures as well as data memory.

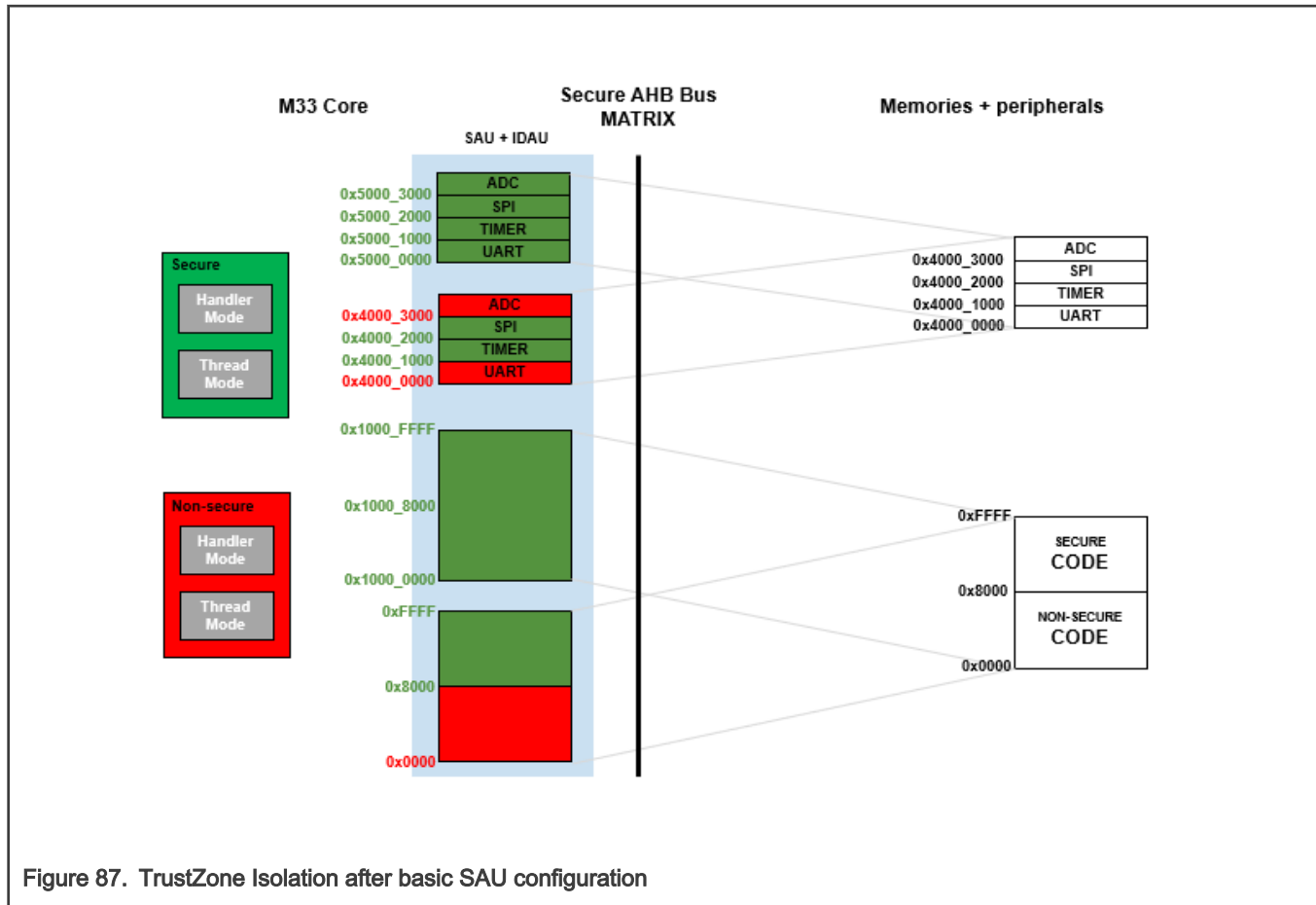


Figure 87. TrustZone Isolation after basic SAU configuration

After basic SAU configuration, the TrustZone isolation is fully functional, but there are following limitations:

- Every contiguous memory space requires one SAU region. Since there are 8 SAU regions only, this approach is suitable for simple applications.
- Since IDAU/SAU manages CM33 core transactions only, there is no any information about TrustZone isolation on system level. It means that TrustZone isolation is unknown to other bus masters in the system like DMA, USB and others.
- TrustZone isolation relies on SAU configuration only. In case of wrong SAU configuration (due to software error or glitch attack), TrustZone isolation is corrupted also.

Due to these limitations, this way of trusted execution environment configuration (based on basic SAU set up only) is not recommended. A second layer protection using the AHB Secure Controller must be employed. Second protection layer brings much higher flexibility in TrustZone configuration and higher isolation security.

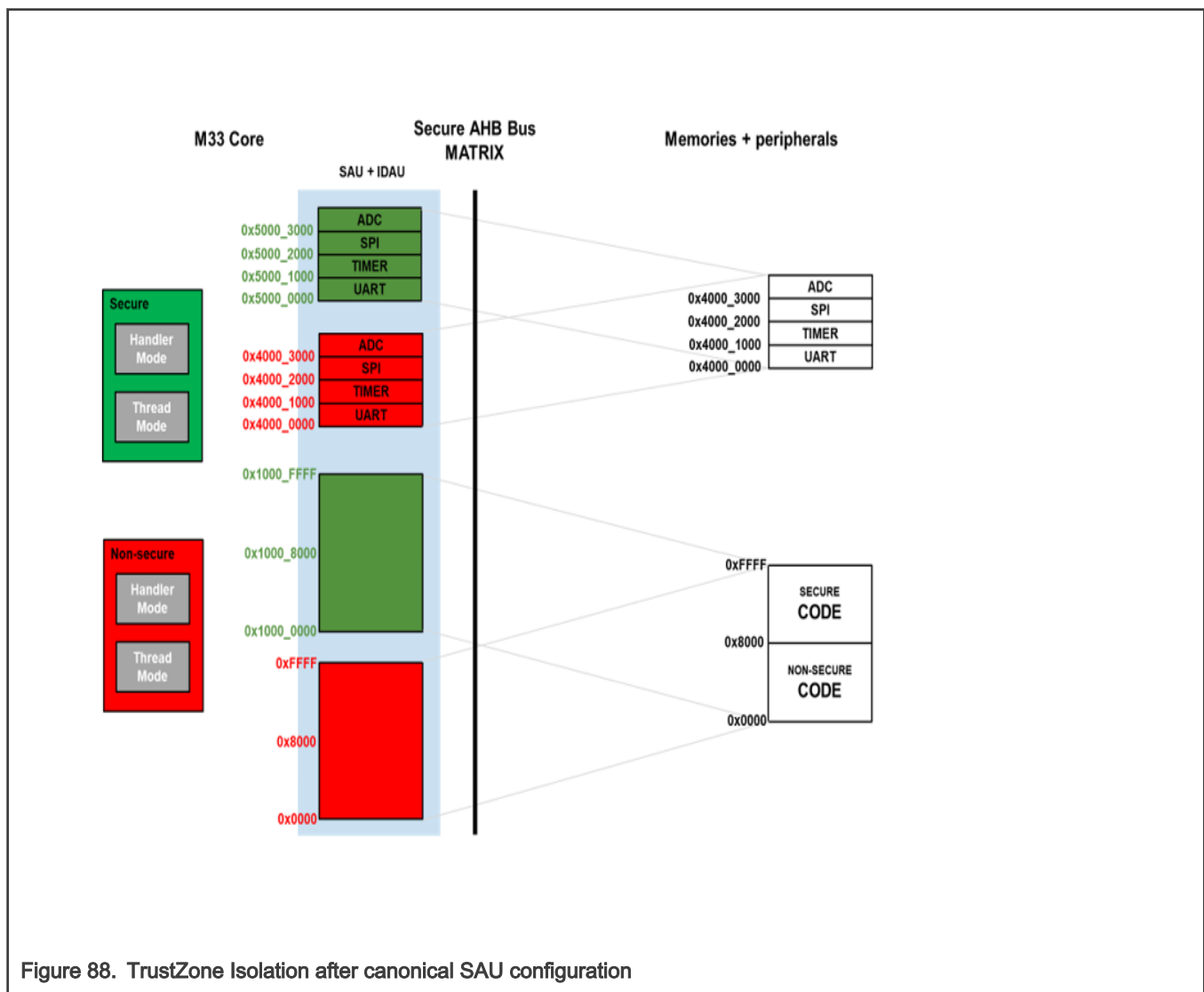
21.2.3 Canonical SAU Configuration

Real applications are much more complex than presented example. Especially peripherals configuration can be challenging with 8 SAU regions only. Therefore, different approach for SAU configuration must be chosen. This approach is also called canonical SAU configuration. It lies on the principle, that all memories and peripherals have one secure and one non-secure alias. Another way said, every region in standard Cortex-M address space is divided into halves, where address space with address bit A28=1 means secure address and with A28=0 means non-secure address. The canonical SAU configuration can be seen in the table below.

Table 187. Canonical SAU Configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 0 (code memory)	0x0000_0000	0x1FFF_FFFF	0	1
Region 1 (data memory)	0x2000_0000	0x5FFF_FFFF	0	1

The canonical SAU configuration requires two regions only so there are still six SAU regions available for other purposes, for example for NSC regions definition. The presented example, configured using canonical SAU configuration, can be seen in the figure below.



Comparing TrustZone Isolation after basic SAU configuration and TrustZone Isolation after canonical SAU configuration, it can be seen, that TrustZone isolation is not functional now. For example, secure code in address range 0x1000_8000 – 0x1000_FFFF is also available in non-secure mode in address range 0x8000 – 0xFFFF. This is because the SAU configuration is visible on core level only and thus system cannot know, whether address range 0x8000 – 0xFFFF is assigned to secure or non-secure domain.

Moreover, the address range 0x8000 – 0xFFFF is assigned to both domains on the core level. To make TrustZone isolation functional, the AHB secure controller must be configured and enabled.

21.2.4 TrustZone Isolation after Canonical form of SAU and AHB Secure Controller Configuration

After proper AHB secure controller configuration the TrustZone isolation is fully functional. The non-secure software can still issue non-secure transaction with address 0x8000, but this transaction is blocked by AHB secure controller because the memory with physical address 0x8000 is already assigned to secure domain. It means that secure code is accessible via secure alias (0x1000_8000 – 0x1000FFFF) only.

Another effect of memory and peripheral checkers is also higher configuration flexibility. At canonical SAU configuration, user is not limited by number of SAU regions and flexibility is enhanced up to memory sub-region or peripheral level. Besides protection checkers, the AHB secure controller implements simplified IDAU for all non-core bus master (MSW) so they can also benefit from TrustZone isolation.

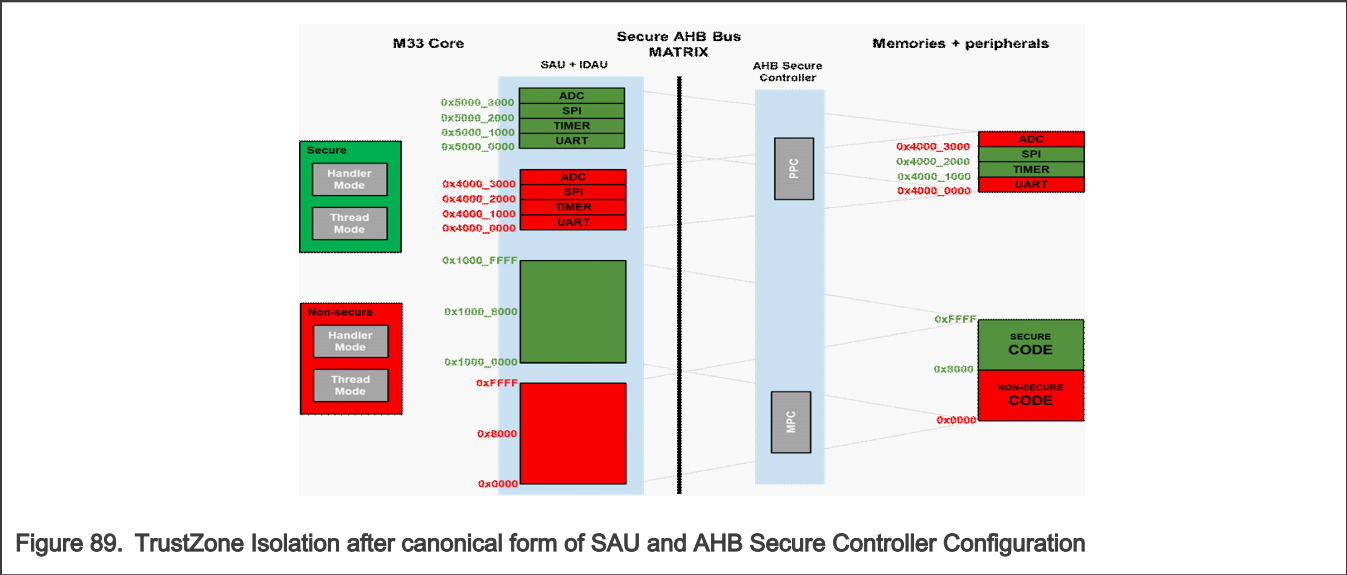


Figure 89. TrustZone Isolation after canonical form of SAU and AHB Secure Controller Configuration

21.2.5 Combined SAU and AHB Secure Controller Configuration

The last SAU configuration example combines both previous approaches together with AHB secure controller. The advantage of basic SAU configuration together with enabled AHB secure controller is the double checking each bus transaction. The first check is done on core level (SAU/IDAU), and the second check is performed on the system level (AHB secure controller). If some inconsistency is detected between SAU and AHB secure controller configurations (due to some software error or glitch attack), the access to the specific resource is blocked.

So, employing both SAU and AHB secure controller for TrustZone isolation makes isolation more robust compared to basic SAU configuration. The TrustZone example shown in figure below uses basic SAU configuration for code and data memories, and canonical SAU configuration for peripherals. This combination achieves the highest isolation robustness while allowing sufficient flexibility even for complex applications.

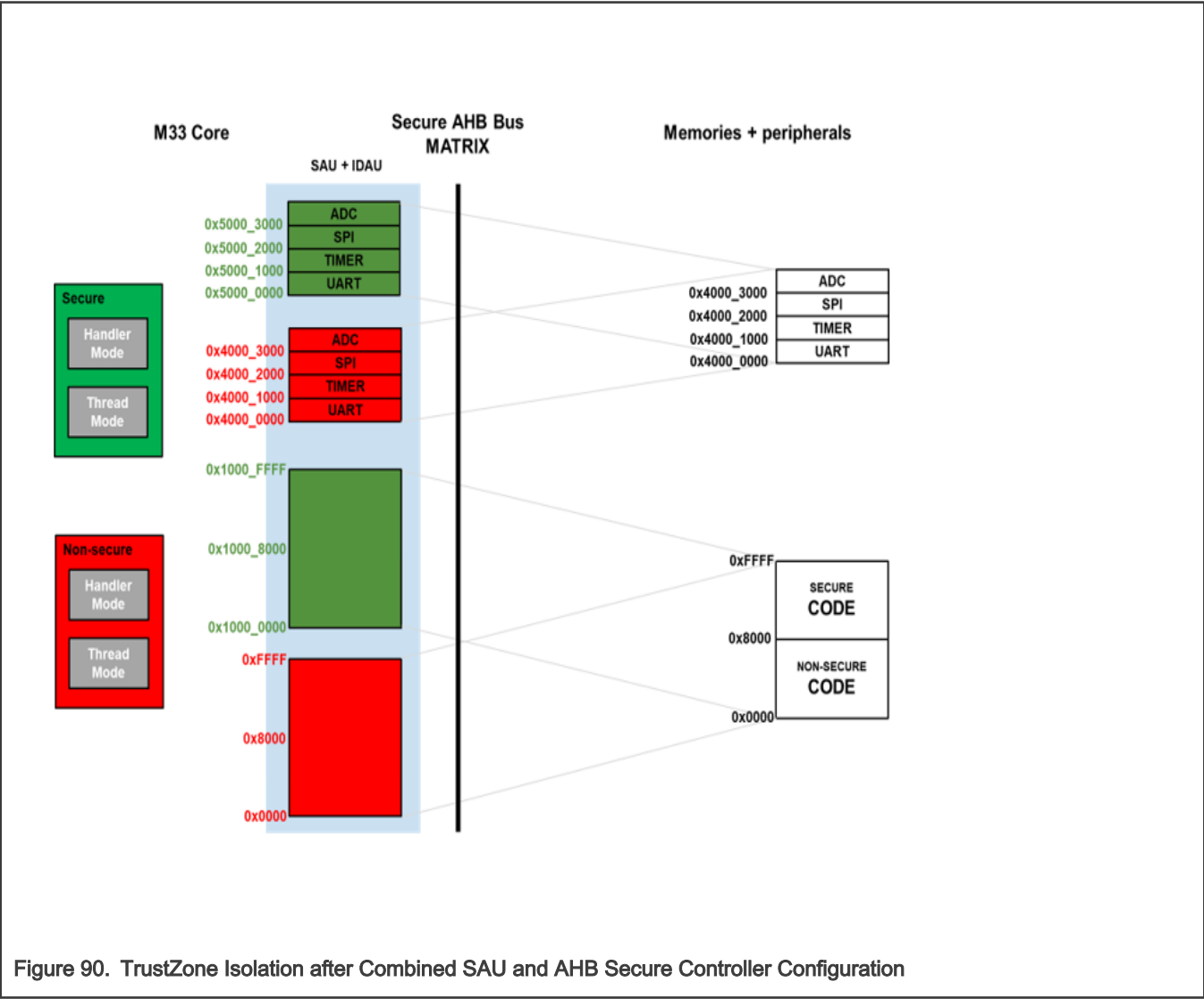
Table 188. Combined SAU Configuration

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 0 (code memory)	0x0000_0000	0x0000_7FFF	0	1

Table continues on the next page...

Table 188. Combined SAU Configuration (continued)

	RBAR	RLAR		
	Base Address	Limit Address	NSC	ENABLE
Region 1 (NSC memory)	0x1000_FC00	0x1000_FFFF	1	1
Region 2 (data memory)	0x2000_0000	0x2000_BFFF	0	1
Region 3 (peripheral memory)	0x4000_0000	0x5FFF_FFFF	0	1



Chapter 22

AHB Secure Controller (AHBSC)

22.1 Chip-specific AHB Secure Controller information

Table 189. Reference links to related information

Topic	Related module	Reference
Full description	AHB Secure Controller	AHB Secure Controller
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

22.1.1 Module instances

This device has one instance of the AHB Secure Controller module.

22.2 Overview

The AHB Secure Controller enables the programming of the security attributes for all MPCs (Memory Protection Checkers), PPCs (Peripheral Protection Checkers), and MSWs (Master Security Wrappers). AHBSC supports the locking of SAU (Security Attribution Unit) settings, secure and non-secure MPU (Memory Protection Unit) settings (MPU_S/MPU_NS), secure and non-secure vector offset settings (VTOR_S/VTOR_NS) for CPU0. This enables the BootROM to safeguard certain security features, and prevents the possibility of enabling those settings dynamically, either unintentionally or maliciously.

22.2.1 Features

Features:

- CPU0 is a core (Cortex-M33) with TrustZone support enabled
- DSP is a core (Cadence Xtensa Fusion F1 Audio DSP CPU)
- Attribution Units: SAU (Security Attribution unit), IDAU (Device Attribution Unit)
- Secure MPU, Secure NVIC, Secure SYSTICK, Secure Stack Pointer
- Secure memory map aliasing
- Support for ARM AMBA 5.0 AHB secure bus
- Secure bus controller
- Memory and peripheral protection checkers
- Security attribution wrapper for AHB masters
- Interrupt masking
- Secure DMA and DMA masking
- Secure GPIO and GPIO masking
- Secure debug

22.3 Functional description

The AHB Secure Controller supports register programming for GPIO masking and interrupt masking. When a security violation is detected, an interrupt is raised by the AHB Secure Controller module. The AHB Secure Controller also logs violation information, like the address being accessed on a AHB slave port when the violation occurred, as well as the access type and security attributes of the master that generated the unauthorized access.

Only an application that can write to the AHB Secure Controller to configure security attributes is the application that has Secure and Privilege access rights. From an application perspective, the highest tier (tier-4) thread from CPU0 can program security attributes for system slaves. Registers programmed in the AHB Secure Controller are retained during deep-sleep and power-down modes; however, after wake-up from deep power-down mode, these AHB Secure Controller registers must be re-programmed.

22.4 Memory Map and Registers

This section includes the AHBSC module memory map and detailed descriptions of all registers.

22.4.1 AHBSC register descriptions

22.4.1.1 AHBSC memory map

AHB_SECURE0 base address: 4014_8000h

Offset	Register	Width (In bits)	Access	Reset value
10h - 1Ch	Memory ROM Rule(n) Register (ROM_MEM_RULE0 - ROM_MEM_RULE3)	32	RW	0000_0000h
30h - 3Ch	FLEXSPI0 Region 0 Rule(n) Register (FLEXSPI0_REGION0_RULE0 - FLEXSPI0_REGION0_RULE3)	32	RW	0000_0000h
40h	FLEXSPI0 Region 1 Rule 0 Register (FLEXSPI0_REGION1_RULE0)	32	RW	0000_0000h
50h	FLEXSPI0 Region 2 Rule 0 Register (FLEXSPI0_REGION2_RULE0)	32	RW	0000_0000h
60h	FLEXSPI0 Region 3 Rule 0 Register (FLEXSPI0_REGION3_RULE0)	32	RW	0000_0000h
70h	FLEXSPI0 Region 4 Rule 0 Register (FLEXSPI0_REGION4_RULE0)	32	RW	0000_0000h
90h - 9Ch	SRAM Partition 00 Rule(n) Register (RAM00_RULE0 - RAM00_RULE3)	32	RW	0000_0000h
A0h - ACh	SRAM Partition 01 Rule(n) Register (RAM01_RULE0 - RAM01_RULE3)	32	RW	0000_0000h
C0h - CCh	SRAM Partition 02 Rule(n) Register (RAM02_RULE0 - RAM02_RULE3)	32	RW	0000_0000h
D0h - DCh	SRAM Partition 03 Rule(n) Register (RAM03_RULE0 - RAM03_RULE3)	32	RW	0000_0000h
F0h - FCh	SRAM Partition 04 Rule(n) Register (RAM04_RULE0 - RAM04_RULE3)	32	RW	0000_0000h
100h - 10Ch	SRAM Partition 05 Rule(n) Register (RAM05_RULE0 - RAM05_RULE3)	32	RW	0000_0000h
110h - 11Ch	SRAM Partition 06 Rule(n) Register (RAM06_RULE0 - RAM06_RULE3)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
120h - 12Ch	SRAM Partition 07 Rule(n) Register (RAM07_RULE0 - RAM07_RULE3)	32	RW	0000_0000h
140h - 14Ch	SRAM Partition 08 Rule(n) Register (RAM08_RULE0 - RAM08_RULE3)	32	RW	0000_0000h
150h - 15Ch	SRAM Partition 09 Rule(n) Register (RAM09_RULE0 - RAM09_RULE3)	32	RW	0000_0000h
160h - 16Ch	SRAM Partition 10 Rule(n) Register (RAM10_RULE0 - RAM10_RULE3)	32	RW	0000_0000h
170h - 17Ch	SRAM Partition 11 Rule(n) Register (RAM11_RULE0 - RAM11_RULE3)	32	RW	0000_0000h
190h - 19Ch	SRAM Partition 12 Rule(n) Register (RAM12_RULE0 - RAM12_RULE3)	32	RW	0000_0000h
1A0h - 1ACh	SRAM Partition 13 Rule(n) Register (RAM13_RULE0 - RAM13_RULE3)	32	RW	0000_0000h
1B0h - 1BCh	SRAM Partition 14 Rule(n) Register (RAM14_RULE0 - RAM14_RULE3)	32	RW	0000_0000h
1C0h - 1CCh	SRAM Partition 15 Rule(n) Register (RAM15_RULE0 - RAM15_RULE3)	32	RW	0000_0000h
1E0h - 1ECh	SRAM Partition 16 Rule(n) Register (RAM16_RULE0 - RAM16_RULE3)	32	RW	0000_0000h
1F0h - 1FCh	SRAM Partition 17 Rule(n) Register (RAM17_RULE0 - RAM17_RULE3)	32	RW	0000_0000h
200h - 20Ch	SRAM Partition 18 Rule(n) Register (RAM18_RULE0 - RAM18_RULE3)	32	RW	0000_0000h
210h - 21Ch	SRAM Partition 19 Rule(n) Register (RAM19_RULE0 - RAM19_RULE3)	32	RW	0000_0000h
230h - 23Ch	SRAM Partition 20 Rule(n) Register (RAM20_RULE0 - RAM20_RULE3)	32	RW	0000_0000h
240h - 24Ch	SRAM Partition 21 Rule(n) Register (RAM21_RULE0 - RAM21_RULE3)	32	RW	0000_0000h
250h - 25Ch	SRAM Partition 22 Rule(n) Register (RAM22_RULE0 - RAM22_RULE3)	32	RW	0000_0000h
260h - 26Ch	SRAM Partition 23 Rule(n) Register (RAM23_RULE0 - RAM23_RULE3)	32	RW	0000_0000h
280h - 28Ch	SRAM Partition 24 Rule(n) Register (RAM24_RULE0 - RAM24_RULE3)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
290h - 29Ch	SRAM Partition 25 Rule(n) Register (RAM25_RULE0 - RAM25_RULE3)	32	RW	0000_0000h
2A0h - 2ACh	SRAM Partition 26 Rule(n) Register (RAM26_RULE0 - RAM26_RULE3)	32	RW	0000_0000h
2B0h - 2BCh	SRAM Partition 27 Rule(n) Register (RAM27_RULE0 - RAM27_RULE3)	32	RW	0000_0000h
2D0h - 2DCh	SRAM Partition 28 Rule(n) Register (RAM28_RULE0 - RAM28_RULE3)	32	RW	0000_0000h
2E0h - 2ECh	SRAM Partition 29 Rule(n) Register (RAM29_RULE0 - RAM29_RULE3)	32	RW	0000_0000h
2F0h - 2FCh	SRAM Partition 30 Rule(n) Register (RAM30_RULE0 - RAM30_RULE3)	32	RW	0000_0000h
300h - 30Ch	SRAM Partition 31 Rule(n) Register (RAM31_RULE0 - RAM31_RULE3)	32	RW	0000_0000h
320h - 32Ch	Smart DMA (SDMA) RAM Rule(n) Register (SDMA_RAM_RULE0 - SDMA_RAM_RULE3)	32	RW	0000_0000h
340h - 34Ch	FlexSPI1 Region 0 Rule(n) Register (FLEXSPI1_REGION0_RULE0 - FLEXSPI1_REGION0_RULE3)	32	RW	0000_0000h
350h	FlexSPI1 Region 1 Rule 0 Register (FLEXSPI1_REGION1_RULE0)	32	RW	0000_0000h
360h	FlexSPI1 Region 2 Rule 0 Register (FLEXSPI1_REGION2_RULE0)	32	RW	0000_0000h
370h	FlexSPI1 Region 3 Rule 0 Register (FLEXSPI1_REGION3_RULE0)	32	RW	0000_0000h
380h	FlexSPI1 Region 4 Rule 0 Register (FLEXSPI1_REGION4_RULE0)	32	RW	0000_0000h
3A0h	APB Bridge Peripheral 0 Rule 0 Register (APB_BRIDGE_PER0_RULE0)	32	RW	0000_0000h
3A4h	APB Bridge Peripheral 0 Rule 1 Register (APB_BRIDGE_PER0_RULE1)	32	RW	0000_0000h
3ACh	APB Bridge Peripheral 0 Rule 3 Register (APB_BRIDGE_PER0_RULE3)	32	RW	0000_0000h
3B0h	APB Bridge Peripheral 1 Rule 0 Register (APB_BRIDGE_PER1_RULE0)	32	RW	0000_0000h
3B4h	APB Bridge Peripheral 1 Rule 1 Register (APB_BRIDGE_PER1_RULE1)	32	RW	0000_0000h
3B8h	APB Bridge Peripheral 1 Rule 2 Register (APB_BRIDGE_PER1_RULE2)	32	RW	0000_0000h
3BCh	APB Bridge Peripheral 1 Rule 3 Register (APB_BRIDGE_PER1_RULE3)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
3C0h	AHB Peripheral 0 Slave Rule 0 Register (AHB_PERIPH0_SLAVE_RULE0)	32	RW	0000_0000h
3D0h	AIPS Bridge Peripheral 0 Rule 0 Register (AIPS_BRIDGE0_PER_RULE0)	32	RW	0000_0000h
3E0h	AHB Peripheral 1 Slave Rule 0 Register (AHB_PERIPH1_SLAVE_RULE0)	32	RW	0000_0000h
3E4h	AHB Peripheral 1 Slave Rule 1 Register (AHB_PERIPH1_SLAVE_RULE1)	32	RW	0000_0000h
400h	AIPS Bridge Peripheral 1 Rule 0 Register (AIPS_BRIDGE1_PER_RULE0)	32	RW	0000_0000h
404h	AIPS Bridge Peripheral 1 Rule 1 Register (AIPS_BRIDGE1_PER_RULE1)	32	RW	0000_0000h
410h	AHB Peripheral 2 Slave Rule 0 Register (AHB_PERIPH2_SLAVE_RULE0)	32	RW	0000_0000h
420h	AHB Secure Control Peripheral Rule 0 Register (AHB_SECURE_CTRL_PERIPH_RULE0)	32	RW	0000_0000h
430h	AHB Peripheral 3 Slave Rule 0 Register (AHB_PERIPH3_SLAVE_RULE0)	32	RW	0000_0000h
434h	AHB Peripheral 3 Slave Rule 1 Register (AHB_PERIPH3_SLAVE_RULE1)	32	RW	0000_0000h
E00h - E44h	Security Violation Address(n) Register (SEC_VIO_ADDR0 - SEC_VIO_ADDR17)	32	R	0000_0000h
E80h - EC4h	Security Violation Miscellaneous Information at Address(n) Register (SEC_VIO_MISC_INFO0 - SEC_VIO_MISC_INFO17)	32	R	0000_0000h
F00h	Security Violation Info Validity for Address(n) Register (SEC_VIO_INFO_VALID)	32	RW	0000_0000h
F80h	GPIO Mask for Port 0 Register (SEC_GPIO_MASK0)	32	RW	FFFF_FFFFh
F84h	GPIO Mask for Port 1 Register (SEC_GPIO_MASK1)	32	RW	FFFF_FFFFh
F88h	GPIO Mask for Port 2 Register (SEC_GPIO_MASK2)	32	RW	FFFF_FFFFh
FA0h	Secure Interrupt Mask for DSP Register (DSP_INT_MASK0)	32	RW	FFFF_FFFFh
FBCh	Secure Mask Lock Register (SEC_MASK_LOCK)	32	RW	0002_AAAAh
FD0h	Master Secure Level Register (MASTER_SEC_LEVEL)	32	RW	8000_0000h
FD4h	Master Secure Level Register (MASTER_SEC_ANTI_POL_REG)	32	RW	BFFF_FFFFh
FECh	Miscellaneous CPU0 Control Signals Register (CM33_LOCK_REG)	32	RW	8000_02AAh
FF8h	Secure Control Duplicate Register (MISC_CTRL_DP_REG)	32	RW	0000_AAAAh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
FFCh	Secure Control Register (MISC_CTRL_REG)	32	RW	0000_AAAAh

22.4.1.2 Memory ROM Rule(n) Register (ROM_MEM_RULE0 - ROM_MEM_RULE3)

Offset

Register	Offset
ROM_MEM_RULE0	10h
ROM_MEM_RULE1	14h
ROM_MEM_RULE2	18h
ROM_MEM_RULE3	1Ch

Function

These 4 registers control access to the ROM. For each register, each rule field controls access to a specific range in ROM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 190. ROM_MEM_RULE0

Rule	Address Access
0	0x0300 0000 – 0x0300 0FFF
1	0x0300 1000 – 0x0300 1FFF
2	0x0300 2000 – 0x0300 2FFF
3	0x0300 3000 – 0x0300 3FFF
4	0x0300 4000 – 0x0300 4FFF
5	0x0300 5000 – 0x0300 5FFF
6	0x0300 6000 – 0x0300 6FFF
7	0x0300 7000 – 0x0300 7FFF

Table 191. ROM_MEM_RULE1

Rule	Address Access
0	0x0300 8000 – 0x0300 8FFF
1	0x0300 9000 – 0x0300 9FFF
2	0x0300 A000 – 0x0300 AFFF

Table continues on the next page...

Table 191. ROM_MEM_RULE1 (continued)

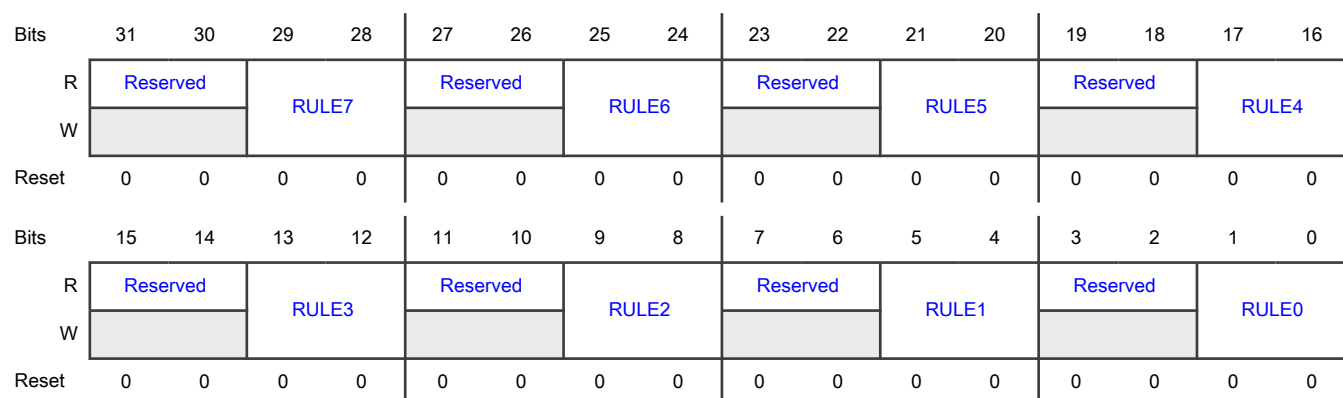
Rule	Address Access
3	0x0300 B000 – 0x0300 BFFF
4	0x0300 C000 – 0x0300 CFFF
5	0x0300 D000 – 0x0300 DFFF
6	0x0300 E000 – 0x0300 EFFF
7	0x0300 F000 – 0x0300 FFFF

Table 192. ROM_MEM_RULE2

Rule	Address Access
0	0x0301 0000 – 0x0301 0FFF
1	0x0301 1000 – 0x0301 1FFF
2	0x0301 2000 – 0x0301 2FFF
3	0x0301 3000 – 0x0301 3FFF
4	0x0301 4000 – 0x0301 4FFF
5	0x0301 5000 – 0x0301 5FFF
6	0x0301 6000 – 0x0301 6FFF
7	0x0301 7000 – 0x0301 7FFF

Table 193. ROM_MEM_RULE3

Rule	Address Access
0	0x0301 8000 – 0x0301 8FFF
1	0x0301 9000 – 0x0301 9FFF
2	0x0301 A000 – 0x0301 AFFF
3	0x0301 B000 – 0x0301 BFFF
4	0x0301 C000 – 0x0301 CFFF
5	0x0301 D000 – 0x0301 DFFF
6	0x0301 E000 – 0x0301 EFFF
7	0x0301 F000 – 0x0301 FFFF

Diagram**Fields**

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the ROM memory at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the ROM memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.3 FLEXSPI0 Region 0 Rule(n) Register (FLEXSPI0_REGION0_RULE0 - FLEXSPI0_REGION0_RULE3)

Offset

Register	Offset
FLEXSPI0_REGION0_RULE0	30h
FLEXSPI0_REGION0_RULE1	34h
FLEXSPI0_REGION0_RULE2	38h
FLEXSPI0_REGION0_RULE3	3Ch

Function

These 4 registers control access to the FLEXSPI0 External Memory interface. For each register, each rule field controls access to a specific range in FLEXSPI0 External Memory interface.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 194. FLEXSPI0_REGION0_RULE0

Rule	Address Access
0	0x0800 0000 – 0x0803 FFFF

Table continues on the next page...

Table 194. FLEXSPI0_REGION0_RULE0 (continued)

Rule	Address Access
1	0x0804 0000 – 0x0807 FFFF
2	0x0808 0000 – 0x080B FFFF
3	0x080C 0000 – 0x080F FFFF
4	0x0810 0000 – 0x0813 FFFF
5	0x0814 0000 – 0x0817 FFFF
6	0x0818 0000 – 0x081B FFFF
7	0x081C 0000 – 0x081F FFFF

Table 195. FLEXSPI0_REGION0_RULE1

Rule	Address Access
0	0x0820 0000 – 0x0823 FFFF
1	0x0824 0000 – 0x0827 FFFF
2	0x0828 0000 – 0x082B FFFF
3	0x082C 0000 – 0x082F FFFF
4	0x0830 0000 – 0x0833 FFFF
5	0x0834 0000 – 0x0837 FFFF
6	0x0838 0000 – 0x083B FFFF
7	0x083C 0000 – 0x083F FFFF

Table 196. FLEXSPI0_REGION0_RULE2

Rule	Address Access
0	0x0840 0000 – 0x0843 FFFF
1	0x0844 0000 – 0x0847 FFFF
2	0x0848 0000 – 0x084B FFFF
3	0x084C 0000 – 0x084F FFFF
4	0x0850 0000 – 0x0853 FFFF
5	0x0854 0000 – 0x0857 FFFF
6	0x0858 0000 – 0x085B FFFF
7	0x085C 0000 – 0x085F FFFF

Table 197. FLEXSPI0_REGION0_RULE3

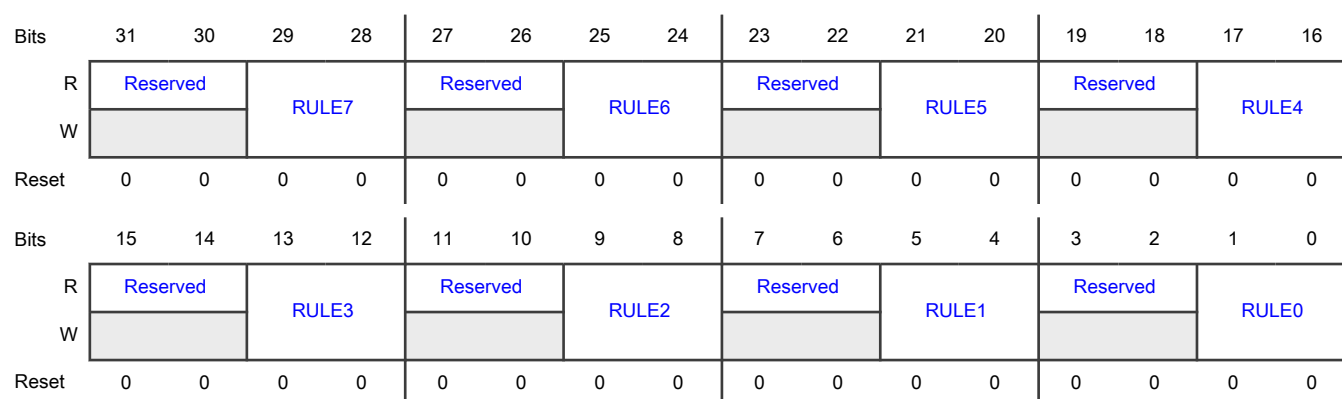
Rule	Address Access
0	0x0860 0000 – 0x0863 FFFF
1	0x0864 0000 – 0x0867 FFFF

Table continues on the next page...

Table 197. FLEXSPI0_REGION0_RULE3 (continued)

Rule	Address Access
2	0x0868 0000 – 0x086B FFFF
3	0x086C 0000 – 0x086 FFFF
4	0x0870 0000 – 0x0873 FFFF
5	0x0874 0000 – 0x0877 FFFF
6	0x0878 0000 – 0x087B FFFF
7	0x087C 0000 – 0x087F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9-8 RULE2	Rule 2 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the FLEXSPI0 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.4 FLEXSPI0 Region a Rule 0 Register (FLEXSPI0_REGION1_RULE0 - FLEXSPI0_REGION4_RULE0)

Offset

Register	Offset
FLEXSPI0_REGION1_RULE0	40h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
FLEXSPI0_REGION2_RULE0	50h
FLEXSPI0_REGION3_RULE0	60h
FLEXSPI0_REGION4_RULE0	70h

Function

These 4 registers control access to the FLEXSPI0 external memory interface. For each register, each rule field controls access to a specific range in FLEXSPI0 external memory interface.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

FLEXSPI0_REGION1_RULE0	0x0880 0000 – 0x08FF FFFF
FLEXSPI0_REGION2_RULE0	0x0900 0000 – 0x09FF FFFF
FLEXSPI0_REGION3_RULE0	0x0A00 0000 – 0x0BFFFFFF
FLEXSPI0_REGION4_RULE0	0x0C00 0000 – 0x0FFF FFFF

Table 198. FLEXSPI0_REGION1_RULE0

Rule	Address Access
0	0x0880 0000 – 0x089F FFFF
1	0x08A0 0000 – 0x08BF FFFF
2	0x08C0 0000 – 0x08DF FFFF
3	0x08E0 0000 – 0x08FF FFFF

Table 199. FLEXSPI0_REGION2_RULE0

Rule	Address Access
0	0x0900 0000 – 0x093F FFFF
1	0x0940 0000 – 0x097F FFFF
2	0x0980 0000 – 0x09BF FFFF
3	0x09C0 0000 – 0x09FF FFFF

Table 200. FLEXSPI0_REGION3_RULE0

Rule	Address Access
0	0x0A00 0000 – 0x0A7F FFFF

Table continues on the next page...

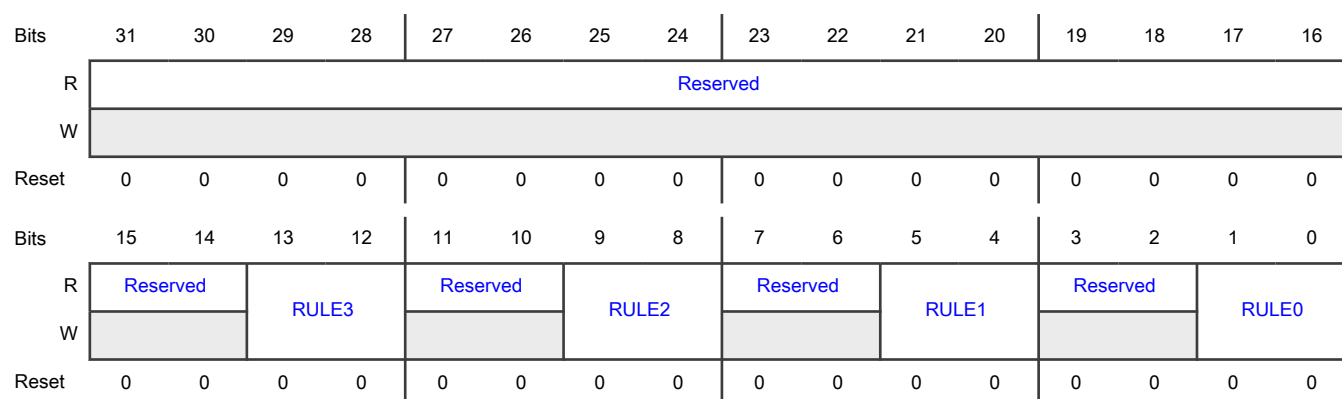
Table 200. FLEXSPI0_REGION3_RULE0 (continued)

Rule	Address Access
1	0x0A80 0000 – 0x0AFF FFFF
2	0x0B00 0000 – 0x0B7F FFFF
3	0x0B80 0000 – 0x0BFF FFFF

Table 201. FLEXSPI0_REGION4_RULE0

Rule	Address Access
0	0x0C00 0000 – 0x0CFF FFFF
1	0x0D00 0000 – 0x0DFF FFFF
2	0x0E00 0000 – 0x0EFF FFFF
3	0x0F00 0000 – 0x0FFF FFFF

Diagram



Fields

Field	Function
31-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the FLEXSPI0 external memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9-8 RULE2	Rule 2 Defines the minimal security level to access the FLEXSPI0 external memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the FLEXSPI0 external memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the FLEXSPI0 external memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.5 SRAM Partition 00 Rule(n) Register (RAM00_RULE0 - RAM00_RULE3)

Offset

Register	Offset
RAM00_RULE0	90h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
RAM00_RULE1	94h
RAM00_RULE2	98h
RAM00_RULE3	9Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 202. RAM00_RULE0

Rule	Address Access
0	0x2000 0000 – 0x2000 03FF
1	0x2000 0400 – 0x2000 07FF
2	0x2000 0800 – 0x2000 0BFF
3	0x2000 0C00 – 0x2000 0FFF
4	0x2000 1000 – 0x2000 13FF
5	0x2000 1400 – 0x2000 17FF
6	0x2000 1800 – 0x2000 1BFF
7	0x2000 1C00 – 0x2000 1FFF

Table 203. RAM00_RULE1

Rule	Address Access
0	0x2000 2000 – 0x2000 23FF
1	0x2000 2400 – 0x2000 27FF
2	0x2000 2800 – 0x2000 2BFF
3	0x2000 2C00 – 0x2000 2FFF
4	0x2000 3000 – 0x2000 33FF
5	0x2000 3400 – 0x2000 37FF
6	0x2000 3800 – 0x2000 3BFF
7	0x2000 3C00 – 0x2000 3FFF

Table 204. RAM00_RULE2

Rule	Address Access
0	0x2000 4000 – 0x2000 43FF

Table continues on the next page...

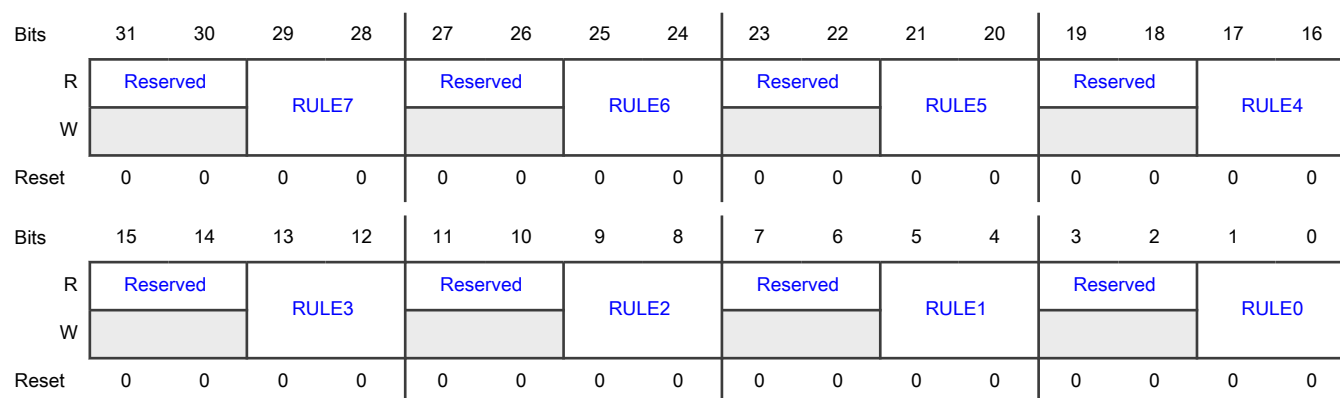
Table 204. RAM00_RULE2 (continued)

Rule	Address Access
1	0x2000 4400 – 0x2000 47FF
2	0x2000 4800 – 0x2000 4BFF
3	0x2000 4C00 – 0x2000 4FFF
4	0x2000 5000 – 0x2000 53FF
5	0x2000 5400 – 0x2000 57FF
6	0x2000 5800 – 0x2000 5BFF
7	0x2000 5C00 – 0x2000 5FFF

Table 205. RAM00_RULE3

Rule	Address Access
0	0x2000 6000 – 0x2000 63FF
1	0x2000 6400 – 0x2000 67FF
2	0x2000 6800 – 0x2000 6BFF
3	0x2000 6C00 – 0x2000 6FFF
4	0x2000 7000 – 0x2000 73FF
5	0x2000 7400 – 0x2000 77FF
6	0x2000 7800 – 0x2000 7BFF
7	0x2000 7C00 – 0x2000 7FFF

Diagram



Fields

Field	Function
31-30	Reserved; the value read from a reserved bit is not defined.
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.6 SRAM Partition 01 Rule(n) Register (RAM01_RULE0 - RAM01_RULE3)

Offset

Register	Offset
RAM01_RULE0	A0h
RAM01_RULE1	A4h
RAM01_RULE2	A8h
RAM01_RULE3	ACh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 206. RAM01_RULE0

Rule	Address Access
0	0x2000 8000 – 0x2000 83FF
1	0x2000 8400 – 0x2000 87FF
2	0x2000 8800 – 0x2000 8BFF
3	0x2000 8C00 – 0x2000 8FFF
4	0x2000 9000 – 0x2000 93FF
5	0x2000 9400 – 0x2000 97FF
6	0x2000 9800 – 0x2000 9BFF
7	0x2000 9C00 – 0x2000 9FFF

Table 207. RAM01_RULE1

Rule	Address Access
0	0x2000 A000 – 0x2000 A3FF
1	0x2000 A400 – 0x2000 A7FF
2	0x2000 A800 – 0x2000 ABFF
3	0x2000 AC00 – 0x2000 AFFF
4	0x2000 B000 – 0x2000 B3FF
5	0x2000 B400 – 0x2000 B7FF
6	0x2000 B800 – 0x2000 BBFF
7	0x2000 BC00 – 0x2000 BFFF

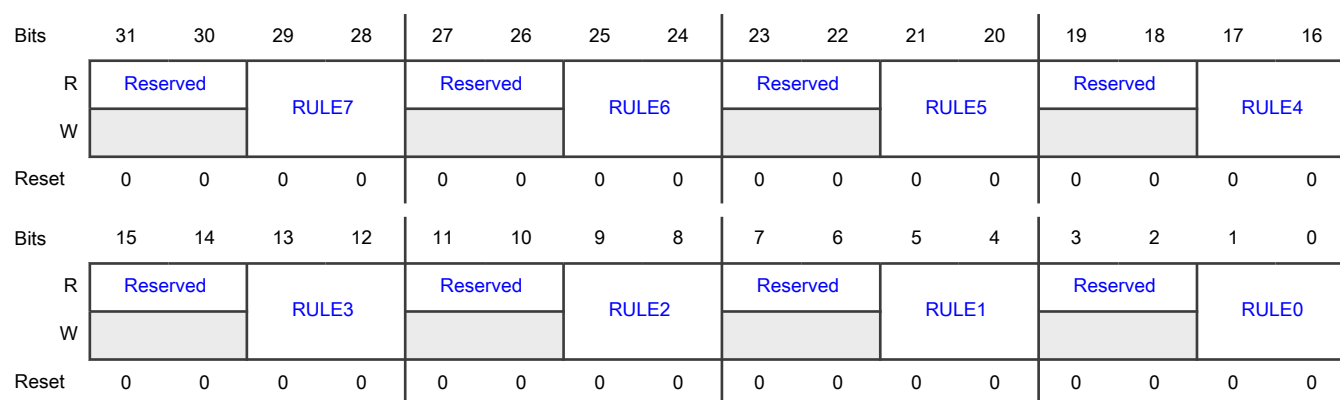
Table 208. RAM01_RULE2

Rule	Address Access
0	0x2000 C000 – 0x2000 C3FF
1	0x2000 C400 – 0x2000 C7FF
2	0x2000 C800 – 0x2000 CBFF
3	0x2000 CC00 – 0x2000 CFFF
4	0x2000 D000 – 0x2000 D3FF
5	0x2000 D400 – 0x2000 D7FF
6	0x2000 D800 – 0x2000 DBFF
7	0x2000 DC00 – 0x2000 DFFF

Table 209. RAM01_RULE3

Rule	Address Access
0	0x2000 E000 – 0x2000 E3FF
1	0x2000 E400 – 0x2000 E7FF
2	0x2000 E800 – 0x2000 EBFF
3	0x2000 EC00 – 0x2000 EFFF
4	0x2000 F000 – 0x2000 F3FF
5	0x2000 F400 – 0x2000 F7FF
6	0x2000 F800 – 0x2000 FBFF
7	0x2000 FC00 – 0x2000 FFFF

Diagram



Fields

Field	Function
31-30	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Secure and privilege user access allowed

22.4.1.7 SRAM Partition 02 Rule(n) Register (RAM02_RULE0 - RAM02_RULE3)

Offset

Register	Offset
RAM02_RULE0	C0h
RAM02_RULE1	C4h
RAM02_RULE2	C8h
RAM02_RULE3	CCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 210. RAM02_RULE0

Rule	Address Access
0	0x2001 0000 – 0x2001 03FF
1	0x2001 0400 – 0x2001 07FF
2	0x2001 0800 – 0x2001 0BFF
3	0x2001 0C00 – 0x2001 0FFF
4	0x2001 1000 – 0x2001 13FF
5	0x2001 1400 – 0x2001 17FF
6	0x2001 1800 – 0x2001 1BFF
7	0x2001 1C00 – 0x2001 1FFF

Table 211. RAM02_RULE1

Rule	Address Access
0	0x2001 2000 – 0x2001 23FF
1	0x2001 2400 – 0x2001 27FF
2	0x2001 2800 – 0x2001 2BFF
3	0x2001 2C00 – 0x2001 2FFF

Table continues on the next page...

Table 211. RAM02_RULE1 (continued)

Rule	Address Access
4	0x2001 3000 – 0x2001 33FF
5	0x2001 3400 – 0x2001 37FF
6	0x2001 3800 – 0x2001 3BFF
7	0x2001 3C00 – 0x2001 3FFF

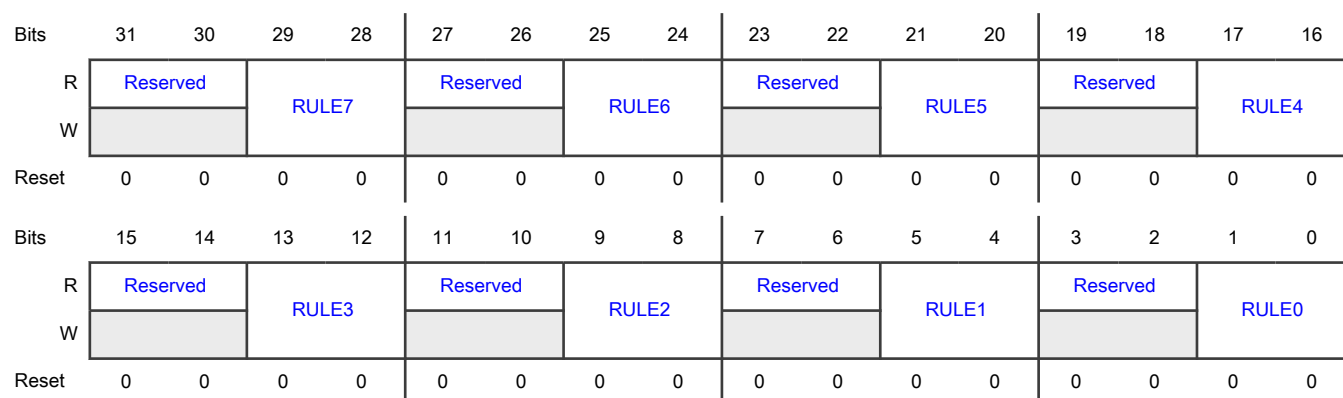
Table 212. RAM02_RULE2

Rule	Address Access
0	0x2001 4000 – 0x2001 43FF
1	0x2001 4400 – 0x2001 47FF
2	0x2001 4800 – 0x2001 4BFF
3	0x2001 4C00 – 0x2001 4FFF
4	0x2001 5000 – 0x2001 53FF
5	0x2001 5400 – 0x2001 57FF
6	0x2001 5800 – 0x2001 5BFF
7	0x2001 5C00 – 0x2001 5FFF

Table 213. RAM02_RULE3

Rule	Address Access
0	0x2001 6000 – 0x2001 63FF
1	0x2001 6400 – 0x2001 67FF
2	0x2001 6800 – 0x2001 6BFF
3	0x2001 6C00 – 0x2001 6FFF
4	0x2001 7000 – 0x2001 73FF
5	0x2001 7400 – 0x2001 77FF
6	0x2001 7800 – 0x2001 7BFF
7	0x2001 7C00 – 0x2001 7FFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.8 SRAM Partition 03 Rule(n) Register (RAM03_RULE0 - RAM03_RULE3)

Offset

Register	Offset
RAM03_RULE0	D0h
RAM03_RULE1	D4h
RAM03_RULE2	D8h
RAM03_RULE3	DCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 214. RAM03_RULE0

Rule	Address Access
0	0x2001 8000 – 0x2001 83FF
1	0x2001 8400 – 0x2001 87FF
2	0x2001 8800 – 0x2001 8BFF
3	0x2001 8C00 – 0x2001 8FFF

Table continues on the next page...

Table 214. RAM03_RULE0 (continued)

Rule	Address Access
4	0x2001 9000 – 0x2001 93FF
5	0x2001 9400 – 0x2001 97FF
6	0x2001 9800 – 0x2001 9BFF
7	0x2001 9C00 – 0x2001 9FFF

Table 215. RAM03_RULE1

Rule	Address Access
0	0x2001 A000 – 0x2001 A3FF
1	0x2001 A400 – 0x2001 A7FF
2	0x2001 A800 – 0x2001 ABFF
3	0x2001 AC00 – 0x2001 AFFF
4	0x2001 B000 – 0x2001 B3FF
5	0x2001 B400 – 0x2001 B7FF
6	0x2001 B800 – 0x2001 BBFF
7	0x2001 BC00 – 0x2001 BFFF

Table 216. RAM03_RULE2

Rule	Address Access
0	0x2001 C000 – 0x2001 C3FF
1	0x2001 C400 – 0x2001 C7FF
2	0x2001 C800 – 0x2001 CBFF
3	0x2001 CC00 – 0x2001 CFFF
4	0x2001 D000 – 0x2001 D3FF
5	0x2001 D400 – 0x2001 D7FF
6	0x2001 D800 – 0x2001 DBFF
7	0x2001 DC00 – 0x2001 DFFF

Table 217. RAM03_RULE3

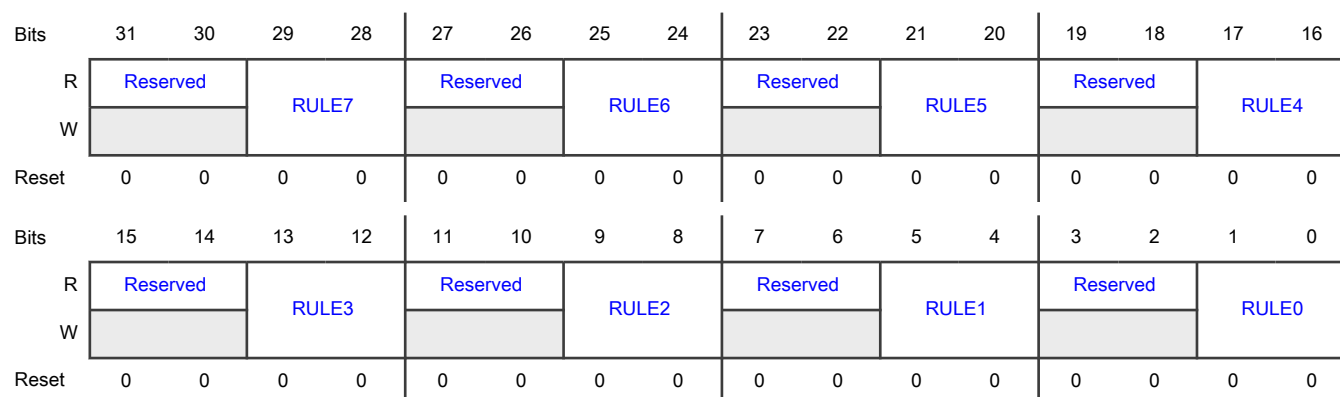
Rule	Address Access
0	0x2001 E000 – 0x2001 E3FF
1	0x2001 E400 – 0x2001 E7FF
2	0x2001 E800 – 0x2001 EBFF
3	0x2001 EC00 – 0x2001 EFFF
4	0x2001 F000 – 0x2001 F3FF

Table continues on the next page...

Table 217. RAM03_RULE3 (continued)

Rule	Address Access
5	0x2001 F400 – 0x2001 F7FF
6	0x2001 F800 – 0x2001 FBFF
7	0x2001 FC00 – 0x2001 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.9 SRAM Partition 04 Rule(n) Register (RAM04_RULE0 - RAM04_RULE3)

Offset

Register	Offset
RAM04_RULE0	F0h
RAM04_RULE1	F4h
RAM04_RULE2	F8h
RAM04_RULE3	FCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 218. RAM04_RULE0

Rule	Address Access
0	0x2002 0000 – 0x2002 03FF
1	0x2002 0400 – 0x2002 07FF
2	0x2002 0800 – 0x2002 0BFF
3	0x2002 0C00 – 0x2002 0FFF
4	0x2002 1000 – 0x2002 13FF
5	0x2002 1400 – 0x2002 17FF
6	0x2002 1800 – 0x2002 1BFF
7	0x2002 1C00 – 0x2002 1FFF

Table 219. RAM04_RULE1

Rule	Address Access
0	0x2002 2000 – 0x2002 23FF
1	0x2002 2400 – 0x2002 27FF
2	0x2002 2800 – 0x2002 2BFF
3	0x2002 2C00 – 0x2002 2FFF
4	0x2002 3000 – 0x2002 33FF
5	0x2002 3400 – 0x2002 37FF
6	0x2002 3800 – 0x2002 3BFF
7	0x2002 3C00 – 0x2002 3FFF

Table 220. RAM04_RULE2

Rule	Address Access
0	0x2002 4000 – 0x2002 43FF
1	0x2002 4400 – 0x2002 47FF
2	0x2002 4800 – 0x2002 4BFF
3	0x2002 4C00 – 0x2002 4FFF
4	0x2002 5000 – 0x2002 53FF
5	0x2002 5400 – 0x2002 57FF
6	0x2002 5800 – 0x2002 5BFF
7	0x2002 5C00 – 0x2002 5FFF

Table 221. RAM04_RULE3

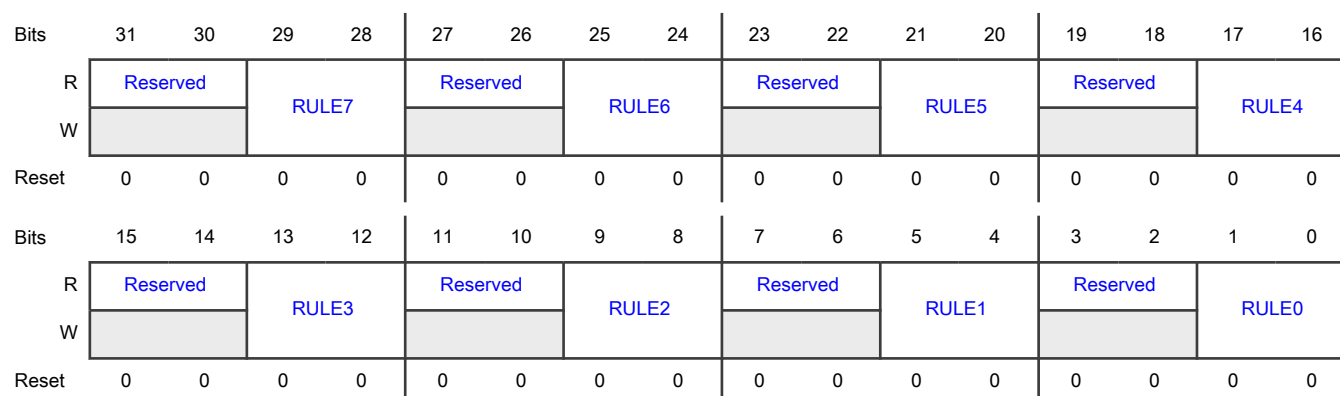
Rule	Address Access
0	0x2002 6000 – 0x2002 63FF

Table continues on the next page...

Table 221. RAM04_RULE3 (continued)

Rule	Address Access
1	0x2002 6400 – 0x2002 67FF
2	0x2002 6800 – 0x2002 6BFF
3	0x2002 6C00 – 0x2002 6FFF
4	0x2002 7000 – 0x2002 73FF
5	0x2002 7400 – 0x2002 77FF
6	0x2002 7800 – 0x2002 7BFF
7	0x2002 7C00 – 0x2002 7FFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the RAM at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8	Rule 2

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE2	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.10 SRAM Partition 05 Rule(n) Register (RAM05_RULE0 - RAM05_RULE3)

Offset

Register	Offset
RAM05_RULE0	100h
RAM05_RULE1	104h
RAM05_RULE2	108h
RAM05_RULE3	10Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 222. RAM05_RULE0

Rule	Address Access
0	0x2002 8000 – 0x2002 83FF
1	0x2002 8400 – 0x2002 87FF
2	0x2002 8800 – 0x2002 8BFF
3	0x2002 8C00 – 0x2002 8FFF
4	0x2002 9000 – 0x2002 93FF
5	0x2002 9400 – 0x2002 97FF
6	0x2002 9800 – 0x2002 9BFF
7	0x2002 9C00 – 0x2002 9FFF

Table 223. RAM05_RULE1

Rule	Address Access
0	0x2002 A000 – 0x2002 A3FF
1	0x2002 A400 – 0x2002 A7FF
2	0x2002 A800 – 0x2002 ABFF
3	0x2002 AC00 – 0x2002 AFFF
4	0x2002 B000 – 0x2002 B3FF
5	0x2002 B400 – 0x2002 B7FF
6	0x2002 B800 – 0x2002 BBFF
7	0x2002 BC00 – 0x2002 BFFF

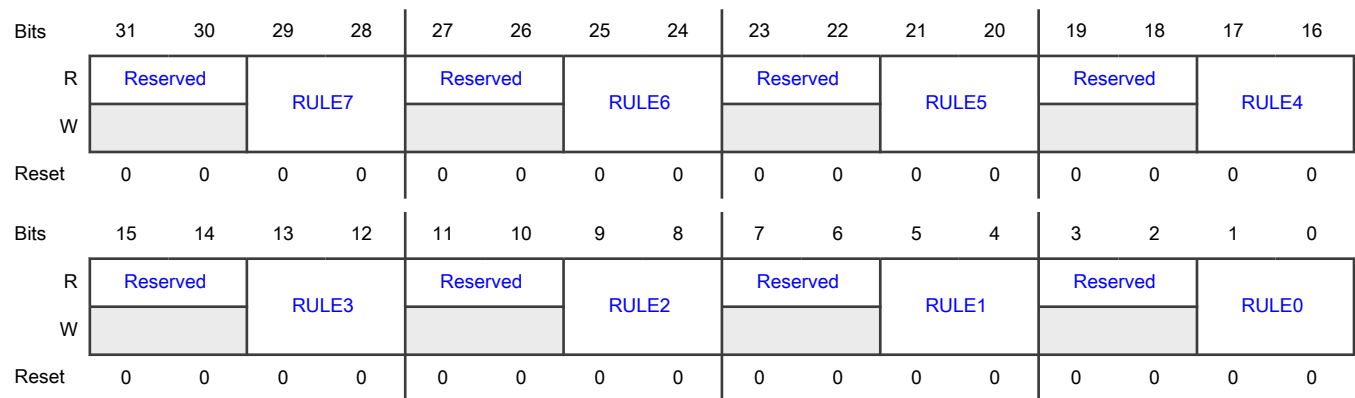
Table 224. RAM05_RULE2

Rule	Address Access
0	0x2002 C000 – 0x2002 C3FF
1	0x2002 C400 – 0x2002 C7FF
2	0x2002 C800 – 0x2002 CBFF
3	0x2002 CC00 – 0x2002 CFFF
4	0x2002 D000 – 0x2002 D3FF
5	0x2002 D400 – 0x2002 D7FF
6	0x2002 D800 – 0x2002 DBFF
7	0x2002 DC00 – 0x2002 DFFF

Table 225. RAM05_RULE3

Rule	Address Access
0	0x2002 E000 – 0x2002 E3FF
1	0x2002 E400 – 0x2002 E7FF
2	0x2002 E800 – 0x2002 EBFF
3	0x2002 EC00 – 0x2002 EFFF
4	0x2002 F000 – 0x2002 F3FF
5	0x2002 F400 – 0x2002 F7FF
6	0x2002 F800 – 0x2002 FBFF
7	0x2002 FC00 – 0x2002 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.11 SRAM Partition 06 Rule(n) Register (RAM06_RULE0 - RAM06_RULE3)

Offset

Register	Offset
RAM06_RULE0	110h
RAM06_RULE1	114h
RAM06_RULE2	118h
RAM06_RULE3	11Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 226. RAM06_RULE0

Rule	Address Access
0	0x2003 0000 – 0x2003 03FF
1	0x2003 0400 – 0x2003 07FF
2	0x2003 0800 – 0x2003 0BFF
3	0x2003 0C00 – 0x2003 0FFF
4	0x2003 1000 – 0x2003 13FF
5	0x2003 1400 – 0x2003 17FF
6	0x2003 1800 – 0x2003 1BFF
7	0x2003 1C00 – 0x2003 1FFF

Table 227. RAM06_RULE1

Rule	Address Access
0	0x2003 2000 – 0x2003 23FF
1	0x2003 2400 – 0x2003 27FF
2	0x2003 2800 – 0x2003 2BFF
3	0x2003 2C00 – 0x2003 2FFF
4	0x2003 3000 – 0x2003 33FF
5	0x2003 3400 – 0x2003 37FF
6	0x2003 3800 – 0x2003 3BFF
7	0x2003 3C00 – 0x2003 3FFF

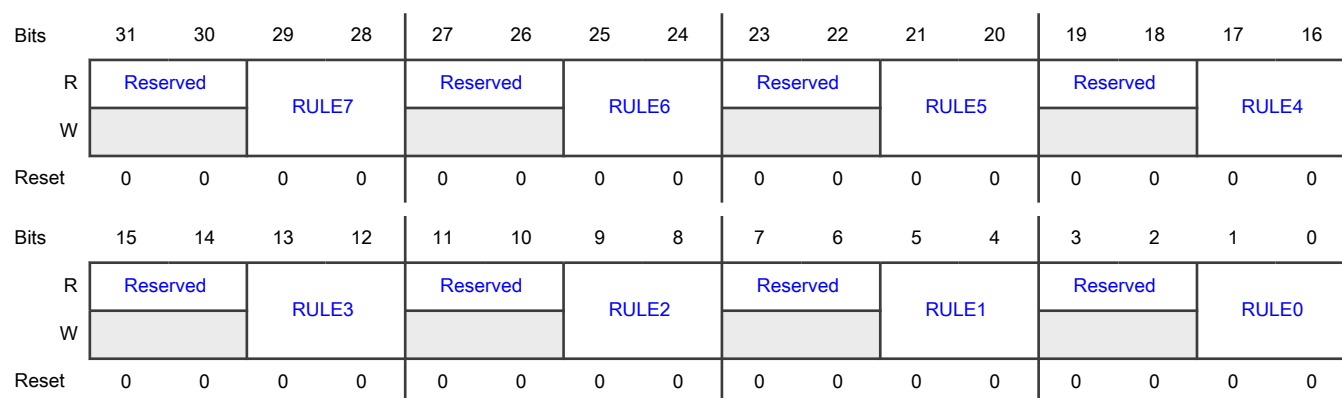
Table 228. RAM06_RULE2

Rule	Address Access
0	0x2003 4000 – 0x2003 43FF
1	0x2003 4400 – 0x2003 47FF
2	0x2003 4800 – 0x2003 4BFF
3	0x2003 4C00 – 0x2003 4FFF
4	0x2003 5000 – 0x2003 53FF
5	0x2003 5400 – 0x2003 57FF
6	0x2003 5800 – 0x2003 5BFF
7	0x2003 5C00 – 0x2003 5FFF

Table 229. RAM06_RULE3

Rule	Address Access
0	0x2003 6000 – 0x2003 63FF
1	0x2003 6400 – 0x2003 67FF
2	0x2003 6800 – 0x2003 6BFF
3	0x2003 6C00 – 0x2003 6FFF
4	0x2003 7000 – 0x2003 73FF
5	0x2003 7400 – 0x2003 77FF
6	0x2003 7800 – 0x2003 7BFF
7	0x2003 7C00 – 0x2003 7FFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.12 SRAM Partition 07 Rule(n) Register (RAM07_RULE0 - RAM07_RULE3)

Offset

Register	Offset
RAM07_RULE0	120h
RAM07_RULE1	124h
RAM07_RULE2	128h
RAM07_RULE3	12Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 230. RAM07_RULE0

Rule	Address Access
0	0x2003 8000 – 0x2003 83FF
1	0x2003 8400 – 0x2003 87FF
2	0x2003 8800 – 0x2003 8BFF
3	0x2003 8C00 – 0x2003 8FFF
4	0x2003 9000 – 0x2003 93FF
5	0x2003 9400 – 0x2003 97FF
6	0x2003 9800 – 0x2003 9BFF
7	0x2003 9C00 – 0x2003 9FFF

Table 231. RAM07_RULE1

Rule	Address Access
0	0x2003 A000 – 0x2003 A3FF
1	0x2003 A400 – 0x2003 A7FF
2	0x2003 A800 – 0x2003 ABFF
3	0x2003 AC00 – 0x2003 AFFF
4	0x2003 B000 – 0x2003 B3FF
5	0x2003 B400 – 0x2003 B7FF
6	0x2003 B800 – 0x2003 BBFF
7	0x2003 BC00 – 0x2003 BFFF

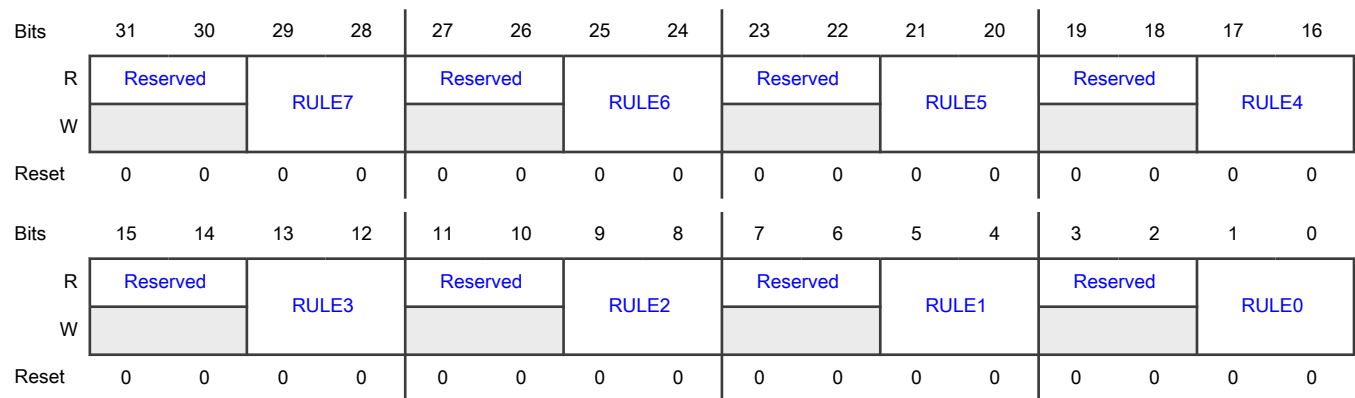
Table 232. RAM07_RULE2

Rule	Address Access
0	0x2003 C000 – 0x2003 C3FF
1	0x2003 C400 – 0x2003 C7FF
2	0x2003 C800 – 0x2003 CBFF
3	0x2003 CC00 – 0x2003 CFFF
4	0x2003 D000 – 0x2003 D3FF
5	0x2003 D400 – 0x2003 D7FF
6	0x2003 D800 – 0x2003 DBFF
7	0x2003 DC00 – 0x2003 DFFF

Table 233. RAM07_RULE3

Rule	Address Access
0	0x2003 E000 – 0x2003 E3FF
1	0x2003 E400 – 0x2003 E7FF
2	0x2003 E800 – 0x2003 EBFF
3	0x2003 EC00 – 0x2003 EFFF
4	0x2003 F000 – 0x2003 F3FF
5	0x2003 F400 – 0x2003 F7FF
6	0x2003 F800 – 0x2003 FBFF
7	0x2003 FC00 – 0x2003 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.13 SRAM Partition 08 Rule(n) Register (RAM08_RULE0 - RAM08_RULE3)

Offset

Register	Offset
RAM08_RULE0	140h
RAM08_RULE1	144h
RAM08_RULE2	148h
RAM08_RULE3	14Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 234. RAM08_RULE0

Rule	Address Access
0	0x2004 0000 – 0x2004 07FF
1	0x2004 0800 – 0x2004 0FFF
2	0x2004 1000 – 0x2004 17FF
3	0x2004 1800 – 0x2004 1FFF
4	0x2004 2000 – 0x2004 27FF
5	0x2004 2800 – 0x2004 2FFF
6	0x2004 3000 – 0x2004 37FF
7	0x2004 3800 – 0x2004 3FFF

Table 235. RAM08_RULE1

Rule	Address Access
0	0x2004 4000 – 0x2004 47FF
1	0x2004 4800 – 0x2004 4FFF
2	0x2004 5000 – 0x2004 57FF
3	0x2004 5800 – 0x2004 5FFF
4	0x2004 6000 – 0x2004 67FF
5	0x2004 6800 – 0x2004 6FFF
6	0x2004 7000 – 0x2004 77FF
7	0x2004 7800 – 0x2004 7FFF

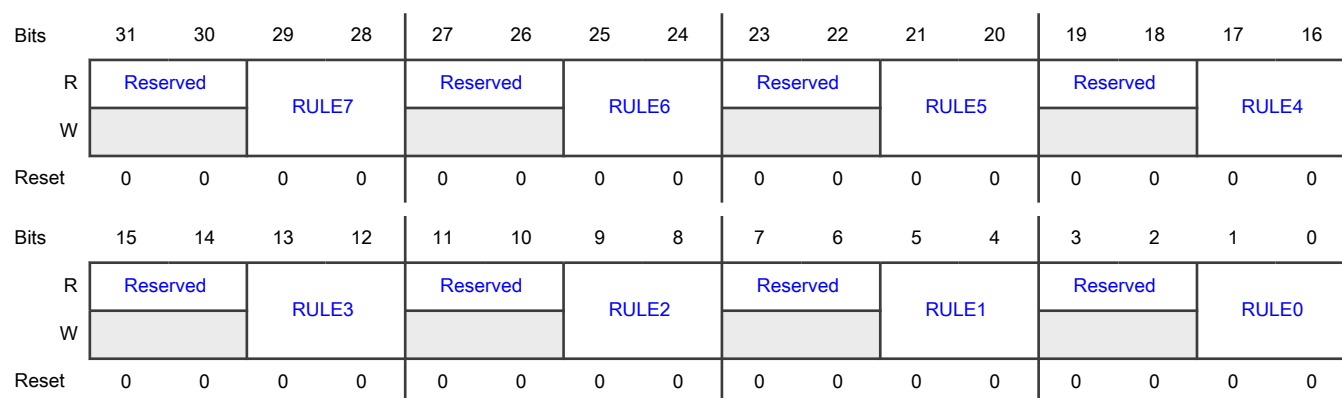
Table 236. RAM08_RULE2

Rule	Address Access
0	0x2004 8000 – 0x2004 87FF
1	0x2004 8800 – 0x2004 8FFF
2	0x2004 9000 – 0x2004 97FF
3	0x2004 9800 – 0x2004 9FFF
4	0x2004 A000 – 0x2004 A7FF
5	0x2004 A800 – 0x2004 AFFF
6	0x2004 B000 – 0x2004 B7FF
7	0x2004 B800 – 0x2004 BFFF

Table 237. RAM08_RULE3

Rule	Address Access
0	0x2004 C000 – 0x2004 C7FF
1	0x2004 C800 – 0x2004 CFFF
2	0x2004 D000 – 0x2004 D7FF
3	0x2004 D800 – 0x2004 DFFF
4	0x2004 E000 – 0x2004 E7FF
5	0x2004 E800 – 0x2004 EFFF
6	0x2004 F000 – 0x2004 F7FF
7	0x2004 F800 – 0x2004 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.14 SRAM Partition 09 Rule(n) Register (RAM09_RULE0 - RAM09_RULE3)

Offset

Register	Offset
RAM09_RULE0	150h
RAM09_RULE1	154h
RAM09_RULE2	158h
RAM09_RULE3	15Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 238. RAM09_RULE0

Rule	Address Access
0	0x2005 0000 – 0x2005 07FF
1	0x2005 0800 – 0x2005 0FFF
2	0x2005 1000 – 0x2005 17FF
3	0x2005 1800 – 0x2005 1FFF
4	0x2005 2000 – 0x2005 27FF
5	0x2005 2800 – 0x2005 2FFF
6	0x2005 3000 – 0x2005 37FF
7	0x2005 3800 – 0x2005 3FFF

Table 239. RAM09_RULE1

Rule	Address Access
0	0x2005 4000 – 0x2005 47FF
1	0x2005 4800 – 0x2005 4FFF
2	0x2005 5000 – 0x2005 57FF
3	0x2005 5800 – 0x2005 5FFF
4	0x2005 6000 – 0x2005 67FF
5	0x2005 6800 – 0x2005 6FFF
6	0x2005 7000 – 0x2005 77FF
7	0x2005 7800 – 0x2005 7FFF

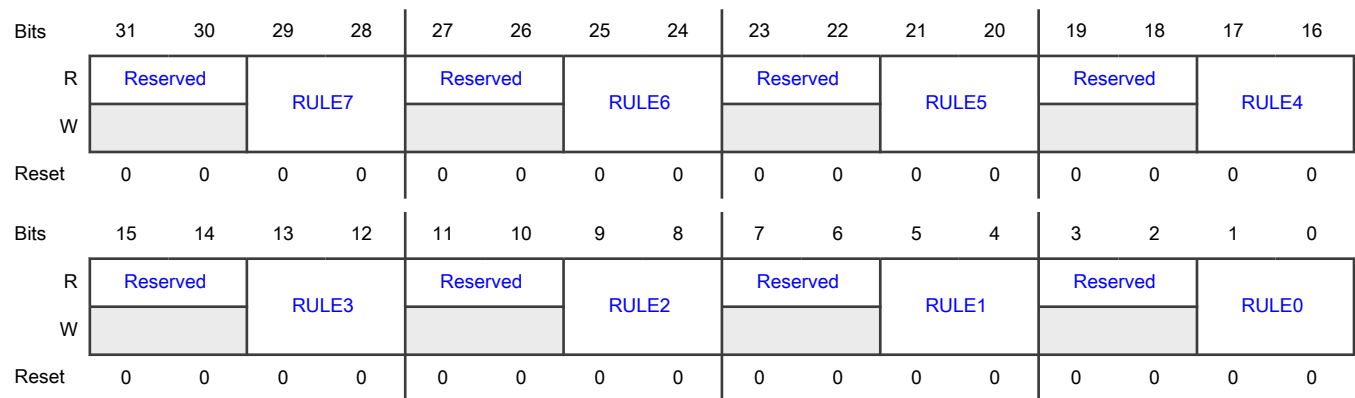
Table 240. RAM09_RULE2

Rule	Address Access
0	0x2005 8000 – 0x2005 87FF
1	0x2005 8800 – 0x2005 8FFF
2	0x2005 9000 – 0x2005 97FF
3	0x2005 9800 – 0x2005 9FFF
4	0x2005 A000 – 0x2005 A7FF
5	0x2005 A800 – 0x2005 AFFF
6	0x2005 B000 – 0x2005 B7FF
7	0x2005 B800 – 0x2005 BFFF

Table 241. RAM09_RULE3

Rule	Address Access
0	0x2005 C000 – 0x2005 C7FF
1	0x2005 C800 – 0x2005 CFFF
2	0x2005 D000 – 0x2005 D7FF
3	0x2005 D800 – 0x2005 DFFF
4	0x2005 E000 – 0x2005 E7FF
5	0x2005 E800 – 0x2005 EFFF
6	0x2005 F000 – 0x2005 F7FF
7	0x2005 F800 – 0x2005 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.15 SRAM Partition 10 Rule(n) Register (RAM10_RULE0 - RAM10_RULE3)

Offset

Register	Offset
RAM10_RULE0	160h
RAM10_RULE1	164h
RAM10_RULE2	168h
RAM10_RULE3	16Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 242. RAM10_RULE0

Rule	Address Access
0	0x2006 0000 – 0x2006 07FF
1	0x2006 0800 – 0x2006 0FFF
2	0x2006 1000 – 0x2006 17FF
3	0x2006 1800 – 0x2006 1FFF
4	0x2006 2000 – 0x2006 27FF
5	0x2006 2800 – 0x2006 2FFF
6	0x2006 3000 – 0x2006 37FF
7	0x2006 3800 – 0x2006 3FFF

Table 243. RAM10_RULE1

Rule	Address Access
0	0x2006 4000 – 0x2006 47FF
1	0x2006 4800 – 0x2006 4FFF
2	0x2006 5000 – 0x2006 57FF
3	0x2006 5800 – 0x2006 5FFF
4	0x2006 6000 – 0x2006 67FF
5	0x2006 6800 – 0x2006 6FFF
6	0x2006 7000 – 0x2006 77FF
7	0x2006 7800 – 0x2006 7FFF

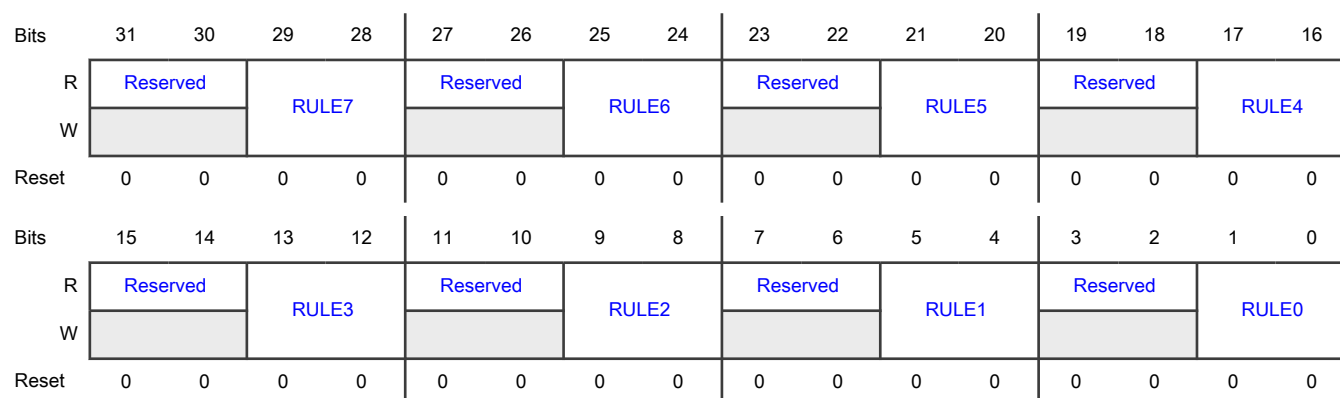
Table 244. RAM10_RULE2

Rule	Address Access
0	0x2006 8000 – 0x2006 87FF
1	0x2006 8800 – 0x2006 8FFF
2	0x2006 9000 – 0x2006 97FF
3	0x2006 9800 – 0x2006 9FFF
4	0x2006 A000 – 0x2006 A7FF
5	0x2006 A800 – 0x2006 AFFF
6	0x2006 B000 – 0x2006 B7FF
7	0x2006 B800 – 0x2006 BFFF

Table 245. RAM10_RULE3

Rule	Address Access
0	0x2006 C000 – 0x2006 C7FF
1	0x2006 C800 – 0x2006 CFFF
2	0x2006 D000 – 0x2006 D7FF
3	0x2006 D800 – 0x2006 DFFF
4	0x2006 E000 – 0x2006 E7FF
5	0x2006 E800 – 0x2006 EFFF
6	0x2006 F000 – 0x2006 F7FF
7	0x2006 F800 – 0x2006 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.16 SRAM Partition 11 Rule(n) Register (RAM11_RULE0 - RAM11_RULE3)

Offset

Register	Offset
RAM11_RULE0	170h
RAM11_RULE1	174h
RAM11_RULE2	178h
RAM11_RULE3	17Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 246. RAM11_RULE0

Rule	Address Access
0	0x2007 0000 – 0x2007 07FF
1	0x2007 0800 – 0x2007 0FFF
2	0x2007 1000 – 0x2007 17FF
3	0x2007 1800 – 0x2007 1FFF
4	0x2007 2000 – 0x2007 27FF
5	0x2007 2800 – 0x2007 2FFF
6	0x2007 3000 – 0x2007 37FF
7	0x2007 3800 – 0x2007 3FFF

Table 247. RAM11_RULE1

Rule	Address Access
0	0x2007 4000 – 0x2007 47FF
1	0x2007 4800 – 0x2007 4FFF
2	0x2007 5000 – 0x2007 57FF
3	0x2007 5800 – 0x2007 5FFF
4	0x2007 6000 – 0x2007 67FF
5	0x2007 6800 – 0x2007 6FFF
6	0x2007 7000 – 0x2007 77FF
7	0x2007 7800 – 0x2007 7FFF

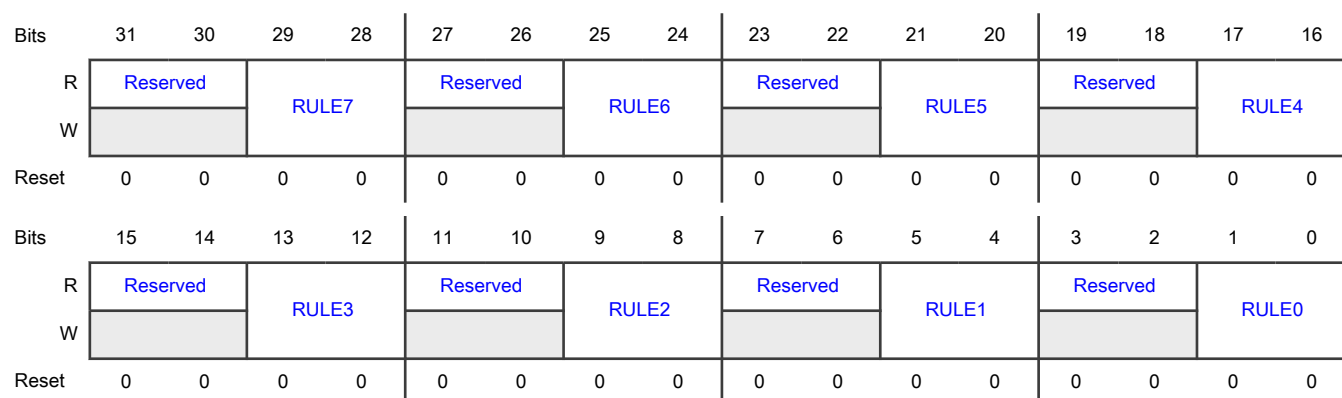
Table 248. RAM11_RULE2

Rule	Address Access
0	0x2007 8000 – 0x2007 87FF
1	0x2007 8800 – 0x2007 8FFF
2	0x2007 9000 – 0x2007 97FF
3	0x2007 9800 – 0x2007 9FFF
4	0x2007 A000 – 0x2007 A7FF
5	0x2007 A800 – 0x2007 AFFF
6	0x2007 B000 – 0x2007 B7FF
7	0x2007 B800 – 0x2007 BFFF

Table 249. RAM11_RULE3

Rule	Address Access
0	0x2007 C000 – 0x2007 C7FF
1	0x2007 C800 – 0x2007 CFFF
2	0x2007 D000 – 0x2007 D7FF
3	0x2007 D800 – 0x2007 DFFF
4	0x2007 E000 – 0x2007 E7FF
5	0x2007 E800 – 0x2007 EFFF
6	0x2007 F000 – 0x2007 F7FF
7	0x2007 F800 – 0x2007 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.17 SRAM Partition 12 Rule(n) Register (RAM12_RULE0 - RAM12_RULE3)

Offset

Register	Offset
RAM12_RULE0	190h
RAM12_RULE1	194h
RAM12_RULE2	198h
RAM12_RULE3	19Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 250. RAM12_RULE0

Rule	Address Access
0	0x2008 0000 – 0x2008 0FFF
1	0x2008 1000 – 0x2008 1FFF
2	0x2008 2000 – 0x2008 2FFF
3	0x2008 3000 – 0x2008 3FFF
4	0x2008 4000 – 0x2008 4FFF
5	0x2008 5000 – 0x2008 5FFF
6	0x2008 6000 – 0x2008 6FFF
7	0x2008 7000 – 0x2008 7FFF

Table 251. RAM12_RULE1

Rule	Address Access
0	0x2008 8000 – 0x2008 8FFF
1	0x2008 9000 – 0x2008 9FFF
2	0x2008 A000 – 0x2008 AFFF
3	0x2008 B000 – 0x2008 BFFF
4	0x2008 C000 – 0x2008 CFFF
5	0x2008 D000 – 0x2008 DFFF
6	0x2008 E000 – 0x2008 EFFF
7	0x2008 F000 – 0x2008 FFFF

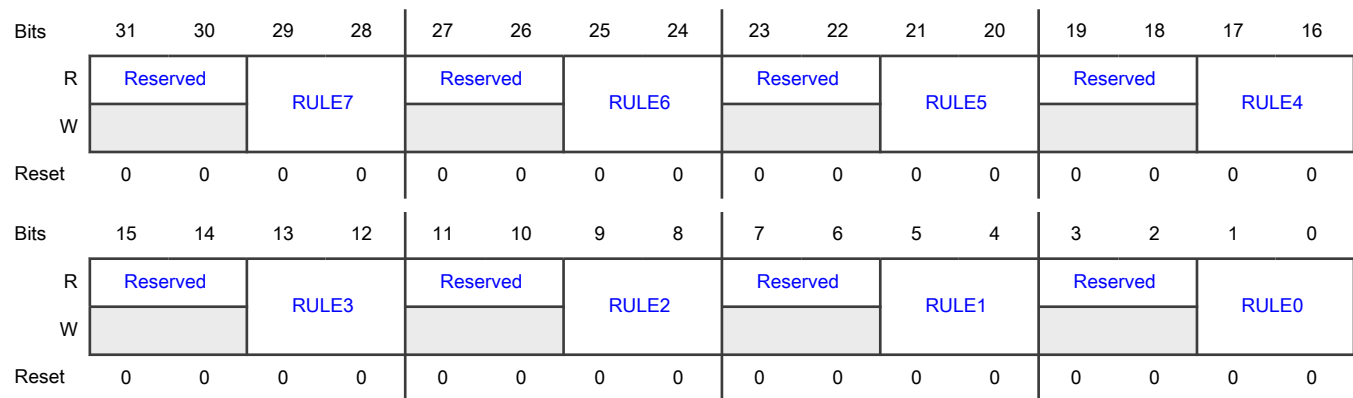
Table 252. RAM12_RULE2

Rule	Address Access
0	0x2009 0000 – 0x2009 0FFF
1	0x2009 1000 – 0x2009 1FFF
2	0x2009 2000 – 0x2009 2FFF
3	0x2009 3000 – 0x2009 3FFF
4	0x2009 4000 – 0x2009 4FFF
5	0x2009 5000 – 0x2009 5FFF
6	0x2009 6000 – 0x2009 6FFF
7	0x2009 7000 – 0x2009 7FFF

Table 253. RAM12_RULE3

Rule	Address Access
0	0x2009 8000 – 0x2009 8FFF
1	0x2009 9000 – 0x2009 9FFF
2	0x2009 A000 – 0x2009 AFFF
3	0x2009 B000 – 0x2009 BFFF
4	0x2009 C000 – 0x2009 CFFF
5	0x2009 D000 – 0x2009 DFFF
6	0x2009 E000 – 0x2009 EFFF
7	0x2009 F000 – 0x2009 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.18 SRAM Partition 13 Rule(n) Register (RAM13_RULE0 - RAM13_RULE3)

Offset

Register	Offset
RAM13_RULE0	1A0h
RAM13_RULE1	1A4h
RAM13_RULE2	1A8h
RAM13_RULE3	1ACh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 254. RAM13_RULE0

Rule	Address Access
0	0x200A 0000 – 0x200A 0FFF
1	0x200A 1000 – 0x200A 1FFF
2	0x200A 2000 – 0x200A 2FFF
3	0x200A 3000 – 0x200A 3FFF
4	0x200A 4000 – 0x200A 4FFF
5	0x200A 5000 – 0x200A 5FFF
6	0x200A 6000 – 0x200A 6FFF
7	0x200A 7000 – 0x200A 7FFF

Table 255. RAM13_RULE1

Rule	Address Access
0	0x200A 8000 – 0x200A 8FFF
1	0x200A 9000 – 0x200A 9FFF
2	0x200A A000 – 0x200A AFFF
3	0x200A B000 – 0x200A BFFF
4	0x200A C000 – 0x200A CFFF
5	0x200A D000 – 0x200A DFFF
6	0x200A E000 – 0x200A EFFF
7	0x200A F000 – 0x200A FFFF

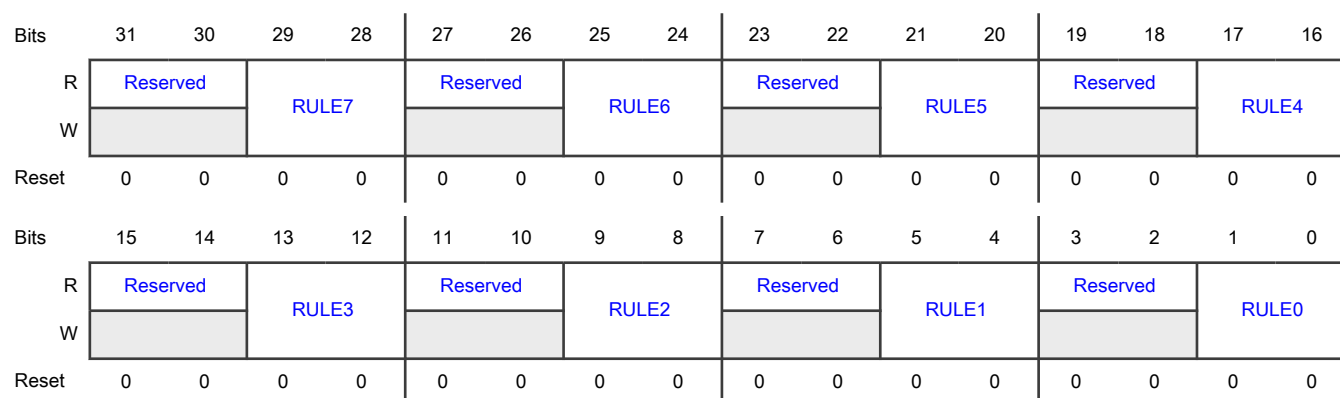
Table 256. RAM13_RULE2

Rule	Address Access
0	0x200B 0000 – 0x200B 0FFF
1	0x200B 1000 – 0x200B 1FFF
2	0x200B 2000 – 0x200B 2FFF
3	0x200B 3000 – 0x200B 3FFF
4	0x200B 4000 – 0x200B 4FFF
5	0x200B 5000 – 0x200B 5FFF
6	0x200B 6000 – 0x200B 6FFF
7	0x200B 7000 – 0x200B 7FFF

Table 257. RAM13_RULE3

Rule	Address Access
0	0x200B 8000 – 0x200B 8FFF
1	0x200B 9000 – 0x200B 9FFF
2	0x200B A000 – 0x200B AFFF
3	0x200B B000 – 0x200B BFFF
4	0x200B C000 – 0x200B CFFF
5	0x200B D000 – 0x200B DFFF
6	0x200B E000 – 0x200B EFFF
7	0x200B F000 – 0x200B FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.19 SRAM Partition 14 Rule(n) Register (RAM14_RULE0 - RAM14_RULE3)

Offset

Register	Offset
RAM14_RULE0	1B0h
RAM14_RULE1	1B4h
RAM14_RULE2	1B8h
RAM14_RULE3	1BCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 258. RAM14_RULE0

Rule	Address Access
0	0x200C 0000 – 0x200C 0FFF
1	0x200C 1000 – 0x200C 1FFF
2	0x200C 2000 – 0x200C 2FFF
3	0x200C 3000 – 0x200C 3FFF
4	0x200C 4000 – 0x200C 4FFF
5	0x200C 5000 – 0x200C 5FFF
6	0x200C 6000 – 0x200C 6FFF
7	0x200C 7000 – 0x200C 7FFF

Table 259. RAM14_RULE1

Rule	Address Access
0	0x200C 8000 – 0x200C 8FFF
1	0x200C 9000 – 0x200C 9FFF
2	0x200C A000 – 0x200C AFFF
3	0x200C B000 – 0x200C BFFF
4	0x200C C000 – 0x200C CFFF
5	0x200C D000 – 0x200C DFFF
6	0x200C E000 – 0x200C EFFF
7	0x200C F000 – 0x200C FFFF

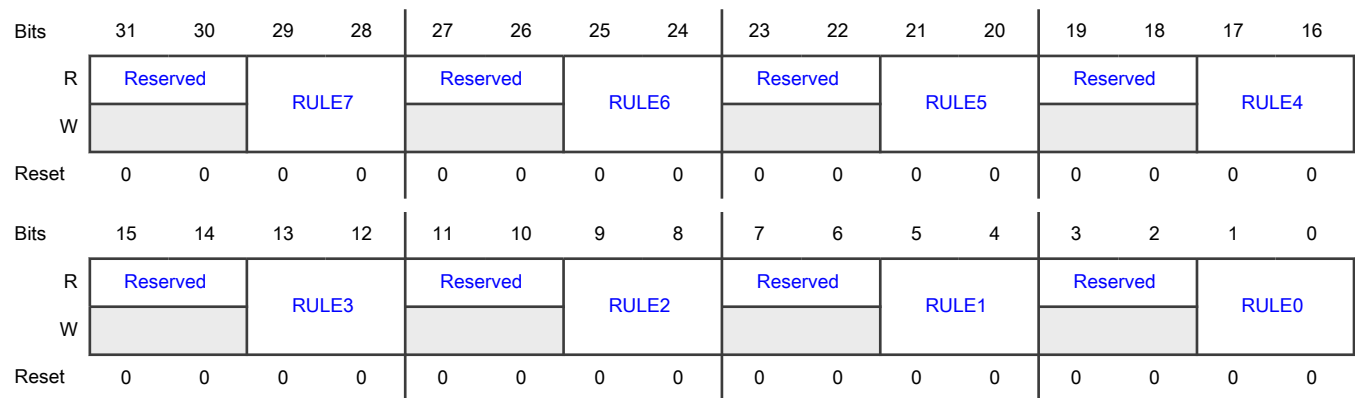
Table 260. RAM14_RULE2

Rule	Address Access
0	0x200D 0000 – 0x200D 0FFF
1	0x200D 1000 – 0x200D 1FFF
2	0x200D 2000 – 0x200D 2FFF
3	0x200D 3000 – 0x200D 3FFF
4	0x200D 4000 – 0x200D 4FFF
5	0x200D 5000 – 0x200D 5FFF
6	0x200D 6000 – 0x200D 6FFF
7	0x200D 7000 – 0x200D 7FFF

Table 261. RAM14_RULE3

Rule	Address Access
0	0x200D 8000 – 0x200D 8FFF
1	0x200D 9000 – 0x200D 9FFF
2	0x200D A000 – 0x200D AFFF
3	0x200D B000 – 0x200D BFFF
4	0x200D C000 – 0x200D CFFF
5	0x200D D000 – 0x200D DFFF
6	0x200D E000 – 0x200D EFFF
7	0x200D F000 – 0x200D FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.20 SRAM Partition 15 Rule(n) Register (RAM15_RULE0 - RAM15_RULE3)

Offset

Register	Offset
RAM15_RULE0	1C0h
RAM15_RULE1	1C4h
RAM15_RULE2	1C8h
RAM15_RULE3	1CCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 262. RAM15_RULE0

Rule	Address Access
0	0x200E 0000 – 0x200E 0FFF
1	0x200E 1000 – 0x200E 1FFF
2	0x200E 2000 – 0x200E 2FFF
3	0x200E 3000 – 0x200E 3FFF
4	0x200E 4000 – 0x200E 4FFF
5	0x200E 5000 – 0x200E 5FFF
6	0x200E 6000 – 0x200E 6FFF
7	0x200E 7000 – 0x200E 7FFF

Table 263. RAM15_RULE1

Rule	Address Access
0	0x200E 8000 – 0x200E 8FFF
1	0x200E 9000 – 0x200E 9FFF
2	0x200E A000 – 0x200E AFFF
3	0x200E B000 – 0x200E BFFF
4	0x200E C000 – 0x200E CFFF
5	0x200E D000 – 0x200E DFFF
6	0x200E E000 – 0x200E EFFF
7	0x200E F000 – 0x200E FFFF

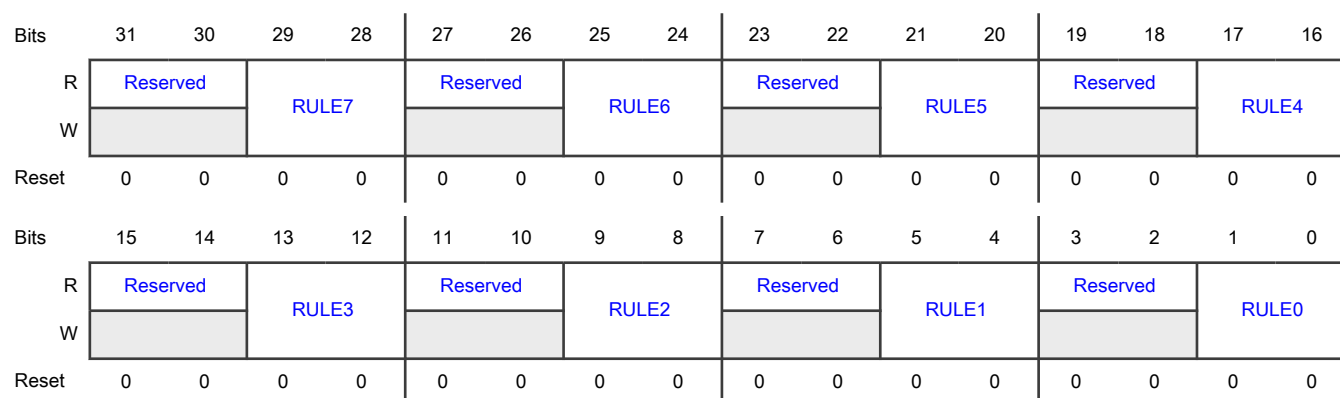
Table 264. RAM15_RULE2

Rule	Address Access
0	0x200F 0000 – 0x200F 0FFF
1	0x200F 1000 – 0x200F 1FFF
2	0x200F 2000 – 0x200F 2FFF
3	0x200F 3000 – 0x200F 3FFF
4	0x200F 4000 – 0x200F 4FFF
5	0x200F 5000 – 0x200F 5FFF
6	0x200F 6000 – 0x200F 6FFF
7	0x200F 7000 – 0x200F 7FFF

Table 265. RAM15_RULE3

Rule	Address Access
0	0x200F 8000 – 0x200F 8FFF
1	0x200F 9000 – 0x200F 9FFF
2	0x200F A000 – 0x200F AFFF
3	0x200F B000 – 0x200F BFFF
4	0x200F C000 – 0x200F CFFF
5	0x200F D000 – 0x200F DFFF
6	0x200F E000 – 0x200F EFFF
7	0x200F F000 – 0x200F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.21 SRAM Partition 16 Rule(n) Register (RAM16_RULE0 - RAM16_RULE3)

Offset

Register	Offset
RAM16_RULE0	1E0h
RAM16_RULE1	1E4h
RAM16_RULE2	1E8h
RAM16_RULE3	1ECh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 266. RAM16_RULE0

Rule	Address Access
0	0x2010 0000 – 0x2010 1FFF
1	0x2010 2000 – 0x2010 3FFF
2	0x2010 4000 – 0x2010 5FFF
3	0x2010 6000 – 0x2010 7FFF
4	0x2010 8000 – 0x2010 9FFF
5	0x2010 A000 – 0x2010 BFFF
6	0x2010 C000 – 0x2010 DFFF
7	0x2010 E000 – 0x2010 FFFF

Table 267. RAM16_RULE1

Rule	Address Access
0	0x2011 0000 – 0x2011 1FFF
1	0x2011 2000 – 0x2011 3FFF
2	0x2011 4000 – 0x2011 5FFF
3	0x2011 6000 – 0x2011 7FFF
4	0x2011 8000 – 0x2011 9FFF
5	0x2011 A000 – 0x2011 BFFF
6	0x2011 C000 – 0x2011 DFFF
7	0x2011 E000 – 0x2011 FFFF

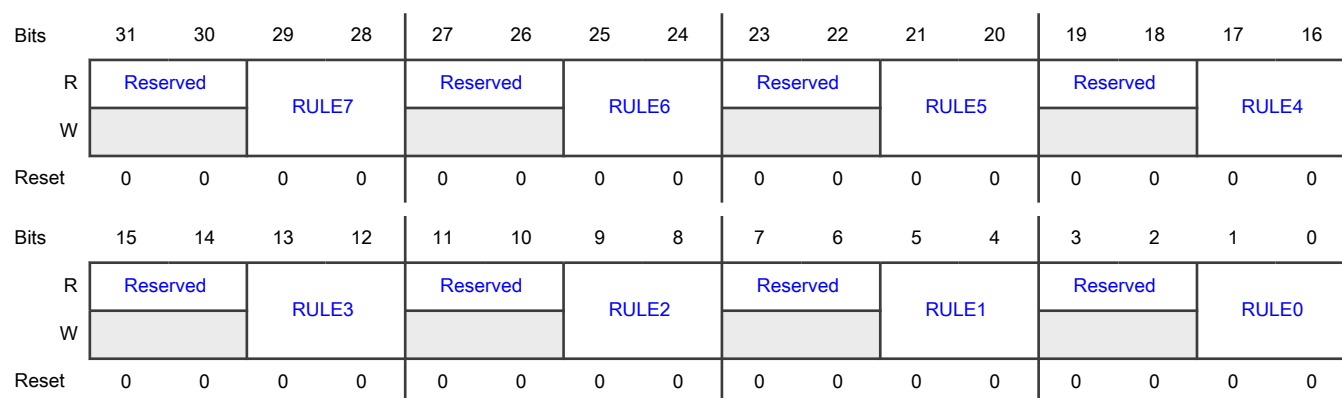
Table 268. RAM16_RULE2

Rule	Address Access
0	0x2012 0000 – 0x2012 1FFF
1	0x2012 2000 – 0x2012 3FFF
2	0x2012 4000 – 0x2012 5FFF
3	0x2012 6000 – 0x2012 7FFF
4	0x2012 8000 – 0x2012 9FFF
5	0x2012 A000 – 0x2012 BFFF
6	0x2012 C000 – 0x2012 DFFF
7	0x2012 E000 – 0x2012 FFFF

Table 269. RAM16_RULE3

Rule	Address Access
0	0x2013 0000 – 0x2013 1FFF
1	0x2013 2000 – 0x2013 3FFF
2	0x2013 4000 – 0x2013 5FFF
3	0x2013 6000 – 0x2013 7FFF
4	0x2013 8000 – 0x2013 9FFF
5	0x2013 A000 – 0x2013 BFFF
6	0x2013 C000 – 0x2013 DFFF
7	0x2013 E000 – 0x2013 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.22 SRAM Partition 17 Rule(n) Register (RAM17_RULE0 - RAM17_RULE3)

Offset

Register	Offset
RAM17_RULE0	1F0h
RAM17_RULE1	1F4h
RAM17_RULE2	1F8h
RAM17_RULE3	1FCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 270. RAM17_RULE0

Rule	Address Access
0	0x2014 0000 – 0x2014 1FFF
1	0x2014 2000 – 0x2014 3FFF
2	0x2014 4000 – 0x2014 5FFF
3	0x2014 6000 – 0x2014 7FFF
4	0x2014 8000 – 0x2014 9FFF
5	0x2014 A000 – 0x2014 BFFF
6	0x2014 C000 – 0x2014 DFFF
7	0x2014 E000 – 0x2014 FFFF

Table 271. RAM17_RULE1

Rule	Address Access
0	0x2015 0000 – 0x2015 1FFF
1	0x2015 2000 – 0x2015 3FFF
2	0x2015 4000 – 0x2015 5FFF
3	0x2015 6000 – 0x2015 7FFF
4	0x2015 8000 – 0x2015 9FFF
5	0x2015 A000 – 0x2015 BFFF
6	0x2015 C000 – 0x2015 DFFF
7	0x2015 E000 – 0x2015 FFFF

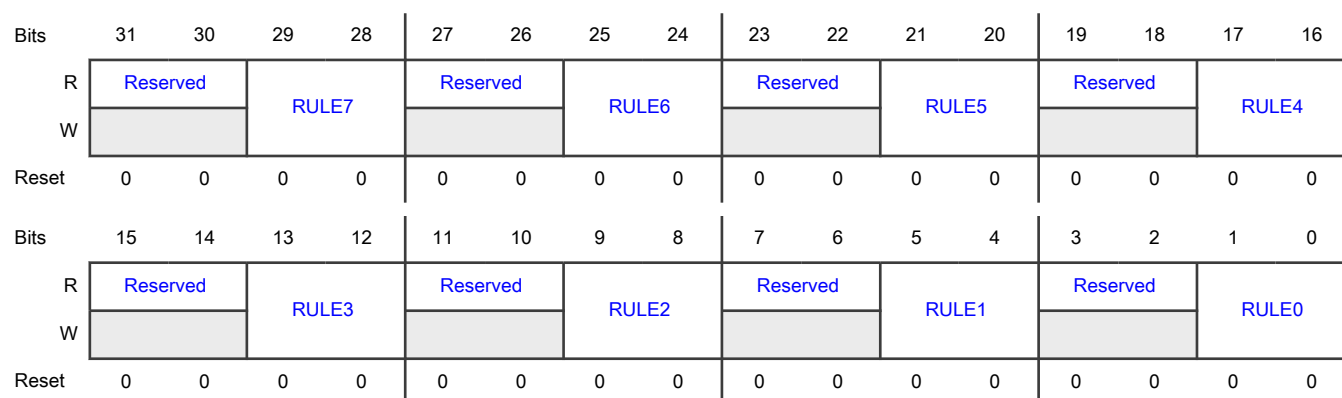
Table 272. RAM17_RULE2

Rule	Address Access
0	0x2016 0000 – 0x2016 1FFF
1	0x2016 2000 – 0x2016 3FFF
2	0x2016 4000 – 0x2016 5FFF
3	0x2016 6000 – 0x2016 7FFF
4	0x2016 8000 – 0x2016 9FFF
5	0x2016 A000 – 0x2016 BFFF
6	0x2016 C000 – 0x2016 DFFF
7	0x2016 E000 – 0x2016 FFFF

Table 273. RAM17_RULE3

Rule	Address Access
0	0x2017 0000 – 0x2017 1FFF
1	0x2017 2000 – 0x2017 3FFF
2	0x2017 4000 – 0x2017 5FFF
3	0x2017 6000 – 0x2017 7FFF
4	0x2017 8000 – 0x2017 9FFF
5	0x2017 A000 – 0x2017 BFFF
6	0x2017 C000 – 0x2017 DFFF
7	0x2017 E000 – 0x2017 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.23 SRAM Partition 18 Rule(n) Register (RAM18_RULE0 - RAM18_RULE3)

Offset

Register	Offset
RAM18_RULE0	200h
RAM18_RULE1	204h
RAM18_RULE2	208h
RAM18_RULE3	20Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 274. RAM18_RULE0

Rule	Address Access
0	0x2018 0000 – 0x2018 1FFF
1	0x2018 2000 – 0x2018 3FFF
2	0x2018 4000 – 0x2018 5FFF
3	0x2018 6000 – 0x2018 7FFF
4	0x2018 8000 – 0x2018 9FFF
5	0x2018 A000 – 0x2018 BFFF
6	0x2018 C000 – 0x2018 DFFF
7	0x2018 E000 – 0x2018 FFFF

Table 275. RAM18_RULE1

Rule	Address Access
0	0x2019 0000 – 0x2019 1FFF
1	0x2019 2000 – 0x2019 3FFF
2	0x2019 4000 – 0x2019 5FFF
3	0x2019 6000 – 0x2019 7FFF
4	0x2019 8000 – 0x2019 9FFF
5	0x2019 A000 – 0x2019 BFFF
6	0x2019 C000 – 0x2019 DFFF
7	0x2019 E000 – 0x2019 FFFF

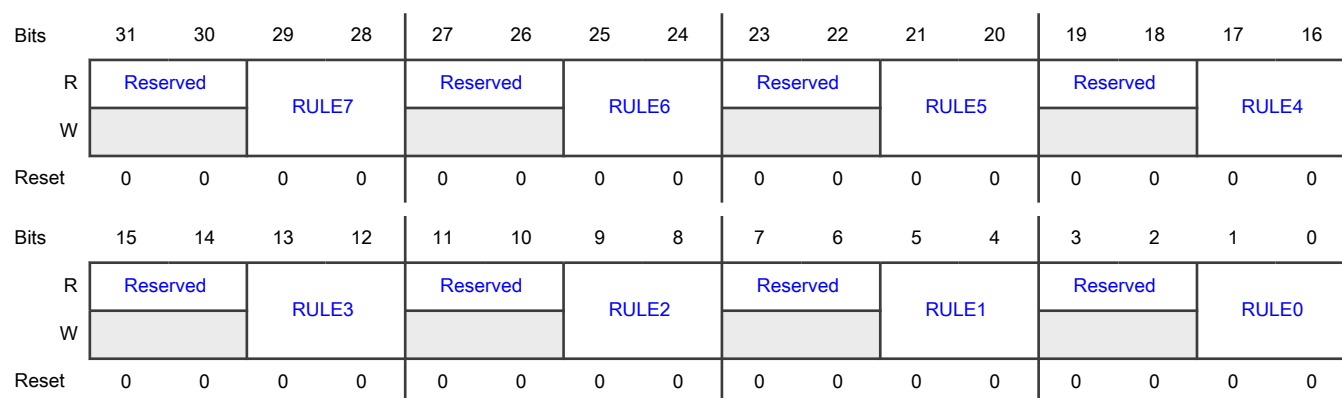
Table 276. RAM18_RULE2

Rule	Address Access
0	0x201A 0000 – 0x201A 1FFF
1	0x201A 2000 – 0x201A 3FFF
2	0x201A 4000 – 0x201A 5FFF
3	0x201A 6000 – 0x201A 7FFF
4	0x201A 8000 – 0x201A 9FFF
5	0x201A A000 – 0x201A BFFF
6	0x201A C000 – 0x201A DFFF
7	0x201A E000 – 0x201A FFFF

Table 277. RAM18_RULE3

Rule	Address Access
0	0x201B 0000 – 0x201B 1FFF
1	0x201B 2000 – 0x201B 3FFF
2	0x201B 4000 – 0x201B 5FFF
3	0x201B 6000 – 0x201B 7FFF
4	0x201B 8000 – 0x201B 9FFF
5	0x201B A000 – 0x201B BFFF
6	0x201B C000 – 0x201B DFFF
7	0x201B E000 – 0x201B FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.24 SRAM Partition 19 Rule(n) Register (RAM19_RULE0 - RAM19_RULE3)

Offset

Register	Offset
RAM19_RULE0	210h
RAM19_RULE1	214h
RAM19_RULE2	218h
RAM19_RULE3	21Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 278. RAM19_RULE0

Rule	Address Access
0	0x201C 0000 – 0x201C 1FFF
1	0x201C 2000 – 0x201C 3FFF
2	0x201C 4000 – 0x201C 5FFF
3	0x201C 6000 – 0x201C 7FFF
4	0x201C 8000 – 0x201C 9FFF
5	0x201C A000 – 0x201C BFFF
6	0x201C C000 – 0x201C DFFF
7	0x201C E000 – 0x201C FFFF

Table 279. RAM19_RULE1

Rule	Address Access
0	0x201D 0000 – 0x201D 1FFF
1	0x201D 2000 – 0x201D 3FFF
2	0x201D 4000 – 0x201D 5FFF
3	0x201D 6000 – 0x201D 7FFF
4	0x201D 8000 – 0x201D 9FFF
5	0x201D A000 – 0x201D BFFF
6	0x201D C000 – 0x201D DFFF
7	0x201D E000 – 0x201D FFFF

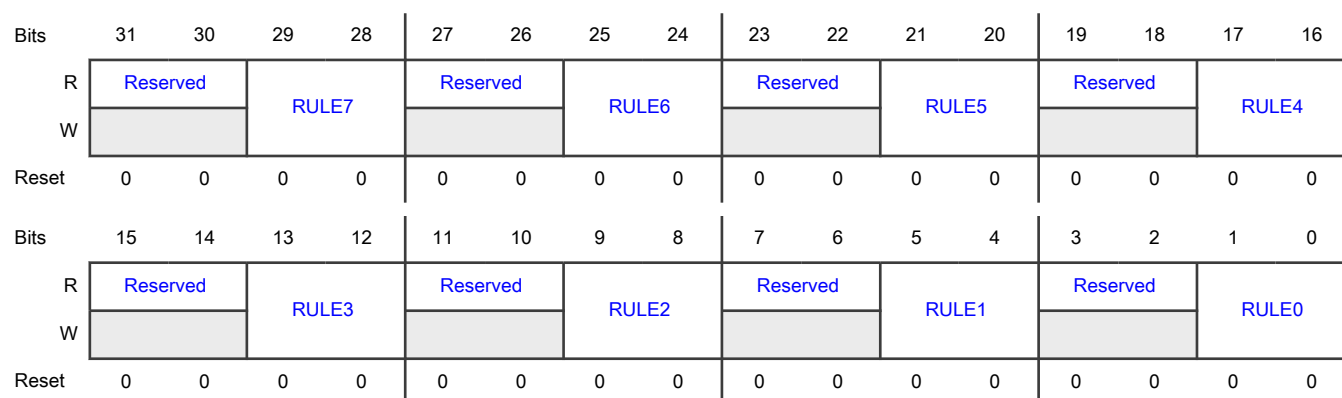
Table 280. RAM19_RULE2

Rule	Address Access
0	0x201E 0000 – 0x201E 1FFF
1	0x201E 2000 – 0x201E 3FFF
2	0x201E 4000 – 0x201E 5FFF
3	0x201E 6000 – 0x201E 7FFF
4	0x201E 8000 – 0x201E 9FFF
5	0x201E A000 – 0x201E BFFF
6	0x201E C000 – 0x201E DFFF
7	0x201E E000 – 0x201E FFFF

Table 281. RAM19_RULE3

Rule	Address Access
0	0x201F 0000 – 0x201F 1FFF
1	0x201F 2000 – 0x201F 3FFF
2	0x201F 4000 – 0x201F 5FFF
3	0x201F 6000 – 0x201F 7FFF
4	0x201F 8000 – 0x201F 9FFF
5	0x201F A000 – 0x201F BFFF
6	0x201F C000 – 0x201F DFFF
7	0x201F E000 – 0x201F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.25 SRAM Partition 20 Rule(n) Register (RAM20_RULE0 - RAM20_RULE3)

Offset

Register	Offset
RAM20_RULE0	230h
RAM20_RULE1	234h
RAM20_RULE2	238h
RAM20_RULE3	23Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 282. RAM20_RULE0

Rule	Address Access
0	0x2020 0000 – 0x2020 1FFF
1	0x2020 2000 – 0x2020 3FFF
2	0x2020 4000 – 0x2020 5FFF
3	0x2020 6000 – 0x2020 7FFF
4	0x2020 8000 – 0x2020 9FFF
5	0x2020 A000 – 0x2020 BFFF
6	0x2020 C000 – 0x2020 DFFF
7	0x2020 E000 – 0x2020 FFFF

Table 283. RAM20_RULE1

Rule	Address Access
0	0x2021 0000 – 0x2021 1FFF
1	0x2021 2000 – 0x2021 3FFF
2	0x2021 4000 – 0x2021 5FFF
3	0x2021 6000 – 0x2021 7FFF
4	0x2021 8000 – 0x2021 9FFF
5	0x2021 A000 – 0x2021 BFFF
6	0x2021 C000 – 0x2021 DFFF
7	0x2021 E000 – 0x2021 FFFF

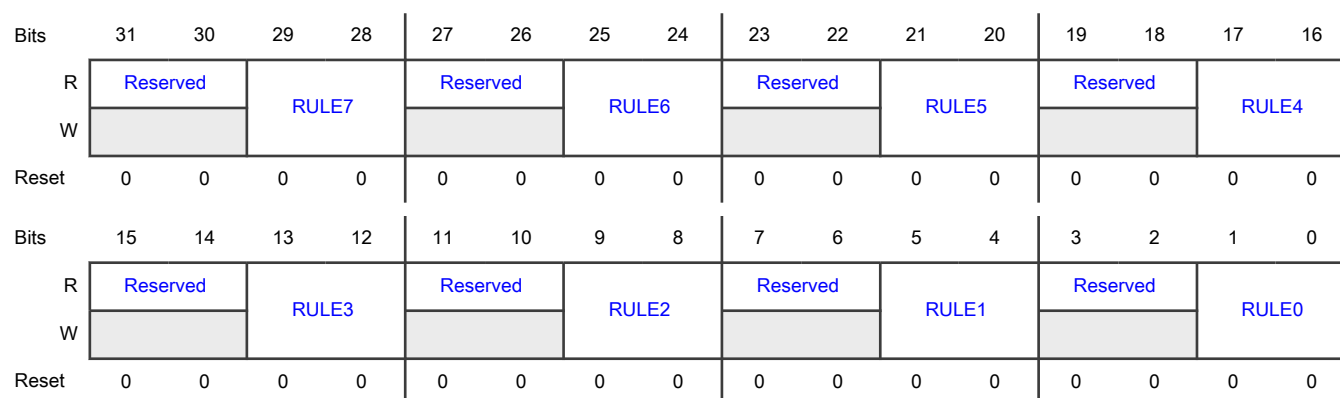
Table 284. RAM20_RULE2

Rule	Address Access
0	0x2022 0000 – 0x2022 1FFF
1	0x2022 2000 – 0x2022 3FFF
2	0x2022 4000 – 0x2022 5FFF
3	0x2022 6000 – 0x2022 7FFF
4	0x2022 8000 – 0x2022 9FFF
5	0x2022 A000 – 0x2022 BFFF
6	0x2022 C000 – 0x2022 DFFF
7	0x2022 E000 – 0x2022 FFFF

Table 285. RAM20_RULE3

Rule	Address Access
0	0x2023 0000 – 0x2023 1FFF
1	0x2023 2000 – 0x2023 3FFF
2	0x2023 4000 – 0x2023 5FFF
3	0x2023 6000 – 0x2023 7FFF
4	0x2023 8000 – 0x2023 9FFF
5	0x2023 A000 – 0x2023 BFFF
6	0x2023 C000 – 0x2023 DFFF
7	0x2023 E000 – 0x2023 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.26 SRAM Partition 21 Rule(n) Register (RAM21_RULE0 - RAM21_RULE3)

Offset

Register	Offset
RAM21_RULE0	240h
RAM21_RULE1	244h
RAM21_RULE2	248h
RAM21_RULE3	24Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 286. RAM21_RULE0

Rule	Address Access
0	0x2024 0000 – 0x2024 1FFF
1	0x2024 2000 – 0x2024 3FFF
2	0x2024 4000 – 0x2024 5FFF
3	0x2024 6000 – 0x2024 7FFF
4	0x2024 8000 – 0x2024 9FFF
5	0x2024 A000 – 0x2024 BFFF
6	0x2024 C000 – 0x2024 DFFF
7	0x2024 E000 – 0x2024 FFFF

Table 287. RAM21_RULE1

Rule	Address Access
0	0x2025 0000 – 0x2025 1FFF
1	0x2025 2000 – 0x2025 3FFF
2	0x2025 4000 – 0x2025 5FFF
3	0x2025 6000 – 0x2025 7FFF
4	0x2025 8000 – 0x2025 9FFF
5	0x2025 A000 – 0x2025 BFFF
6	0x2025 C000 – 0x2025 DFFF
7	0x2025 E000 – 0x2025 FFFF

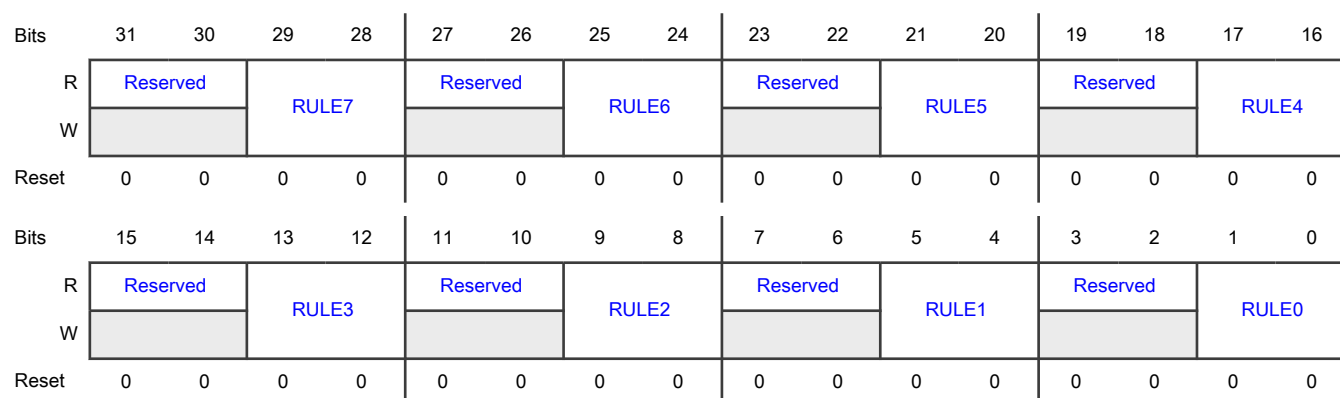
Table 288. RAM21_RULE2

Rule	Address Access
0	0x2026 0000 – 0x2026 1FFF
1	0x2026 2000 – 0x2026 3FFF
2	0x2026 4000 – 0x2026 5FFF
3	0x2026 6000 – 0x2026 7FFF
4	0x2026 8000 – 0x2026 9FFF
5	0x2026 A000 – 0x2026 BFFF
6	0x2026 C000 – 0x2026 DFFF
7	0x2026 E000 – 0x2026 FFFF

Table 289. RAM21_RULE3

Rule	Address Access
0	0x2027 0000 – 0x2027 1FFF
1	0x2027 2000 – 0x2027 3FFF
2	0x2027 4000 – 0x2027 5FFF
3	0x2027 6000 – 0x2027 7FFF
4	0x2027 8000 – 0x2027 9FFF
5	0x2027 A000 – 0x2027 BFFF
6	0x2027 C000 – 0x2027 DFFF
7	0x2027 E000 – 0x2027 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.27 SRAM Partition 22 Rule(n) Register (RAM22_RULE0 - RAM22_RULE3)

Offset

Register	Offset
RAM22_RULE0	250h
RAM22_RULE1	254h
RAM22_RULE2	258h
RAM22_RULE3	25Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 290. RAM22_RULE0

Rule	Address Access
0	0x2028 0000 – 0x2028 1FFF
1	0x2028 2000 – 0x2028 3FFF
2	0x2028 4000 – 0x2028 5FFF
3	0x2028 6000 – 0x2028 7FFF
4	0x2028 8000 – 0x2028 9FFF
5	0x2028 A000 – 0x2028 BFFF
6	0x2028 C000 – 0x2028 DFFF
7	0x2028 E000 – 0x2028 FFFF

Table 291. RAM22_RULE1

Rule	Address Access
0	0x2029 0000 – 0x2029 1FFF
1	0x2029 2000 – 0x2029 3FFF
2	0x2029 4000 – 0x2029 5FFF
3	0x2029 6000 – 0x2029 7FFF
4	0x2029 8000 – 0x2029 9FFF
5	0x2029 A000 – 0x2029 BFFF
6	0x2029 C000 – 0x2029 DFFF
7	0x2029 E000 – 0x2029 FFFF

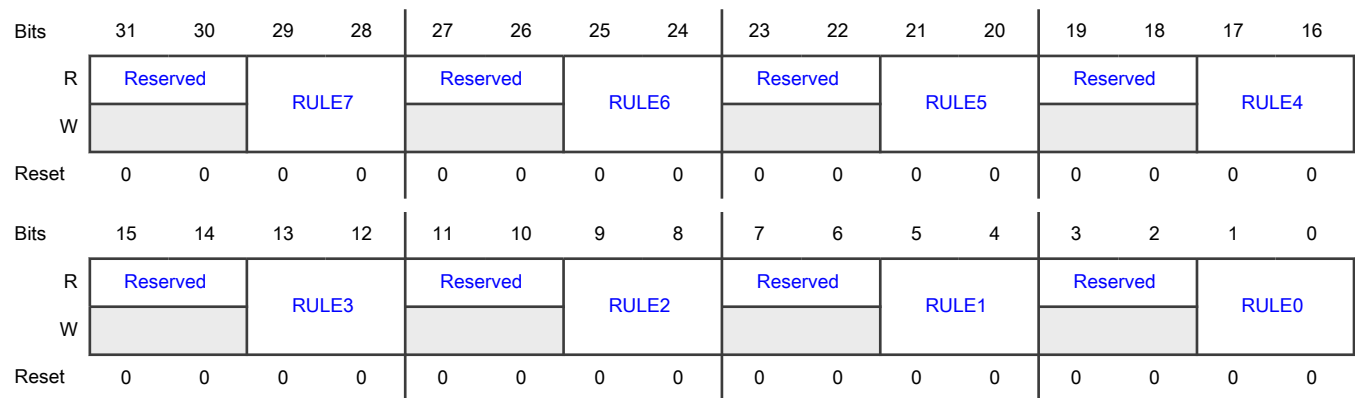
Table 292. RAM22_RULE2

Rule	Address Access
0	0x202A 0000 – 0x202A 1FFF
1	0x202A 2000 – 0x202A 3FFF
2	0x202A 4000 – 0x202A 5FFF
3	0x202A 6000 – 0x202A 7FFF
4	0x202A 8000 – 0x202A 9FFF
5	0x202A A000 – 0x202A BFFF
6	0x202A C000 – 0x202A DFFF
7	0x202A E000 – 0x202A FFFF

Table 293. RAM22_RULE3

Rule	Address Access
0	0x202B 0000 – 0x202B 1FFF
1	0x202B 2000 – 0x202B 3FFF
2	0x202B 4000 – 0x202B 5FFF
3	0x202B 6000 – 0x202B 7FFF
4	0x202B 8000 – 0x202B 9FFF
5	0x202B A000 – 0x202B BFFF
6	0x202B C000 – 0x202B DFFF
7	0x202B E000 – 0x202B FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.28 SRAM Partition 23 Rule(n) Register (RAM23_RULE0 - RAM23_RULE3)

Offset

Register	Offset
RAM23_RULE0	260h
RAM23_RULE1	264h
RAM23_RULE2	268h
RAM23_RULE3	26Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 294. RAM23_RULE0

Rule	Address Access
0	0x202C 0000 – 0x202C 1FFF
1	0x202C 2000 – 0x202C 3FFF
2	0x202C 4000 – 0x202C 5FFF
3	0x202C 6000 – 0x202C 7FFF
4	0x202C 8000 – 0x202C 9FFF
5	0x202C A000 – 0x202C BFFF
6	0x202C C000 – 0x202C DFFF
7	0x202C E000 – 0x202C FFFF

Table 295. RAM23_RULE1

Rule	Address Access
0	0x202D 0000 – 0x202D 1FFF
1	0x202D 2000 – 0x202D 3FFF
2	0x202D 4000 – 0x202D 5FFF
3	0x202D 6000 – 0x202D 7FFF
4	0x202D 8000 – 0x202D 9FFF
5	0x202D A000 – 0x202D BFFF
6	0x202D C000 – 0x202D DFFF
7	0x202D E000 – 0x202D FFFF

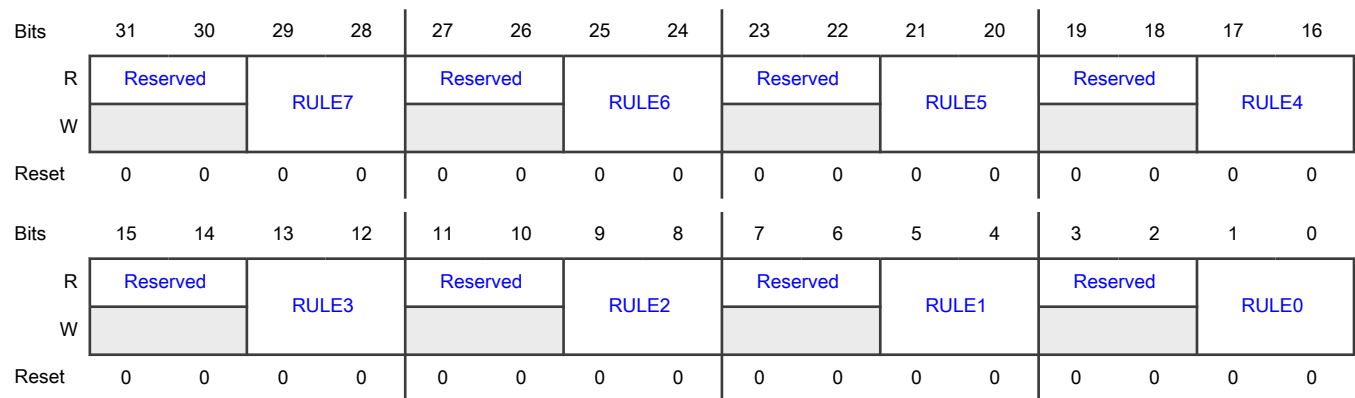
Table 296. RAM23_RULE2

Rule	Address Access
0	0x202E 0000 – 0x202E 1FFF
1	0x202E 2000 – 0x202E 3FFF
2	0x202E 4000 – 0x202E 5FFF
3	0x202E 6000 – 0x202E 7FFF
4	0x202E 8000 – 0x202E 9FFF
5	0x202E A000 – 0x202E BFFF
6	0x202E C000 – 0x202E DFFF
7	0x202E E000 – 0x202E FFFF

Table 297. RAM23_RULE3

Rule	Address Access
0	0x202F 0000 – 0x202F 1FFF
1	0x202F 2000 – 0x202F 3FFF
2	0x202F 4000 – 0x202F 5FFF
3	0x202F 6000 – 0x202F 7FFF
4	0x202F 8000 – 0x202F 9FFF
5	0x202F A000 – 0x202F BFFF
6	0x202F C000 – 0x202F DFFF
7	0x202F E000 – 0x202F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.29 SRAM Partition 24 Rule(n) Register (RAM24_RULE0 - RAM24_RULE3)

Offset

Register	Offset
RAM24_RULE0	280h
RAM24_RULE1	284h
RAM24_RULE2	288h
RAM24_RULE3	28Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 298. RAM24_RULE0

Rule	Address Access
0	0x2030 0000 – 0x2030 1FFF
1	0x2030 2000 – 0x2030 3FFF
2	0x2030 4000 – 0x2030 5FFF
3	0x2030 6000 – 0x2030 7FFF
4	0x2030 8000 – 0x2030 9FFF
5	0x2030 A000 – 0x2030 BFFF
6	0x2030 C000 – 0x2030 DFFF
7	0x2030 E000 – 0x2030 FFFF

Table 299. RAM24_RULE1

Rule	Address Access
0	0x2031 0000 – 0x2031 1FFF
1	0x2031 2000 – 0x2031 3FFF
2	0x2031 4000 – 0x2031 5FFF
3	0x2031 6000 – 0x2031 7FFF
4	0x2031 8000 – 0x2031 9FFF
5	0x2031 A000 – 0x2031 BFFF
6	0x2031 C000 – 0x2031 DFFF
7	0x2031 E000 – 0x2031 FFFF

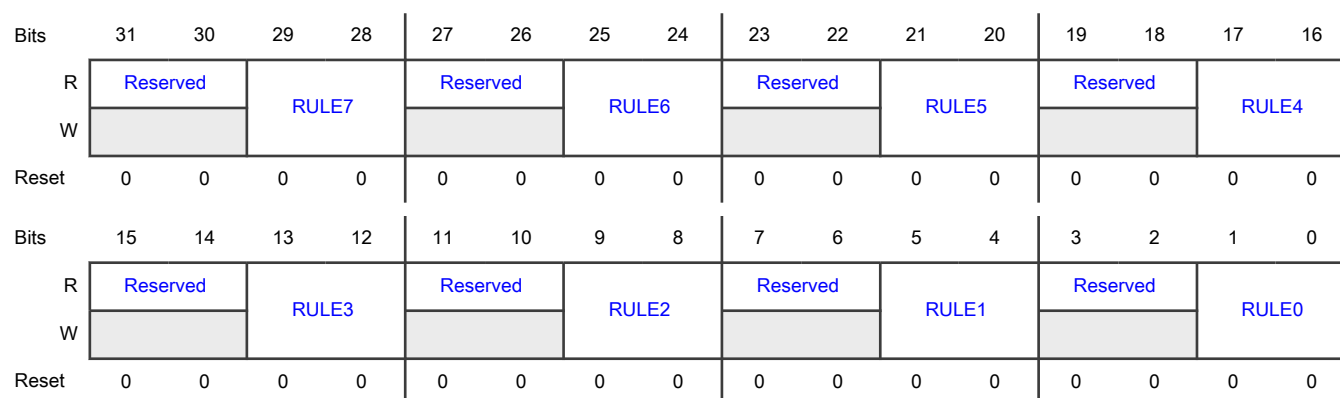
Table 300. RAM24_RULE2

Rule	Address Access
0	0x2032 0000 – 0x2032 1FFF
1	0x2032 2000 – 0x2032 3FFF
2	0x2032 4000 – 0x2032 5FFF
3	0x2032 6000 – 0x2032 7FFF
4	0x2032 8000 – 0x2032 9FFF
5	0x2032 A000 – 0x2032 BFFF
6	0x2032 C000 – 0x2032 DFFF
7	0x2032 E000 – 0x2032 FFFF

Table 301. RAM24_RULE3

Rule	Address Access
0	0x2033 0000 – 0x2033 1FFF
1	0x2033 2000 – 0x2033 3FFF
2	0x2033 4000 – 0x2033 5FFF
3	0x2033 6000 – 0x2033 7FFF
4	0x2033 8000 – 0x2033 9FFF
5	0x2033 A000 – 0x2033 BFFF
6	0x2033 C000 – 0x2033 DFFF
7	0x2033 E000 – 0x2033 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.30 SRAM Partition 25 Rule(n) Register (RAM25_RULE0 - RAM25_RULE3)

Offset

Register	Offset
RAM25_RULE0	290h
RAM25_RULE1	294h
RAM25_RULE2	298h
RAM25_RULE3	29Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 302. RAM25_RULE0

Rule	Address Access
0	0x2034 0000 – 0x2034 1FFF
1	0x2034 2000 – 0x2034 3FFF
2	0x2034 4000 – 0x2034 5FFF
3	0x2034 6000 – 0x2034 7FFF
4	0x2034 8000 – 0x2034 9FFF
5	0x2034 A000 – 0x2034 BFFF
6	0x2034 C000 – 0x2034 DFFF
7	0x2034 E000 – 0x2034 FFFF

Table 303. RAM25_RULE1

Rule	Address Access
0	0x2035 0000 – 0x2035 1FFF
1	0x2035 2000 – 0x2035 3FFF
2	0x2035 4000 – 0x2035 5FFF
3	0x2035 6000 – 0x2035 7FFF
4	0x2035 8000 – 0x2035 9FFF
5	0x2035 A000 – 0x2035 BFFF
6	0x2035 C000 – 0x2035 DFFF
7	0x2035 E000 – 0x2035 FFFF

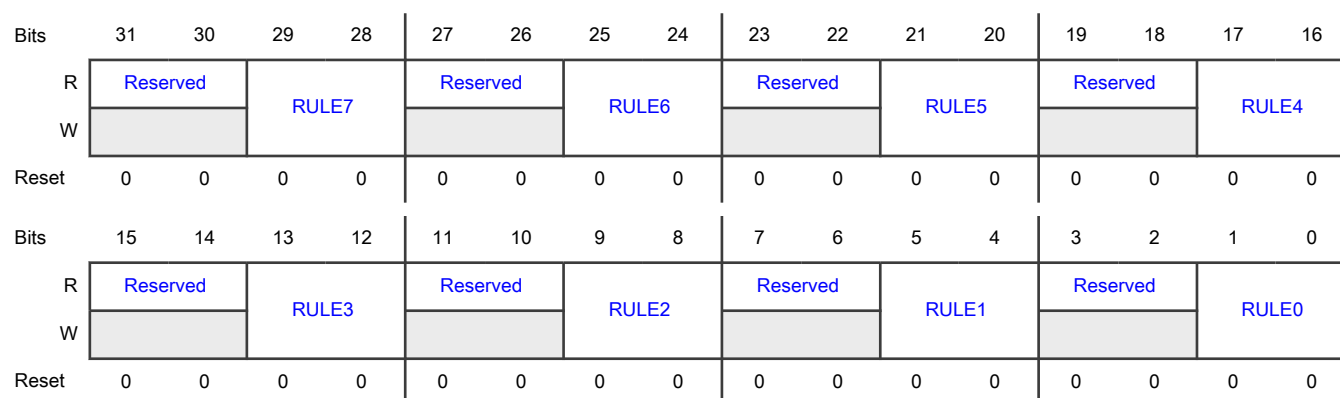
Table 304. RAM25_RULE2

Rule	Address Access
0	0x2036 0000 – 0x2036 1FFF
1	0x2036 2000 – 0x2036 3FFF
2	0x2036 4000 – 0x2036 5FFF
3	0x2036 6000 – 0x2036 7FFF
4	0x2036 8000 – 0x2036 9FFF
5	0x2036 A000 – 0x2036 BFFF
6	0x2036 C000 – 0x2036 DFFF
7	0x2036 E000 – 0x2036 FFFF

Table 305. RAM25_RULE3

Rule	Address Access
0	0x2037 0000 – 0x2037 1FFF
1	0x2037 2000 – 0x2037 3FFF
2	0x2037 4000 – 0x2037 5FFF
3	0x2037 6000 – 0x2037 7FFF
4	0x2037 8000 – 0x2037 9FFF
5	0x2037 A000 – 0x2037 BFFF
6	0x2037 C000 – 0x2037 DFFF
7	0x2037 E000 – 0x2037 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.31 SRAM Partition 26 Rule(n) Register (RAM26_RULE0 - RAM26_RULE3)

Offset

Register	Offset
RAM26_RULE0	2A0h
RAM26_RULE1	2A4h
RAM26_RULE2	2A8h
RAM26_RULE3	2ACh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 306. RAM26_RULE0

Rule	Address Access
0	0x2038 0000 – 0x2038 1FFF
1	0x2038 2000 – 0x2038 3FFF
2	0x2038 4000 – 0x2038 5FFF
3	0x2038 6000 – 0x2038 7FFF
4	0x2038 8000 – 0x2038 9FFF
5	0x2038 A000 – 0x2038 BFFF
6	0x2038 C000 – 0x2038 DFFF
7	0x2038 E000 – 0x2038 FFFF

Table 307. RAM26_RULE1

Rule	Address Access
0	0x2039 0000 – 0x2039 1FFF
1	0x2039 2000 – 0x2039 3FFF
2	0x2039 4000 – 0x2039 5FFF
3	0x2039 6000 – 0x2039 7FFF
4	0x2039 8000 – 0x2039 9FFF
5	0x2039 A000 – 0x2039 BFFF
6	0x2039 C000 – 0x2039 DFFF
7	0x2039 E000 – 0x2039 FFFF

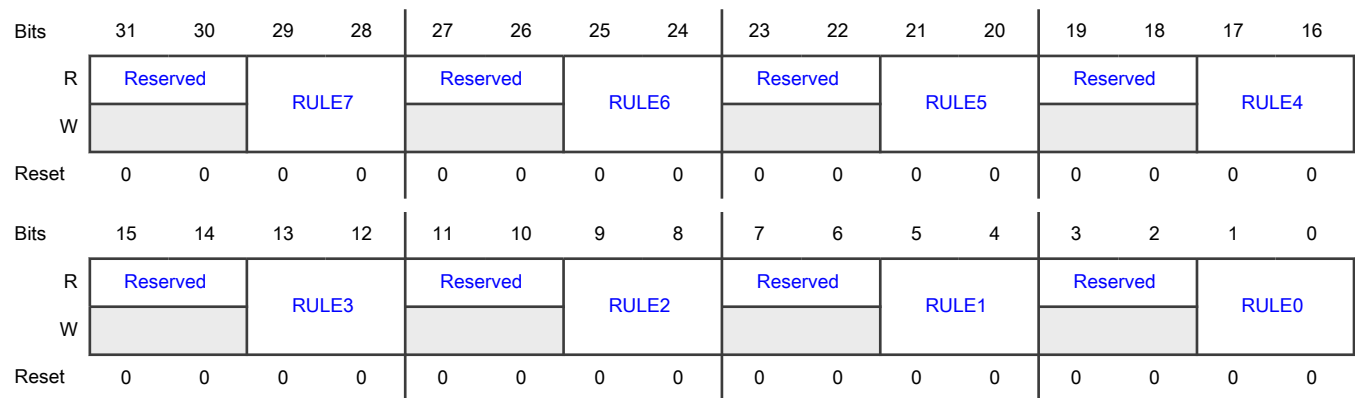
Table 308. RAM26_RULE2

Rule	Address Access
0	0x203A 0000 – 0x203A 1FFF
1	0x203A 2000 – 0x203A 3FFF
2	0x203A 4000 – 0x203A 5FFF
3	0x203A 6000 – 0x203A 7FFF
4	0x203A 8000 – 0x203A 9FFF
5	0x203A A000 – 0x203A BFFF
6	0x203A C000 – 0x203A DFFF
7	0x203A E000 – 0x203A FFFF

Table 309. RAM26_RULE3

Rule	Address Access
0	0x203B 0000 – 0x203B 1FFF
1	0x203B 2000 – 0x203B 3FFF
2	0x203B 4000 – 0x203B 5FFF
3	0x203B 6000 – 0x203B 7FFF
4	0x203B 8000 – 0x203B 9FFF
5	0x203B A000 – 0x203B BFFF
6	0x203B C000 – 0x203B DFFF
7	0x203B E000 – 0x203B FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.32 SRAM Partition 27 Rule(n) Register (RAM27_RULE0 - RAM27_RULE3)

Offset

Register	Offset
RAM27_RULE0	2B0h
RAM27_RULE1	2B4h
RAM27_RULE2	2B8h
RAM27_RULE3	2BCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 310. RAM27_RULE0

Rule	Address Access
0	0x203C 0000 – 0x203C 1FFF
1	0x203C 2000 – 0x203C 3FFF
2	0x203C 4000 – 0x203C 5FFF
3	0x203C 6000 – 0x203C 7FFF
4	0x203C 8000 – 0x203C 9FFF
5	0x203C A000 – 0x203C BFFF
6	0x203C C000 – 0x203C DFFF
7	0x203C E000 – 0x203C FFFF

Table 311. RAM27_RULE1

Rule	Address Access
0	0x203D 0000 – 0x203D 1FFF
1	0x203D 2000 – 0x203D 3FFF
2	0x203D 4000 – 0x203D 5FFF
3	0x203D 6000 – 0x203D 7FFF
4	0x203D 8000 – 0x203D 9FFF
5	0x203D A000 – 0x203D BFFF
6	0x203D C000 – 0x203D DFFF
7	0x203D E000 – 0x203D FFFF

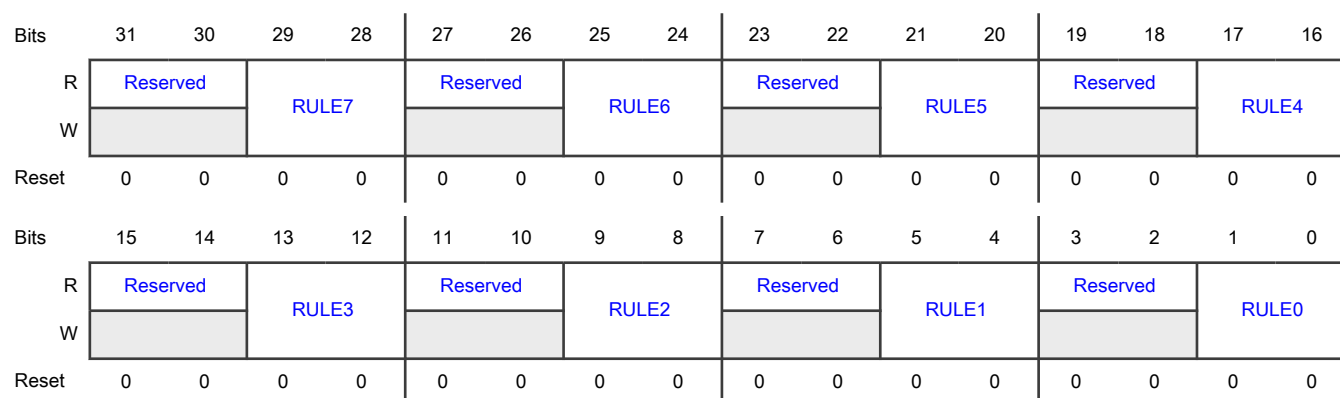
Table 312. RAM27_RULE2

Rule	Address Access
0	0x203E 0000 – 0x203E 1FFF
1	0x203E 2000 – 0x203E 3FFF
2	0x203E 4000 – 0x203E 5FFF
3	0x203E 6000 – 0x203E 7FFF
4	0x203E 8000 – 0x203E 9FFF
5	0x203E A000 – 0x203E BFFF
6	0x203E C000 – 0x203E DFFF
7	0x203E E000 – 0x203E FFFF

Table 313. RAM27_RULE3

Rule	Address Access
0	0x203F 0000 – 0x203F 1FFF
1	0x203F 2000 – 0x203F 3FFF
2	0x203F 4000 – 0x203F 5FFF
3	0x203F 6000 – 0x203F 7FFF
4	0x203F 8000 – 0x203F 9FFF
5	0x203F A000 – 0x203F BFFF
6	0x203F C000 – 0x203F DFFF
7	0x203F E000 – 0x203F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.33 SRAM Partition 28 Rule(n) Register (RAM28_RULE0 - RAM28_RULE3)

Offset

Register	Offset
RAM28_RULE0	2D0h
RAM28_RULE1	2D4h
RAM28_RULE2	2D8h
RAM28_RULE3	2DCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 314. RAM28_RULE0

Rule	Address Access
0	0x2040 0000 – 0x2040 1FFF
1	0x2040 2000 – 0x2040 3FFF
2	0x2040 4000 – 0x2040 5FFF
3	0x2040 6000 – 0x2040 7FFF
4	0x2040 8000 – 0x2040 9FFF
5	0x2040 A000 – 0x2040 BFFF
6	0x2040 C000 – 0x2040 DFFF
7	0x2040 E000 – 0x2040 FFFF

Table 315. RAM28_RULE1

Rule	Address Access
0	0x2041 0000 – 0x2041 1FFF
1	0x2041 2000 – 0x2041 3FFF
2	0x2041 4000 – 0x2041 5FFF
3	0x2041 6000 – 0x2041 7FFF
4	0x2041 8000 – 0x2041 9FFF
5	0x2041 A000 – 0x2041 BFFF
6	0x2041 C000 – 0x2041 DFFF
7	0x2041 E000 – 0x2041 FFFF

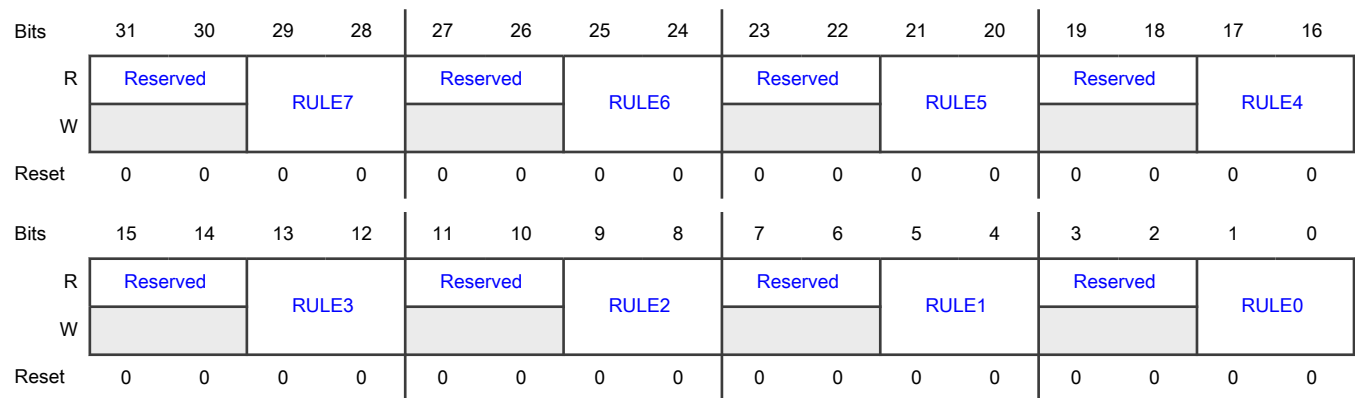
Table 316. RAM28_RULE2

Rule	Address Access
0	0x2042 0000 – 0x2042 1FFF
1	0x2042 2000 – 0x2042 3FFF
2	0x2042 4000 – 0x2042 5FFF
3	0x2042 6000 – 0x2042 7FFF
4	0x2042 8000 – 0x2042 9FFF
5	0x2042 A000 – 0x2042 BFFF
6	0x2042 C000 – 0x2042 DFFF
7	0x2042 E000 – 0x2042 FFFF

Table 317. RAM28_RULE3

Rule	Address Access
0	0x2043 0000 – 0x2043 1FFF
1	0x2043 2000 – 0x2043 3FFF
2	0x2043 4000 – 0x2043 5FFF
3	0x2043 6000 – 0x2043 7FFF
4	0x2043 8000 – 0x2043 9FFF
5	0x2043 A000 – 0x2043 BFFF
6	0x2043 C000 – 0x2043 DFFF
7	0x2043 E000 – 0x2043 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.34 SRAM Partition 29 Rule(n) Register (RAM29_RULE0 - RAM29_RULE3)

Offset

Register	Offset
RAM29_RULE0	2E0h
RAM29_RULE1	2E4h
RAM29_RULE2	2E8h
RAM29_RULE3	2ECh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 318. RAM29_RULE0

Rule	Address Access
0	0x2044 0000 – 0x2044 1FFF
1	0x2044 2000 – 0x2044 3FFF
2	0x2044 4000 – 0x2044 5FFF
3	0x2044 6000 – 0x2044 7FFF
4	0x2044 8000 – 0x2044 9FFF
5	0x2044 A000 – 0x2044 BFFF
6	0x2044 C000 – 0x2044 DFFF
7	0x2044 E000 – 0x2044 FFFF

Table 319. RAM29_RULE1

Rule	Address Access
0	0x2045 0000 – 0x2045 1FFF
1	0x2045 2000 – 0x2045 3FFF
2	0x2045 4000 – 0x2045 5FFF
3	0x2045 6000 – 0x2045 7FFF
4	0x2045 8000 – 0x2045 9FFF
5	0x2045 A000 – 0x2045 BFFF
6	0x2045 C000 – 0x2045 DFFF
7	0x2045 E000 – 0x2045 FFFF

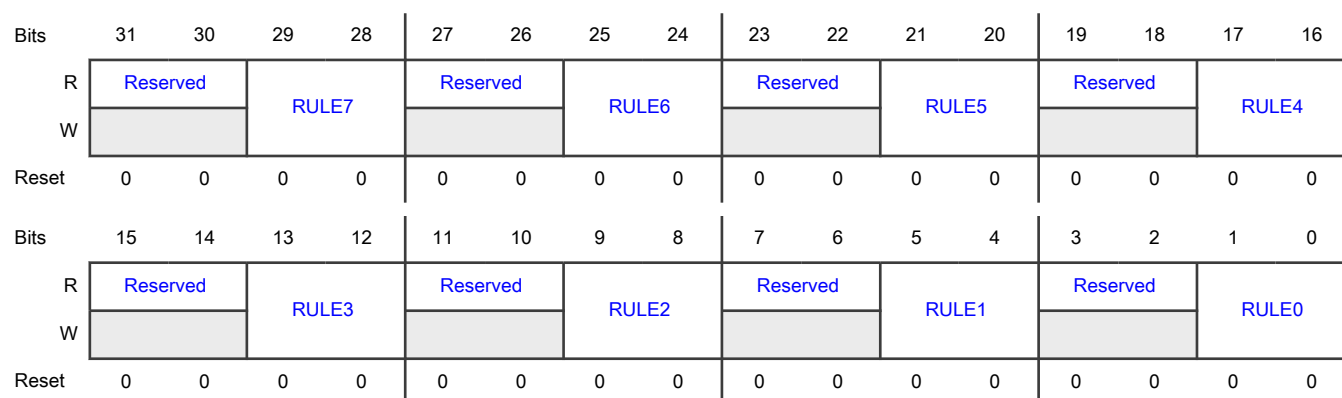
Table 320. RAM29_RULE2

Rule	Address Access
0	0x2046 0000 – 0x2046 1FFF
1	0x2046 2000 – 0x2046 3FFF
2	0x2046 4000 – 0x2046 5FFF
3	0x2046 6000 – 0x2046 7FFF
4	0x2046 8000 – 0x2046 9FFF
5	0x2046 A000 – 0x2046 BFFF
6	0x2046 C000 – 0x2046 DFFF
7	0x2046 E000 – 0x2046 FFFF

Table 321. RAM29_RULE3

Rule	Address Access
0	0x2047 0000 – 0x2047 1FFF
1	0x2047 2000 – 0x2047 3FFF
2	0x2047 4000 – 0x2047 5FFF
3	0x2047 6000 – 0x2047 7FFF
4	0x2047 8000 – 0x2047 9FFF
5	0x2047 A000 – 0x2047 BFFF
6	0x2047 C000 – 0x2047 DFFF
7	0x2047 E000 – 0x2047 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.35 SRAM Partition 30 Rule(n) Register (RAM30_RULE0 - RAM30_RULE3)

Offset

Register	Offset
RAM30_RULE0	2F0h
RAM30_RULE1	2F4h
RAM30_RULE2	2F8h
RAM30_RULE3	2FCh

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 322. RAM30_RULE0

Rule	Address Access
0	0x2048 0000 – 0x2048 1FFF
1	0x2048 2000 – 0x2048 3FFF
2	0x2048 4000 – 0x2048 5FFF
3	0x2048 6000 – 0x2048 7FFF
4	0x2048 8000 – 0x2048 9FFF
5	0x2048 A000 – 0x2048 BFFF
6	0x2048 C000 – 0x2048 DFFF
7	0x2048 E000 – 0x2048 FFFF

Table 323. RAM30_RULE1

Rule	Address Access
0	0x2049 0000 – 0x2049 1FFF
1	0x2049 2000 – 0x2049 3FFF
2	0x2049 4000 – 0x2049 5FFF
3	0x2049 6000 – 0x2049 7FFF
4	0x2049 8000 – 0x2049 9FFF
5	0x2049 A000 – 0x2049 BFFF
6	0x2049 C000 – 0x2049 DFFF
7	0x2049 E000 – 0x2049 FFFF

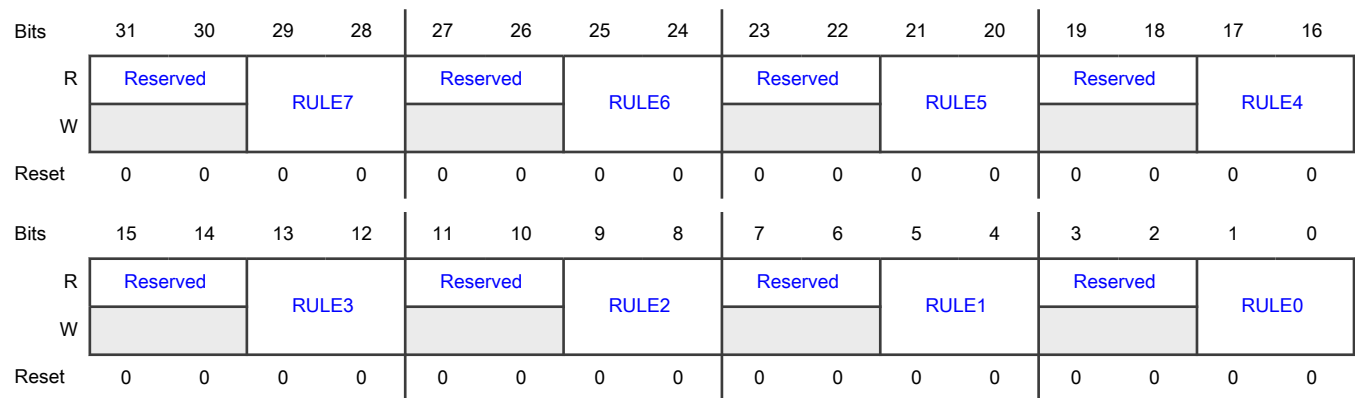
Table 324. RAM30_RULE2

Rule	Address Access
0	0x204A 0000 – 0x204A 1FFF
1	0x204A 2000 – 0x204A 3FFF
2	0x204A 4000 – 0x204A 5FFF
3	0x204A 6000 – 0x204A 7FFF
4	0x204A 8000 – 0x204A 9FFF
5	0x204A A000 – 0x204A BFFF
6	0x204A C000 – 0x204A DFFF
7	0x204A E000 – 0x204A FFFF

Table 325. RAM30_RULE3

Rule	Address Access
0	0x204B 0000 – 0x204B 1FFF
1	0x204B 2000 – 0x204B 3FFF
2	0x204B 4000 – 0x204B 5FFF
3	0x204B 6000 – 0x204B 7FFF
4	0x204B 8000 – 0x204B 9FFF
5	0x204B A000 – 0x204B BFFF
6	0x204B C000 – 0x204B DFFF
7	0x204B E000 – 0x204B FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.36 SRAM Partition 31 Rule(n) Register (RAM31_RULE0 - RAM31_RULE3)

Offset

Register	Offset
RAM31_RULE0	300h
RAM31_RULE1	304h
RAM31_RULE2	308h
RAM31_RULE3	30Ch

Function

This register controls access to the RAM. Each rule field controls access to a specific range in RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 326. RAM31_RULE0

Rule	Address Access
0	0x204C 0000 – 0x204C 1FFF
1	0x204C 2000 – 0x204C 3FFF
2	0x204C 4000 – 0x204C 5FFF
3	0x204C 6000 – 0x204C 7FFF
4	0x204C 8000 – 0x204C 9FFF
5	0x204C A000 – 0x204C BFFF
6	0x204C C000 – 0x204C DFFF
7	0x204C E000 – 0x204C FFFF

Table 327. RAM31_RULE1

Rule	Address Access
0	0x204D 0000 – 0x204D 1FFF
1	0x204D 2000 – 0x204D 3FFF
2	0x204D 4000 – 0x204D 5FFF
3	0x204D 6000 – 0x204D 7FFF
4	0x204D 8000 – 0x204D 9FFF
5	0x204D A000 – 0x204D BFFF
6	0x204D C000 – 0x204D DFFF
7	0x204D E000 – 0x204D FFFF

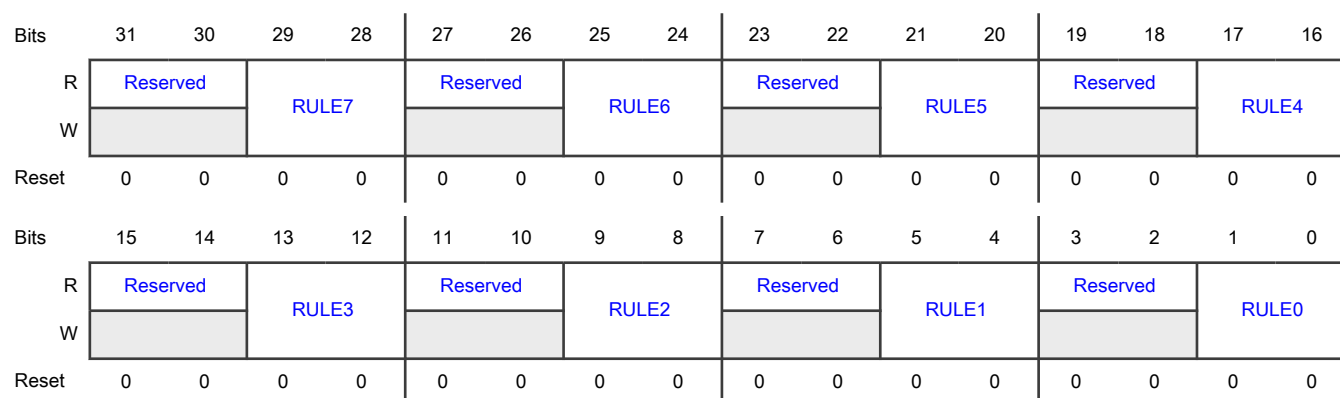
Table 328. RAM31_RULE2

Rule	Address Access
0	0x204E 0000 – 0x204E 1FFF
1	0x204E 2000 – 0x204E 3FFF
2	0x204E 4000 – 0x204E 5FFF
3	0x204E 6000 – 0x204E 7FFF
4	0x204E 8000 – 0x204E 9FFF
5	0x204E A000 – 0x204E BFFF
6	0x204E C000 – 0x204E DFFF
7	0x204E E000 – 0x204E FFFF

Table 329. RAM31_RULE3

Rule	Address Access
0	0x204F 0000 – 0x204F 1FFF
1	0x204F 2000 – 0x204F 3FFF
2	0x204F 4000 – 0x204F 5FFF
3	0x204F 6000 – 0x204F 7FFF
4	0x204F 8000 – 0x204F 9FFF
5	0x204F A000 – 0x204F BFFF
6	0x204F C000 – 0x204F DFFF
7	0x204F E000 – 0x204F FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.37 Smart DMA (SDMA) RAM Rule(n) Register (SDMA_RAM_RULE0 - SDMA_RAM_RULE3)

Offset

Register	Offset
SDMA_RAM_RULE0	320h
SDMA_RAM_RULE1	324h
SDMA_RAM_RULE2	328h
SDMA_RAM_RULE3	32Ch

Function

This register controls access to the SDMA RAM. Each rule field controls access to a specific range in SDMA RAM.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 330. SDMA_RAM_RULE0

Rule	Address Access
0	0x2410 0000 – 0x2410 03FF
1	0x2410 0400 – 0x2410 07FF
2	0x2410 0800 – 0x2410 0BFF
3	0x2410 0C00 – 0x2410 0FFF
4	0x2410 1000 – 0x2410 13FF
5	0x2410 1400 – 0x2410 17FF
6	0x2410 1800 – 0x2410 1BFF
7	0x2410 1C00 – 0x2410 1FFF

Table 331. SDMA_RAM_RULE1

Rule	Address Access
0	0x2410 2000 – 0x2410 23FF
1	0x2410 2400 – 0x2410 27FF
2	0x2410 2800 – 0x2410 2BFF
3	0x2410 2C00 – 0x2410 2FFF
4	0x2410 3000 – 0x2410 33FF
5	0x2410 3400 – 0x2410 37FF
6	0x2410 3800 – 0x2410 3BFF
7	0x2410 3C00 – 0x2410 3FFF

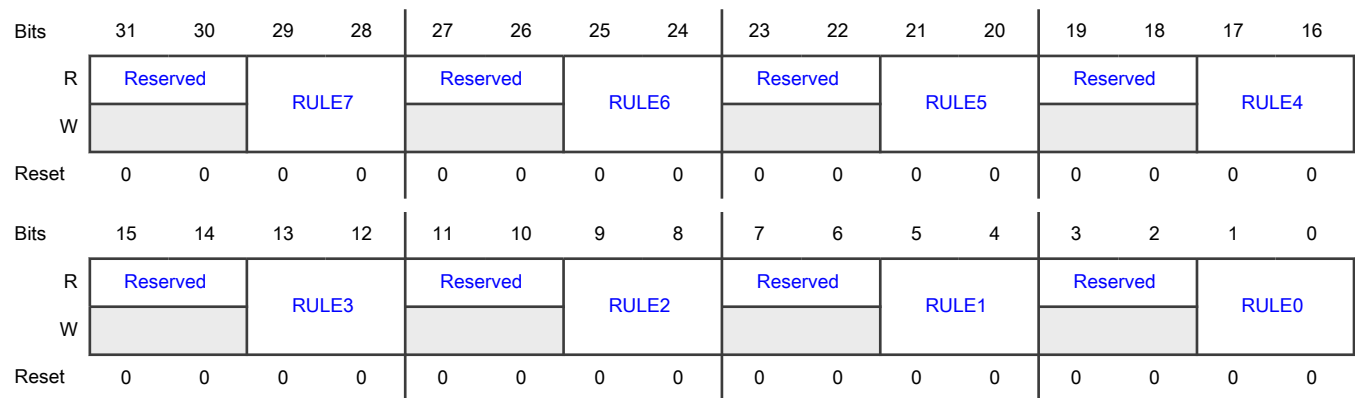
Table 332. SDMA_RAM_RULE2

Rule	Address Access
0	0x2410 4000 – 0x2410 43FF
1	0x2410 4400 – 0x2410 47FF
2	0x2410 4800 – 0x2410 4BFF
3	0x2410 4C00 – 0x2410 4FFF
4	0x2410 5000 – 0x2410 53FF
5	0x2410 5400 – 0x2410 57FF
6	0x2410 5800 – 0x2410 5BFF
7	0x2410 5C00 – 0x2410 5FFF

Table 333. SDMA_RAM_RULE3

Rule	Address Access
0	0x2410 6000 – 0x2410 63FF
1	0x2410 6400 – 0x2410 67FF
2	0x2410 6800 – 0x2410 6BFF
3	0x2410 6C00 – 0x2410 6FFF
4	0x2410 7000 – 0x2410 73FF
5	0x2410 7400 – 0x2410 77FF
6	0x2410 7800 – 0x2410 7BFF
7	0x2410 7C00 – 0x2410 7FFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 RULE7	Rule 7 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	Rule 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
RULE6	Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 RULE2	Rule 2 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the SDMA RAM at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.38 FlexSPI1 Region 0 Rule(n) Register (FLEXSPI1_REGION0_RULE0 - FLEXSPI1_REGION0_RULE3)

Offset

Register	Offset
FLEXSPI1_REGION0_RULE0	340h
FLEXSPI1_REGION0_RULE1	344h
FLEXSPI1_REGION0_RULE2	348h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
FLEXSPI1_REGION0_RULE3	34Ch

Function

These 4 registers control access to the FLEXSPI1 External Memory interface. For each register, each rule field controls access to a specific range in FLEXSPI1 External Memory interface.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

FLEXSPI1_REGION0_RULE0	0x2800 0000 – 0x281F FFFF
FLEXSPI1_REGION0_RULE1	0x2820 0000 – 0x283F FFFF
FLEXSPI1_REGION0_RULE2	0x2840 0000 – 0x285F FFFF
FLEXSPI1_REGION0_RULE3	0x2860 0000 – 0x287F FFFF

Table 334. FLEXSPI1_REGION0_RULE0

Rule	Address Access
0	0x2800 0000 – 0x2803 FFFF
1	0x2804 0000 – 0x2807 FFFF
2	0x2808 0000 – 0x280B FFFF
3	0x280C 0000 – 0x280F FFFF
4	0x2810 0000 – 0x2813 FFFF
5	0x2814 0000 – 0x2817 FFFF
6	0x2818 0000 – 0x281B FFFF
7	0x281C 0000 – 0x281F FFFF

Table 335. FLEXSPI1_REGION0_RULE1

Rule	Address Access
0	0x2820 0000 – 0x2823 FFFF
1	0x2824 0000 – 0x2827 FFFF
2	0x2828 0000 – 0x282B FFFF
3	0x282C 0000 – 0x282F FFFF
4	0x2830 0000 – 0x2833 FFFF
5	0x2834 0000 – 0x2837 FFFF
6	0x2838 0000 – 0x283B FFFF
7	0x283C 0000 – 0x283F FFFF

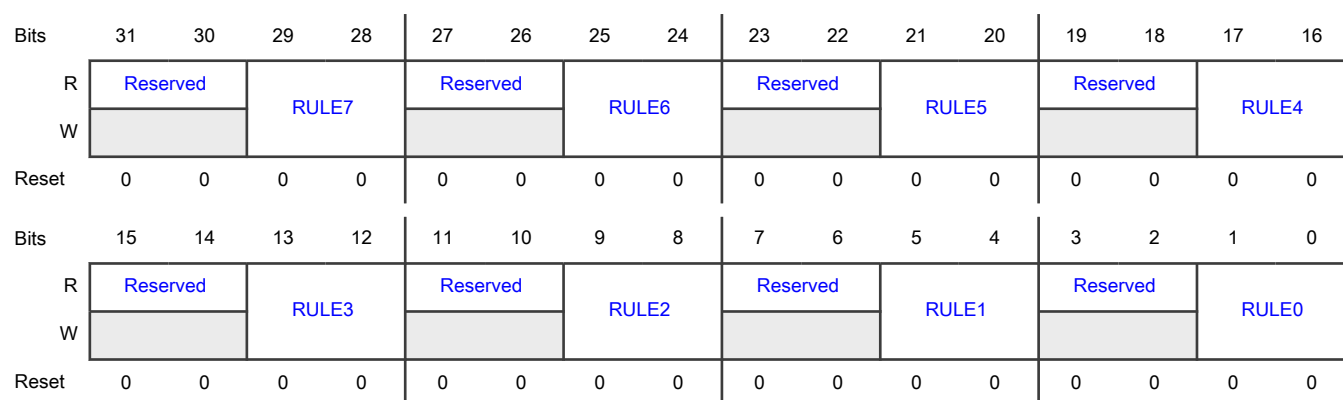
Table 336. FLEXSPI1_REGION0_RULE2

Rule	Address Access
0	0x2840 0000 – 0x2843 FFFF
1	0x2844 0000 – 0x2847 FFFF
2	0x2848 0000 – 0x284B FFFF
3	0x284C 0000 – 0x284F FFFF
4	0x2850 0000 – 0x2853 FFFF
5	0x2854 0000 – 0x2857 FFFF
6	0x2858 0000 – 0x285B FFFF
7	0x285C 0000 – 0x285F FFFF

Table 337. FLEXSPI1_REGION0_RULE3

Rule	Address Access
0	0x2860 0000 – 0x2863 FFFF
1	0x2864 0000 – 0x2867 FFFF
2	0x2868 0000 – 0x286B FFFF
3	0x286C 0000 – 0x286F FFFF
4	0x2870 0000 – 0x2873 FFFF
5	0x2874 0000 – 0x2877 FFFF
6	0x2878 0000 – 0x287B FFFF
7	0x287C 0000 – 0x287F FFFF

Diagram



Fields

Field	Function
31-30	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29-28 RULE7	Rule 7 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 RULE6	Rule 6 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 RULE5	Rule 5 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 RULE4	Rule 4 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 RULE0	Rule 0 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.39 FlexSPI1 Region a Rule 0 Register (FLEXSPI1_REGION1_RULE0 - FLEXSPI1_REGION4_RULE0)

Offset

Register	Offset
FLEXSPI1_REGION1_RULE0	350h
FLEXSPI1_REGION2_RULE0	360h
FLEXSPI1_REGION3_RULE0	370h
FLEXSPI1_REGION4_RULE0	380h

Function

These 4 registers control access to the FLEXSPI1 External Memory interface. For each register, each rule field controls access to a specific range in FLEXSPI1 External Memory interface.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

FLEXSPI1_REGION1_RULE0	0x2880 0000 – 0x28FF FFFF
FLEXSPI1_REGION2_RULE0	0x2900 0000 – 0x29FF FFFF
FLEXSPI1_REGION3_RULE0	0x2A00 0000 – 0x2BFF FFFF
FLEXSPI1_REGION4_RULE0	0x2C00 0000 – 0x2FFF FFFF

Table 338. FLEXSPI1_REGION1_RULE0

Rule	Address Access
0	0x2880 0000 – 0x289F FFFF

Table continues on the next page...

Table 338. FLEXSPI1_REGION1_RULE0 (continued)

Rule	Address Access
1	0x28A0 0000 – 0x28BF FFFF
2	0x28C0 0000 – 0x28DF FFFF
3	0x28E0 0000 – 0x28FF FFFF

Table 339. FLEXSPI1_REGION2_RULE0

Rule	Address Access
0	0x2900 0000 – 0x293F FFFF
1	0x2940 0000 – 0x297F FFFF
2	0x2980 0000 – 0x29BF FFFF
3	0x29C0 0000 – 0x29FF FFFF

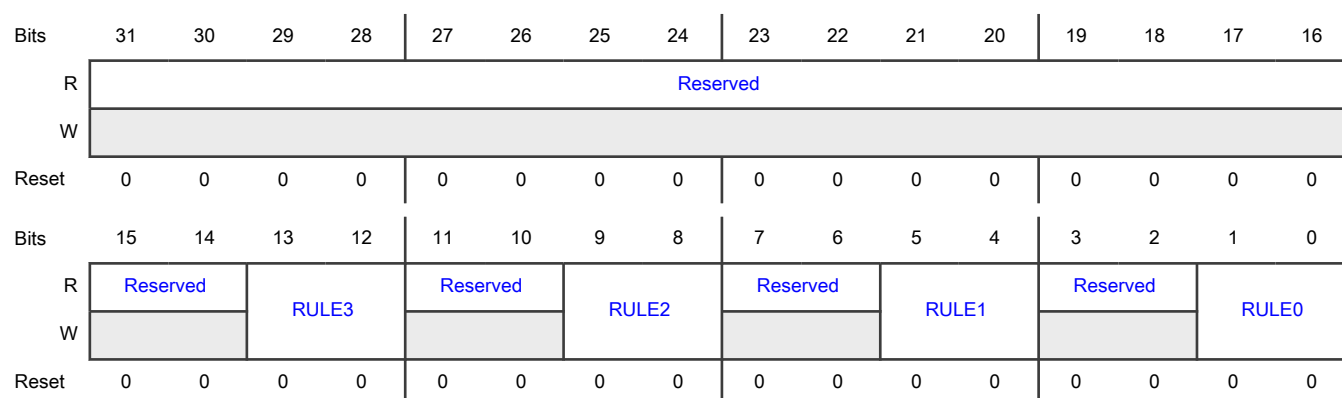
Table 340. FLEXSPI1_REGION3_RULE0

Rule	Address Access
0	0x2A00 0000 – 0x2A7F FFFF
1	0x2A80 0000 – 0x2AFF FFFF
2	0x2B00 0000 – 0x2B7F FFFF
3	0x2B80 0000 – 0x2BFF FFFF

Table 341. FLEXSPI1_REGION4_RULE0

Rule	Address Access
0	0x2C00 0000 – 0x2CFF FFFF
1	0x2D00 0000 – 0x2DFF FFFF
2	0x2E00 0000 – 0x2EFF FFFF
3	0x2F00 0000 – 0x2FFF FFFF

Diagram



Fields

Field	Function
31-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the FLEXSPI1 External Memory interface at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed
	01b - Non-secure and privilege access allowed
	10b - Secure and non-privilege user access allowed
	11b - Secure and privilege user access allowed

22.4.1.40 APB Bridge Peripheral 0 Rule 0 Register (APB_BRIDGE_PER0_RULE0)

Offset

Register	Offset
APB_BRIDGE_PER0_RULE0	3A0h

Function

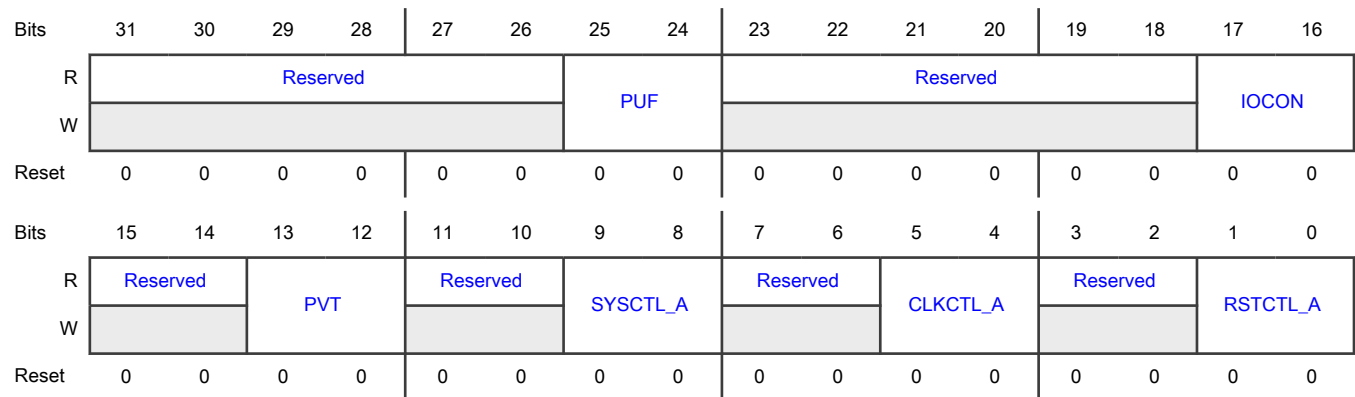
This register controls access to the APB Bridge Peripheral 0. Ranges 0x4000 5000 – 0x4000 5FFF and 0x4000 7000 – 0x4000 7FFF are not used. Each rule field controls access to a specific range in APB Bridge Peripheral 0.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 342. APB_BRIDGE_PER0_RULE0

Rule	Address Access
RSTCTL_A	0x4000 0000 – 0x4000 0FFF
CLKCTL_A	0x4000 1000 – 0x4000 1FFF
SYSCTL_A	0x4000 2000 – 0x4000 2FFF
PVT	0x4000 3000 – 0x4000 3FFF
IOCON	0x4000 4000 – 0x4000 4FFF
PUF	0x4000 6000 – 0x4000 6FFF

Diagram**Fields**

Field	Function
31-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 PUF	PUF Defines the minimal security level to access PUF. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 IOCON	IOCON Defines the minimal security level to access IOCON. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 PVT	PVT Defines the minimal security level to access PVT. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 SYSCTL_A	SYSCTL_A Defines the minimal security level to access SYSCTL_A. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 CLKCTL_A	CLKCTL_A Defines the minimal security level to access the CLKCTL_A. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RSTCTL_A	RSTCTL_A Defines the minimal security level to access RSTCTL_A. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.41 APB Bridge Peripheral 0 Rule 1 Register (APB_BRIDGE_PER0_RULE1)

Offset

Register	Offset
APB_BRIDGE_PER0_RULE1	3A4h

Function

This register controls access to the APB Bridge Peripheral 0. Range 0x4000 8000 - 0x4000 DFFF is not used. Each rule field controls access to a specific range in APB Bridge Peripheral 0.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 343. APB_BRIDGE_PER0_RULE1

Rule	Address Access
WWDT0	0x4000 E000 – 0x4000 EFFF
MICRO_TICK	0x4000 F000 – 0x4000 FFFF

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		MICRO_TICK		Reserved		WWDT0		Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 MICRO_TICK	MICRO_TICK Defines the minimal security level to access the MICRO_TICK. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 WWDT0	WWDT0 Defines the minimal security level to access WWDT0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.42 APB Bridge Peripheral 0 Rule 3 Register (APB_BRIDGE_PER0_RULE3)

Offset

Register	Offset
APB_BRIDGE_PER0_RULE3	3ACh

Function

This register controls access to the APB Bridge Peripheral 0. Range 0x4001 0000 - 0x4001 DFFF is not used. Each rule field controls access to a specific range in APB Bridge Peripheral 0.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 344. APB_BRIDGE_PER0_RULE3

Rule	Address Access
PROBE_IS_SYNC	0x4001 E000 – 0x4001 EFFF
PROBE_IS_XVC	0x4001 F000 – 0x4001 FFFF

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		PROBE_IS_XV		Reserved		PROBE_IS_SY		Reserved							
W			C				NC									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 PROBE_IS_XV C	PROBE_IS (XVC) Defines the minimal security level to access PROBE_IS (XVC). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 PROBE_IS_SY NC	PROBE_IS (SYNC) Defines the minimal security level to access PROBE_IS (SYNC). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.43 APB Bridge Peripheral 1 Rule 0 Register (APB_BRIDGE_PER1_RULE0)

Offset

Register	Offset
APB_BRIDGE_PER1_RULE0	3B0h

Function

This register controls access to the APB Bridge Peripheral 1. Range 0x4002 3000 - 0x4002 4FFF is not used. Each rule field controls access to a specific range in APB Bridge Peripheral 1.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 345. APB_BRIDGE_PER1_RULE0

Rule	Address Access
RSTCTL_B	0x4002 0000 – 0x4002 0FFF
CLKCTL_B	0x4002 1000 – 0x4002 1FFF
SYSCTL_B	0x4002 2000 – 0x4002 2FFF
GPIO_INT	0x4002 5000 – 0x4002 5FFF
PERIPHERAL_MUXES	0x4002 6000 – 0x4002 6FFF
SDMA	0x4002 7000 – 0x4002 7FFF

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		SDMA		Reserved		PERIPHERAL_MUXES		Reserved		GPIO_INT		Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved						SYSCTL_B		Reserved		CLKCTL_B		Reserved		RSTCTL_B	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30	Reserved; the value read from a reserved bit is not defined.
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
29-28 SDMA	Smart DMA (SDMA) Defines the minimal security level to access SDMA. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 PERIPHERAL_MUXES	Peripheral Muxes Defines the minimal security level to access PERIPHERAL_MUXES. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 GPIO_INT	GPIO_INT Defines the minimal security level to access GPIO_INT. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 SYSCTL_B	SYSCTL_B Defines the minimal security level to access SYSCTL_B. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5-4 CLKCTL_B	CLKCTL_B Defines the minimal security level to access CLKCTL_B. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RSTCTL_B	RSTCTL_B Defines the minimal security level to access RSTCTL_B. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.44 APB Bridge Peripheral 1 Rule 1 Register (APB_BRIDGE_PER1_RULE1)

Offset

Register	Offset
APB_BRIDGE_PER1_RULE1	3B4h

Function

This register controls access to the APB Bridge Peripheral 1. Each rule field controls access to a specific range in APB Bridge Peripheral 1.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 346. APB_BRIDGE_PER1_RULE1

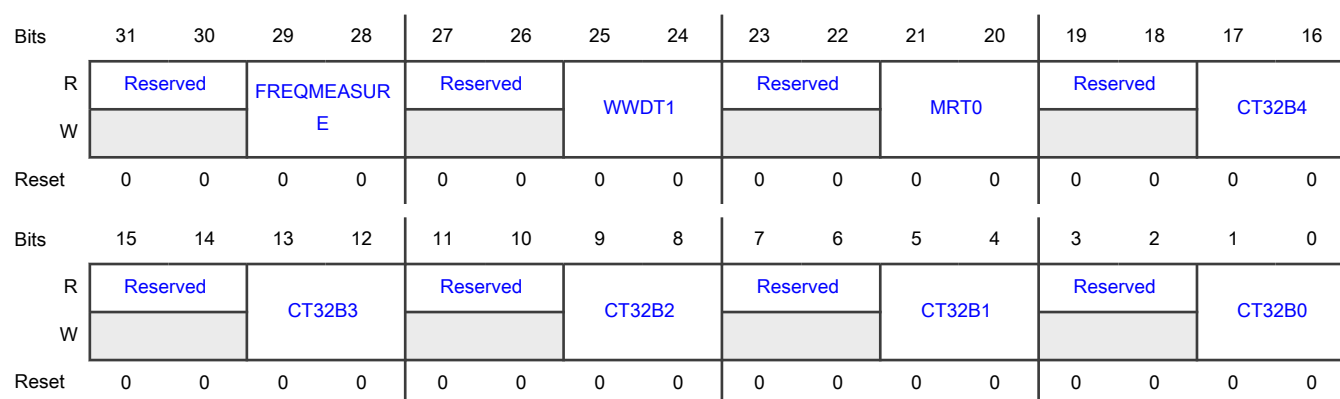
Rule	Address Access
CT32B0	0x4002 8000 – 0x4002 8FFF
CT32B1	0x4002 9000 – 0x4002 9FFF

Table continues on the next page...

Table 346. APB_BRIDGE_PER1_RULE1 (continued)

Rule	Address Access
CT32B2	0x4002 A000 – 0x4002 AFFF
CT32B3	0x4002 B000 – 0x4002 BFFF
CT32B4	0x4002 C000 – 0x4002 CFFF
MRT0	0x4002 D000 – 0x4002 DFFF
WWDT1	0x4002 E000 – 0x4002 EFFF
FREQMEASURE	0x4002 F000 – 0x4002 FFFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 FREQMEASURE	Defines the minimal security level to access FREQMEASURE. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 WWDT1	WWDT1 Defines the minimal security level to access WWDT1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 MRT0	MRT0 Defines the minimal security level to access MRT0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 CT32B4	CT32B4 Defines the minimal security level to access CT32B4. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 CT32B3	CT32B3 Defines the minimal security level to access CT32B3. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8	CT32B2

Table continues on the next page...

Table continued from the previous page...

Field	Function
CT32B2	Defines the minimal security level to access CT32B2. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 CT32B1	CT32B1 Defines the minimal security level to access CT32B1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 CT32B0	CT32B0 Defines the minimal security level to access CT32B0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.45 APB Bridge Peripheral 1 Rule 2 Register (APB_BRIDGE_PER1_RULE2)

Offset

Register	Offset
APB_BRIDGE_PER1_RULE2	3B8h

Function

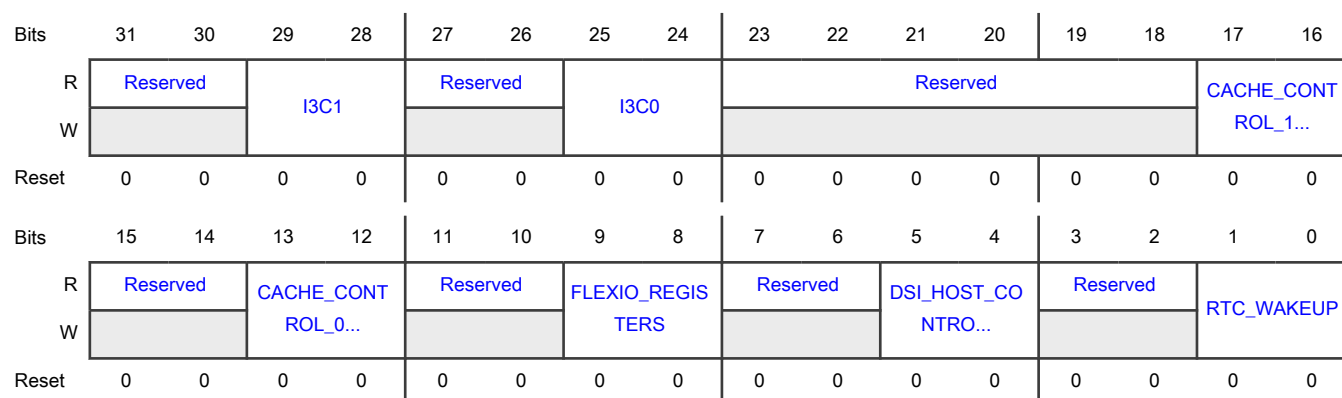
This register controls access to the APB Bridge Peripheral 1. Range 0x4003 5000 – 0x4003 5FFF is not used. Each rule field controls access to a specific range in APB Bridge Peripheral 1.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 347. APB_BRIDGE_PER1_RULE2

Rule	Address Access
RTC_WAKEUP	0x4003 0000 – 0x4003 0FFF
DSI_HOST_CONTROLLER	0x4003 1000 – 0x4003 1FFF
FLEXIO_REGISTERS	0x4003 2000 – 0x4003 2FFF
CACHE_CONTROL_0_REGS	0x4003 3000 – 0x4003 3FFF
CACHE_CONTROL_1_REGS	0x4003 4000 – 0x4003 4FFF
I3C0	0x4003 6000 – 0x4003 6FFF
I3C1	0x4003 7000 – 0x4003 7FFF

Diagram**Fields**

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 I3C1	I3C1 Defines the minimal security level to access I3C1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 I3C0	I3C0 Defines the minimal security level to access I3C0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 CACHE_CONT ROL_1_REGS	Cache Control 1 Registers Defines the minimal security level to access the Cache Control 1 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 CACHE_CONT ROL_0_REGS	Cache Control 0 Registers Defines the minimal security level to access the Cache Control 0 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 FLEXIO_REGIS TERS	FLEXIO Registers Defines the minimal security level to access the FLEXIO registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4	DSI Host Controller

Table continues on the next page...

Table continued from the previous page...

Field	Function
DSI_HOST_CONTROLLER	Defines the minimal security level to access the DSI Host Controller. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RTC_WAKEUP	RTC Wakeup Defines the minimal security level to access RTC Wakeup. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.46 APB Bridge Peripheral 1 Rule 3 Register (APB_BRIDGE_PER1_RULE3)

Offset

Register	Offset
APB_BRIDGE_PER1_RULE3	3BCh

Function

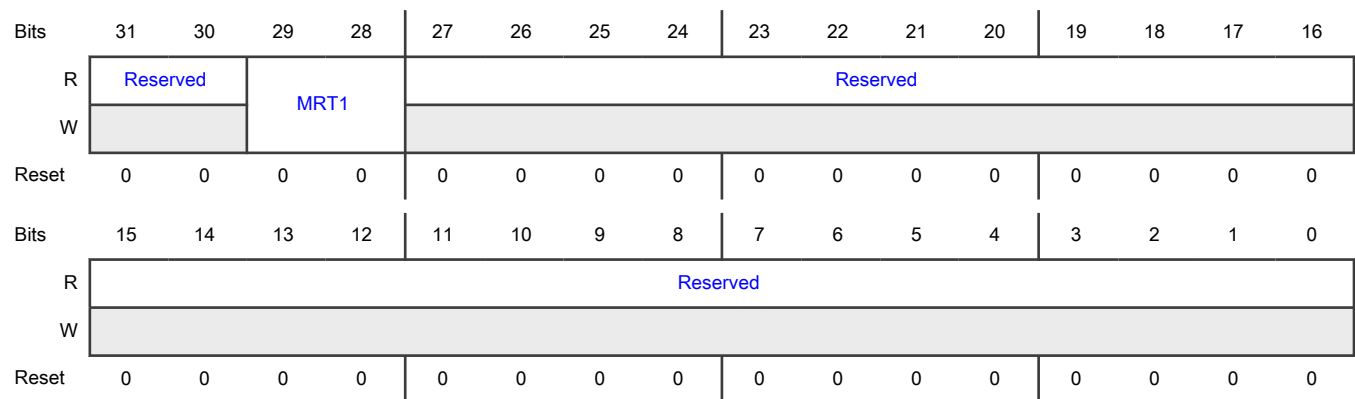
This register controls access to the APB Bridge Peripheral 1. Range 0x4003 8000 - 0x4003 EFFF is not used. Each rule field controls access to a specific range in APB Bridge Peripheral 1.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 348. APB_BRIDGE_PER1_RULE3

Rule	Address Access
MRT1	0x4003 F000 – 0x4003 FFFF

Diagram**Fields**

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 MRT1	MRT1 Defines the minimal security level to access MRT1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.47 AHB Peripheral 0 Slave Rule 0 Register (AHB_PERIPH0_SLAVE_RULE0)**Offset**

Register	Offset
AHB_PERIPH0_SLAVE_RULE0	3C0h

Function

This register controls access to the AHB Peripheral 0 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 0.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 349. AHB_PERIPH0_SLAVE_RULE0

Rule	Address Access
HSGPIO	0x4010 0000 – 0x4010 3FFF
DMA0	0x4010 4000 – 0x4010 4FFF
DMA1	0x4010 5000 – 0x4010 5FFF
FLEXCOMM0	0x4010 6000 – 0x4010 6FFF
FLEXCOMM1	0x4010 7000 – 0x4010 7FFF
FLEXCOMM2	0x4010 8000 – 0x4010 8FFF
FLEXCOMM3	0x4010 9000 – 0x4010 9FFF
DEBUG_MAILBOX	0x4010 F000 – 0x4010 FFFF

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		DEBUG_MAILBOX		Reserved		FLEXCOMM3		Reserved		FLEXCOMM2		Reserved		FLEXCOMM1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		FLEXCOMM0		Reserved		DMA1		Reserved		DMA0		Reserved		HSGPIO	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 DEBUG_MAILBOX	Defines the minimal security level to access DEBUG_MAILBOX. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24	FLEXCOMM 3

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM3	Defines the minimal security level to access FLEXCOMM3. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 FLEXCOMM2	FLEXCOMM 2 Defines the minimal security level to access FLEXCOMM2. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 FLEXCOMM1	FLEXCOMM 1 Defines the minimal security level to access FLEXCOMM1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 FLEXCOMM0	FLEXCOMM 0 Defines the minimal security level to access FLEXCOMM0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-8 DMA1	DMA 1 Defines the minimal security level to access DMA1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 DMA0	DMA 0 Defines the minimal security level to access DMA0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 HSGPIO	HSGPIO Defines the minimal security level to access HSGPIO. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.48 AIPS Bridge Peripheral 0 Rule 0 Register (AIPS_BRIDGE0_PER_RULE0)

Offset

Register	Offset
AIPS_BRIDGE0_PER_RULE0	3D0h

Function

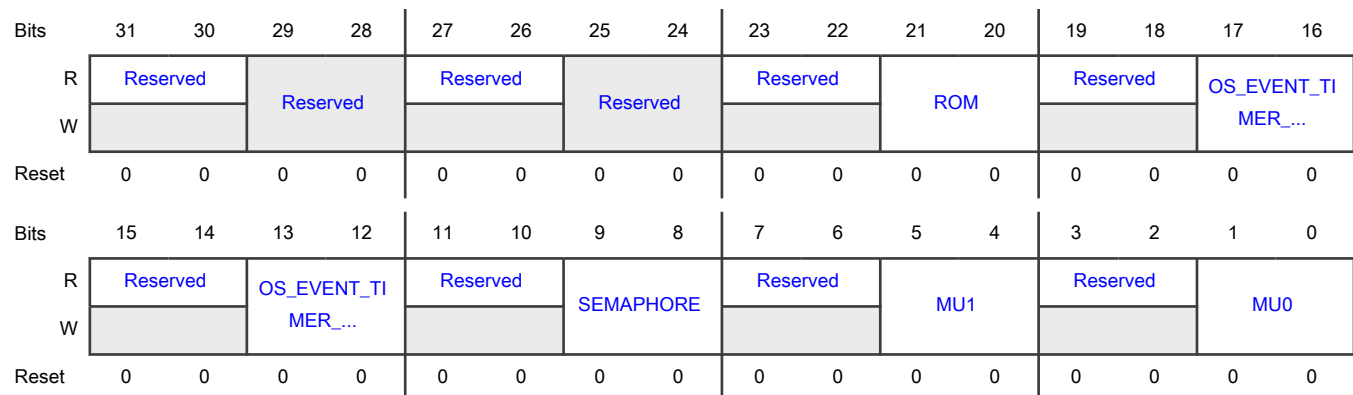
This register controls access to the AIPS Bridge Peripheral 0. Each rule field controls access to a specific range in AIPS Bridge Peripheral 0.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 350. AIPS_BRIDGE0_PER_RULE0

Rule	Address Access
MU0	0x4011 0000 – 0x4011 0FFF
MU1	0x4011 1000 – 0x4011 1FFF
SEMAPHORE	0x4011 2000 – 0x4011 2FFF
OS_EVENT_TIMER_M33_PORT	0x4011 3000 – 0x4011 3FFF
OS_EVENT_TIMER_DSP_PORT	0x4011 4000 – 0x4011 4FFF
ROM	0x4011 5000 – 0x4011 5FFF

Diagram**Fields**

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 —	Reserved
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 —	Reserved
23-22 —	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-20 ROM	ROM Defines the minimal security level to access the ROM Controller. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 OS_EVENT_TIMER_DSP_PORT	OS_EVENT_TIMER (DSP PORT) Defines the minimal security level to access the OS_EVENT_TIMER (DSP PORT). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 OS_EVENT_TIMER_M33_PORT	OS_EVENT_TIMER (M33 PORT) Defines the minimal security level to access the OS_EVENT_TIMER (M33 PORT). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 SEMAPHORE	Semaphore Defines the minimal security level to access Semaphore. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5-4 MU1	MU1 (DSP PORT) Defines the minimal security level to access MU1 (DSP PORT). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 MU0	MU0 (M33 PORT) Defines the minimal security level to access MU0 (M33 PORT). 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.49 AHB Peripheral 1 Slave Rule 0 Register (AHB_PERIPH1_SLAVE_RULE0)

Offset

Register	Offset
AHB_PERIPH1_SLAVE_RULE0	3E0h

Function

This register controls access to the AHB Peripheral 1 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 1 Slave.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 351. AHB_PERIPH1_SLAVE_RULE0

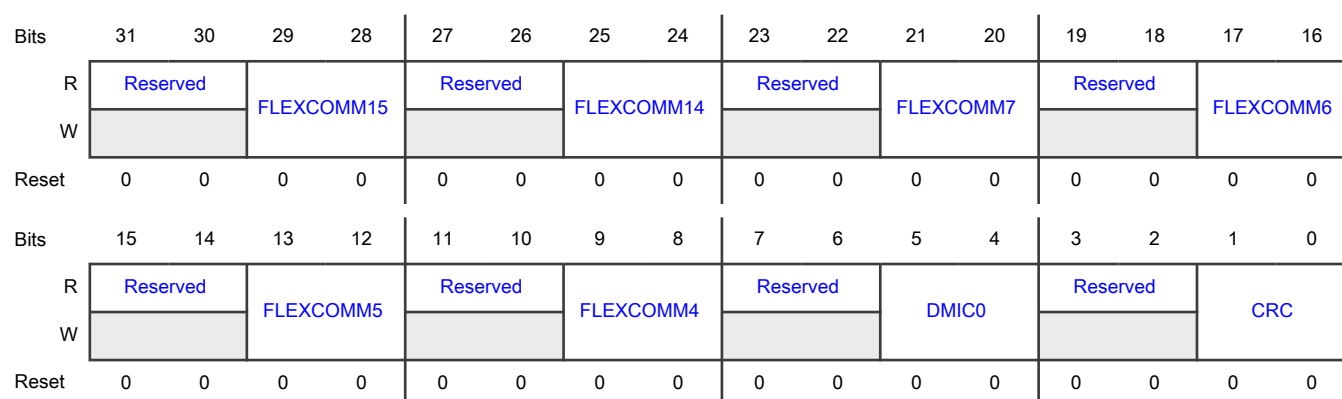
Rule	Address Access
CRC	0x4012 0000 – 0x4012 0FFF
DMIC0	0x4012 1000 – 0x4012 1FFF

Table continues on the next page...

Table 351. AHB_PERIPH1_SLAVE_RULE0 (continued)

Rule	Address Access
FLEXCOMM4	0x4012 2000 – 0x4012 2FFF
FLEXCOMM5	0x4012 3000 – 0x4012 3FFF
FLEXCOMM6	0x4012 4000 – 0x4012 4FFF
FLEXCOMM7	0x4012 5000 – 0x4012 5FFF
FLEXCOMM14	0x4012 6000 – 0x4012 6FFF
FLEXCOMM15	0x4012 7000 – 0x4012 7FFF

Diagram



Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 FLEXCOMM15	FLEXCOMM 15 Defines the minimal security level to access FLEXCOMM15. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 FLEXCOMM14	FLEXCOMM 14 Defines the minimal security level to access FLEXCOMM14.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 FLEXCOMM7	FLEXCOMM 7 Defines the minimal security level to access FLEXCOMM7. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 FLEXCOMM6	FLEXCOMM 6 Defines the minimal security level to access FLEXCOMM6. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 FLEXCOMM5	FLEXCOMM 5 Defines the minimal security level to access FLEXCOMM5. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8	FLEXCOMM 4

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLEXCOMM4	Defines the minimal security level to access FLEXCOMM4. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 DMIC0	DMIC0 Defines the minimal security level to access DMIC0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 CRC	CRC Defines the minimal security level to access CRC. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.50 AHB Peripheral 1 Slave Rule 1 Register (AHB_PERIPH1_SLAVE_RULE1)

Offset

Register	Offset
AHB_PERIPH1_SLAVE_RULE1	3E4h

Function

This register controls access to the AHB Peripheral 1 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 1 Slave.

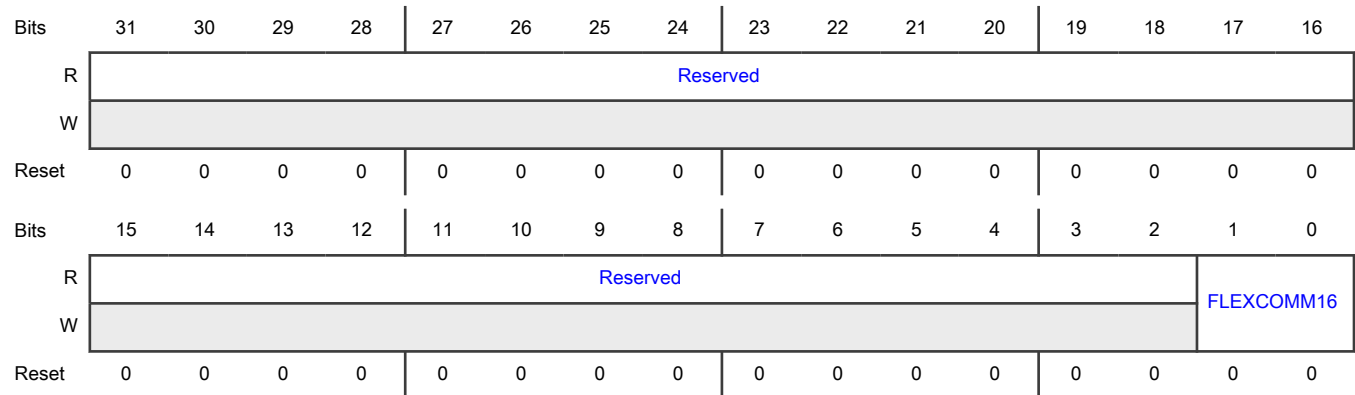
NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 352. AHB_PERIPH1_SLAVE_RULE1

Rule	Address Access
FLEXCOMM16	0x4012 8000 – 0x4012 8FFF

Diagram



Fields

Field	Function
31-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 FLEXCOMM16	FLEXCOMM 16 Defines the minimal security level to access FLEXCOMM16. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.51 AIPS Bridge Peripheral 1 Rule 0 Register (AIPS_BRIDGE1_PER_RULE0)

Offset

Register	Offset
AIPS_BRIDGE1_PER_RULE0	400h

Function

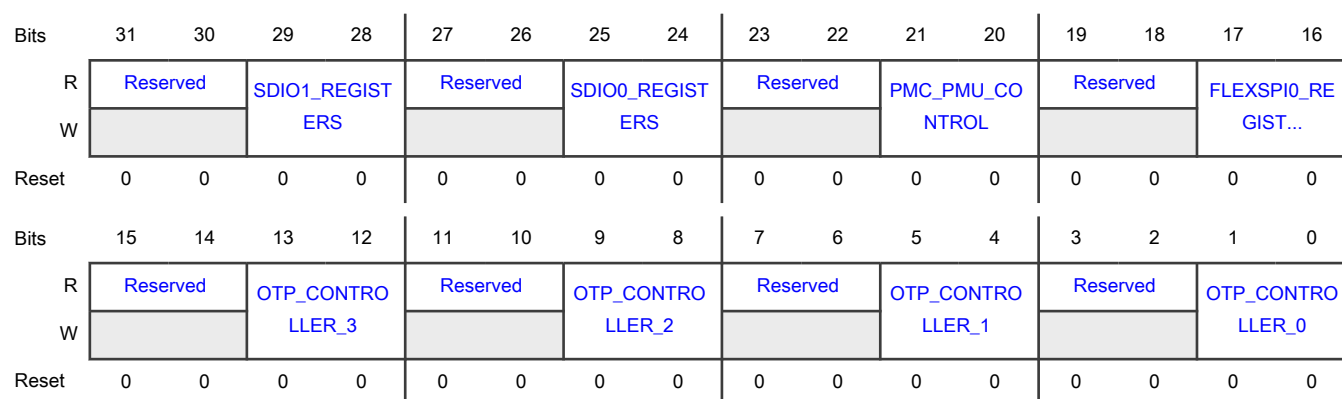
This register controls access to the AIPS Bridge Peripheral 1. Each rule field controls access to a specific range in the AIPS Bridge Peripheral 1.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 353. AIPS_BRIDGE1_PER_RULE0

Rule	Address Access
OTP_CONTROLLER_0	0x4013 0000 – 0x4013 FFFF
OTP_CONTROLLER_1	0x4013 1000 – 0x4013 1FFF
OTP_CONTROLLER_2	0x4013 2000 – 0x4013 2FFF
OTP_CONTROLLER_3	0x4013 3000 – 0x4013 3FFF
FLEXSPI0_REGISTERS	0x4013 4000 – 0x4013 4FFF
PMC_PMU_CONTROL	0x4013 5000 – 0x4013 5FFF
SDIO0_REGISTERS	0x4013 6000 – 0x4013 6FFF
SDIO1_REGISTERS	0x4013 7000 – 0x4013 7FFF

Diagram**Fields**

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 SDIO1_REGISTERS	SDIO 1 Registers Defines the minimal security level to access SDIO 1 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-24 SDIO0_REGISTERS	SDIO 0 Registers Defines the minimal security level to access SDIO 0 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 PMC_PMU_CONTROL	PMC (PMU CONTROL) Defines the minimal security level to access PMC. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 FLEXSPI0_REGISTERS	FLEXSPI0 Registers Defines the minimal security level to access FLEXSPI0 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 OTP_CONTROLLER_3	OTP Controller 3 Defines the minimal security level to access OTP Controller 3. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 OTP_CONTROLLER_2	OTP Controller 2 Defines the minimal security level to access OTP Controller 2. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 OTP_CONTROLLER_1	OTP Controller 1 Defines the minimal security level to access OTP Controller 1. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 OTP_CONTROLLER_0	OTP Controller 0 Defines the minimal security level to access OTP Controller 0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.52 AIPS Bridge Peripheral 1 Rule 1 Register (AIPS_BRIDGE1_PER_RULE1)

Offset

Register	Offset
AIPS_BRIDGE1_PER_RULE1	404h

Function

This register controls access to the AIPS Bridge Peripheral 1; the address range from 0x4013 D000 to 0x4013 FFFF is not used. Each rule field controls access to a specific range in the AIPS Bridge Peripheral 1.

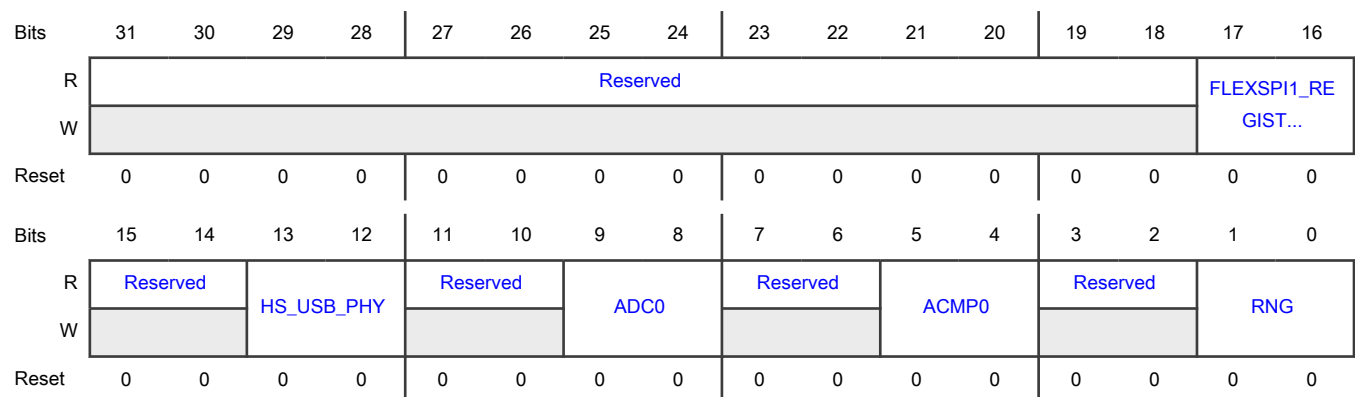
NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 354. AIPS_BRIDGE1_PER_RULE1

Rule	Address Access
RNG	0x4013 8000 – 0x4013 8FFF
ACMP0	0x4013 9000 – 0x4013 9FFF
ADC0	0x4013 A000 – 0x4013 AFFF
HS_USB_PHY	0x4013 B000 – 0x4013 BFFF
FLEXSPI1_REGISTERS	0x4013 C000 – 0x4013 CFFF

Diagram



Fields

Field	Function
31-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 FLEXSPI1_REGISTERS	FLEXSPI1 Registers Defines the minimal security level to access the FLEXSPI1 registers. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14	Reserved; the value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
13-12 HS_USB_PHY	HS USB PHY Defines the minimal security level to access the HS USB PHY. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 ADC0	ADC 0 Defines the minimal security level to access ADC 0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 ACMP0	ACMP 0 Defines the minimal security level to access ACMP 0. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RNG	RNG (Random Number Generator) Defines the minimal security level to access RNG. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.53 AHB Peripheral 2 Slave Rule 0 Register (AHB_PERIPH2_SLAVE_RULE0)

Offset

Register	Offset
AHB_PERIPH2_SLAVE_RULE0	410h

Function

This register controls access to the AHB Peripheral 2 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 2 Slave.

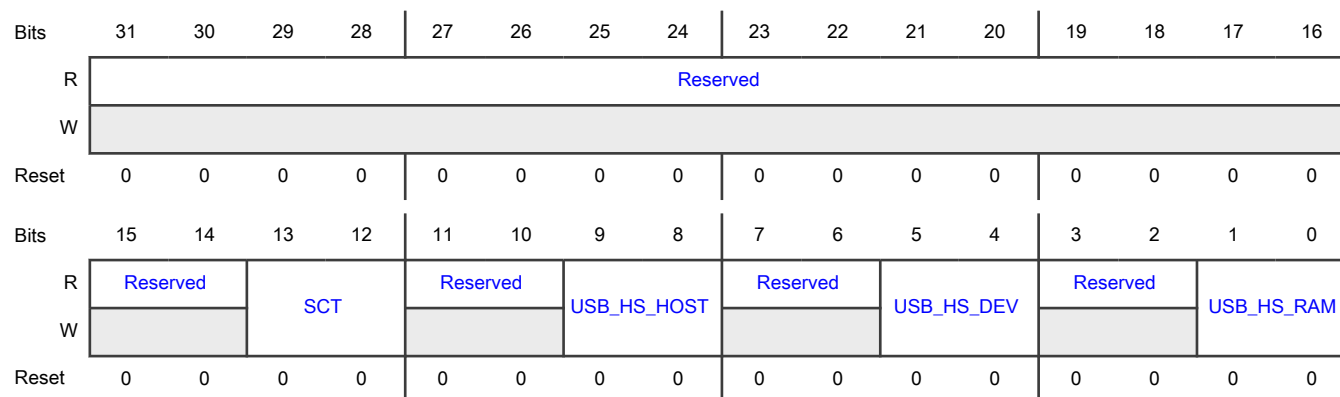
NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 355. AHB_PERIPH2_SLAVE_RULE0

Rule	Address Access
USB_HS_RAM	0x4014 0000 – 0x4014 3FFF
USB_HS_DEV	0x4014 4000 – 0x4014 4FFF
USB_HS_HOST	0x4014 5000 – 0x4014 5FFF
SCT	0x4014 6000 – 0x4014 6FFF

Diagram



Fields

Field	Function
31-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 SCT	SCT Defines the minimal security level to access SCT.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 USB_HS_HOST	USB HS HOST Defines the minimal security level to access the USB HS HOST. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 USB_HS_DEV	USB HS DEV Defines the minimal security level to access the USB HS DEV. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 USB_HS_RAM	USB HS RAM Defines the minimal security level to access the USB HS RAM interface. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.54 AHB Secure Control Peripheral Rule 0 Register (AHB_SECURE_CTRL_PERIPH_RULE0)

Offset

Register	Offset
AHB_SECURE_CTRL_PERIPH_RULE0	420h

Function

This register controls access to the AHB Secure Control Peripheral. Each rule field controls access to a specific range in the AHB Secure Control Peripheral.

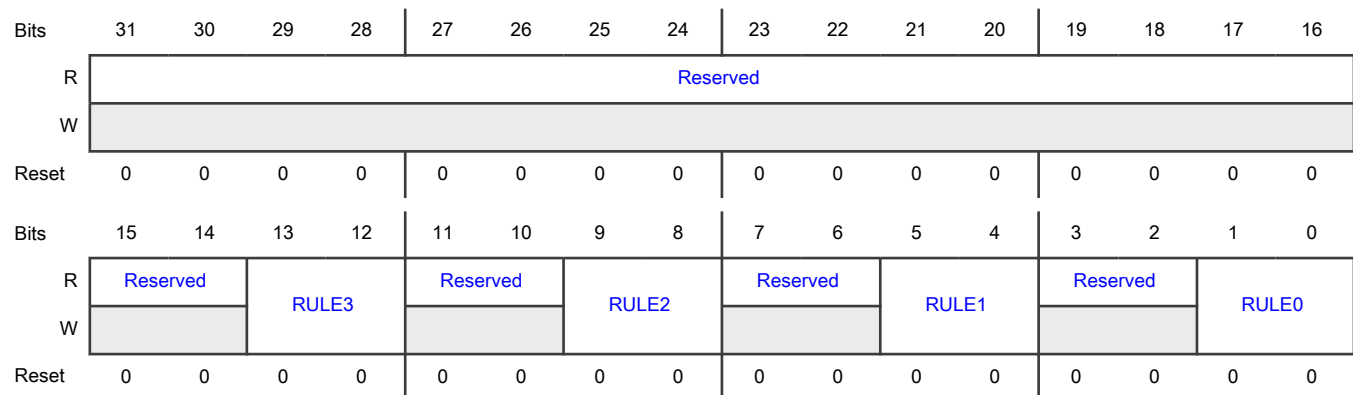
NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 356. AHB_SECURE_CTRL_PERIPH_RULE0

Rule	Address Access
RULE0	0x4014 8000 – 0x4014 8FFF
RULE1	0x4014 9000 – 0x4014 9FFF
RULE2	0x4014 A000 – 0x4014 AFFF
RULE3	0x4014 B000 – 0x4014 BFFF

Diagram



Fields

Field	Function
31-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 RULE3	Rule 3 Defines the minimal security level to access the memory at a specific address range.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 RULE2	Rule 2 Defines the minimal security level to access the memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 RULE1	Rule 1 Defines the minimal security level to access the memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 RULE0	Rule 0 Defines the minimal security level to access the memory at a specific address range. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.55 AHB Peripheral 3 Slave Rule 0 Register (AHB_PERIPH3_SLAVE_RULE0)

Offset

Register	Offset
AHB_PERIPH3_SLAVE_RULE0	430h

Function

This register controls access to the AHB Peripheral 3 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 3 Slave.

NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 357. AHB_PERIPH3_SLAVE_RULE0

Rule	Address Access
POWERQUAD	0x4020 0000 – 0x4020 0FFF
CASPER	0x4020 1000 – 0x4020 1FFF
CASPER_RAM	0x4020 2000 – 0x4020 2FFF
SECURE_GPIO	0x4020 4000 – 0x4020 7FFF
HASH	0x4020 8000 – 0x4020 8FFF
FLEXCOMM8	0x4020 9000 – 0x4020 9FFF
FLEXCOMM9	0x4020 A000 – 0x4020 AFFF
FLEXCOMM10	0x4020 B000 – 0x4020 BFFF

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		FLEXCOMM10		Reserved		FLEXCOMM9		Reserved		FLEXCOMM8		Reserved		HASH	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		SECURE_GPIO		Reserved		CASPER_RAM		Reserved		CASPER		Reserved		POWERQUAD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved; the value read from a reserved bit is not defined.
29-28 FLEXCOMM10	FLEXCOMM 10 Defines the minimal security level to access FLEXCOMM10. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
27-26 —	Reserved; the value read from a reserved bit is not defined.
25-24 FLEXCOMM9	FLEXCOMM 9 Defines the minimal security level to access FLEXCOMM9. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
23-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 FLEXCOMM8	FLEXCOMM 8 Defines the minimal security level to access FLEXCOMM8. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 HASH	HASH Defines the minimal security level to access HASH. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 SECURE_GPIO	Secure GPIO Defines the minimal security level to access Secure GPIO memory. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 CASPER_RAM	CASPER RAM Defines the minimal security level to access the CASPER RAM interface. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 CASPER	CASPER Defines the minimal security level to access CASPER. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 POWERQUAD	POWERQUAD Defines the minimal security level to access POWERQUAD. 00b - Non-secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.56 AHB Peripheral 3 Slave Rule 1 Register (AHB_PERIPH3_SLAVE_RULE1)

Offset

Register	Offset
AHB_PERIPH3_SLAVE_RULE1	434h

Function

This register controls access to the AHB Peripheral 3 Slave. Each rule field controls access to a specific peripheral in AHB Peripheral 3 Slave.

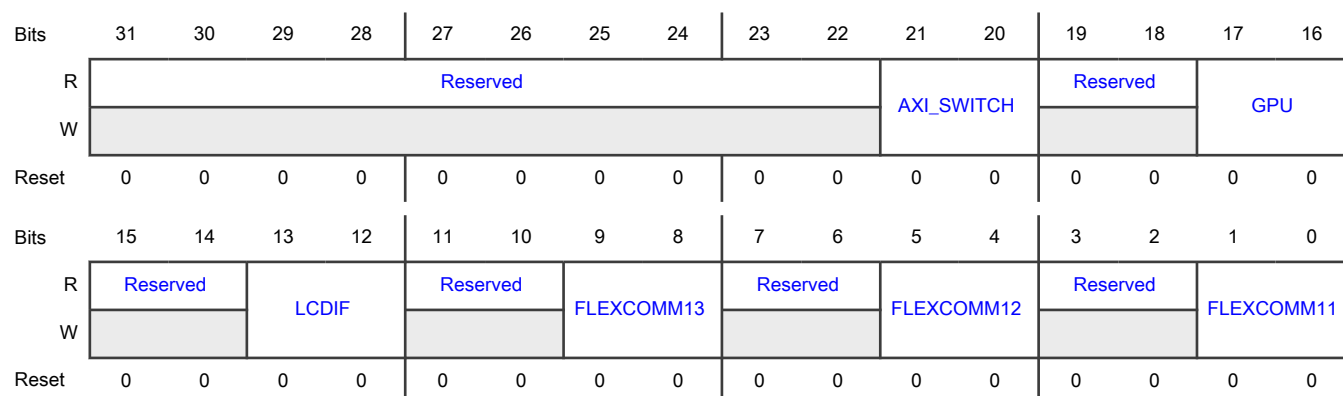
NOTE

Any rule can be set when MIS_CTRL_REG.WRITE_LOCK = 10.

Table 358. AHB_PERIPH3_SLAVE_RULE1

Rule	Address Access
FLEXCOMM11	0x4020 C000 – 0x4020 CFFF
FLEXCOMM12	0x4020 D000 – 0x4020 DFFF
FLEXCOMM13	0x4020 E000 – 0x4020 EFFF
LCDIF	0x4021 0000 – 0x4021 7FFF
GPU	0x4024 0000 – 0x4027 FFFF
AXI_SWITCH	0x4028 0000 – 0x402F FFFF

Diagram



Fields

Field	Function
31-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 AXI_SWITCH	AXI Switch Defines the minimal security level to access the AXI Switch. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
19-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 GPU	GPU Defines the minimal security level to access GPU. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
15-14 —	Reserved; the value read from a reserved bit is not defined.
13-12 LCDIF	LCDIF Defines the minimal security level to access LCDIF. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
11-10 —	Reserved; the value read from a reserved bit is not defined.
9-8 FLEXCOMM13	FLEXCOMM 13 Defines the minimal security level to access FLEXCOMM13. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed

Table continues on the next page...

Table continued from the previous page...

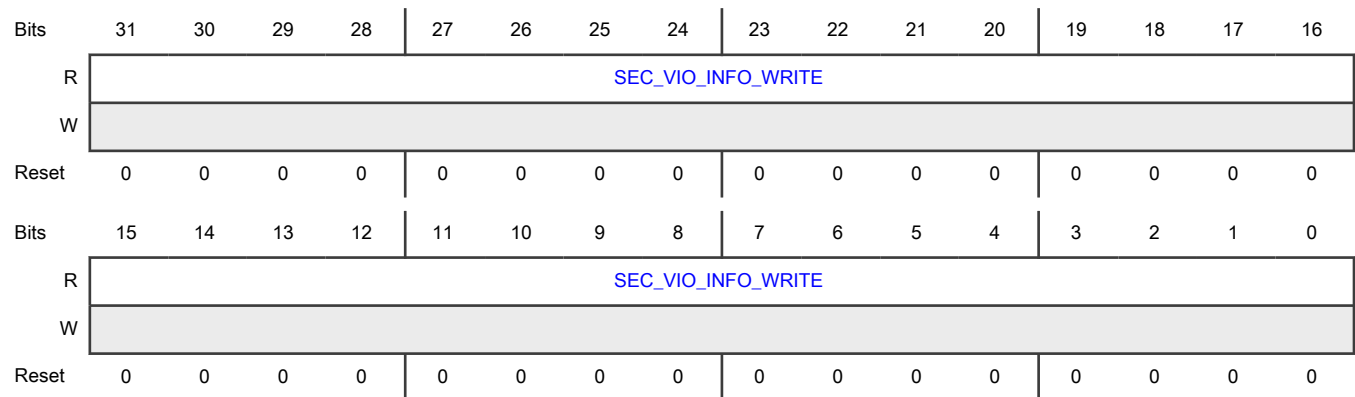
Field	Function
	11b - Secure and privilege user access allowed
7-6 —	Reserved; the value read from a reserved bit is not defined.
5-4 FLEXCOMM12	FLEXCOMM 12 Defines the minimal security level to access FLEXCOMM12. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed
3-2 —	Reserved; the value read from a reserved bit is not defined.
1-0 FLEXCOMM11	FLEXCOMM 11 Defines the minimal security level to access FLEXCOMM11. 00b - Non-secure and non-privilege user access allowed 01b - Non-secure and privilege access allowed 10b - Secure and non-privilege user access allowed 11b - Secure and privilege user access allowed

22.4.1.57 Security Violation Address(n) Register (SEC_VIO_ADDR0 - SEC_VIO_ADDR17)

Offset

For a = 0 to 17:

Register	Offset
SEC_VIO_ADDRa	E00h + (a × 4h)

Diagram**Fields**

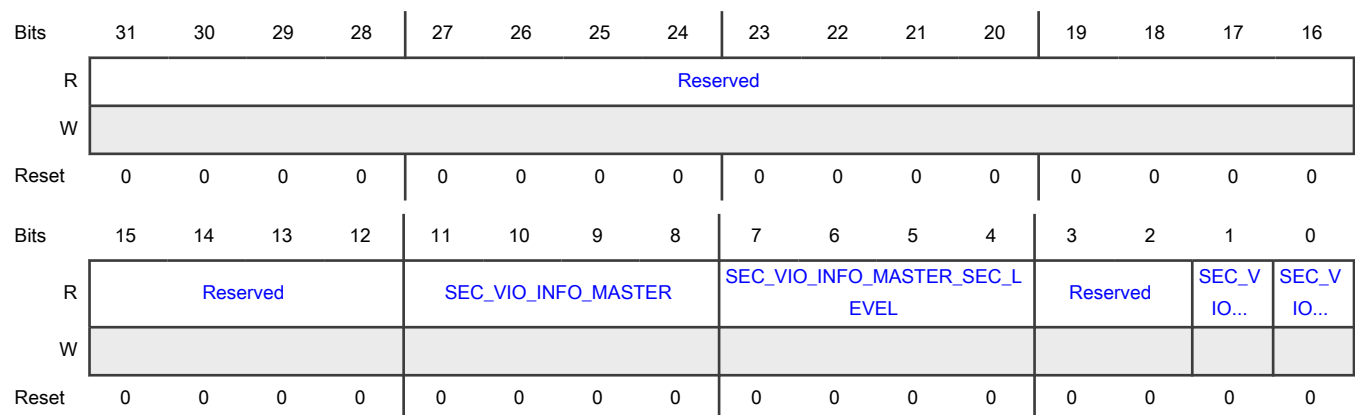
Field	Function
31-0 SEC_VIO_INFO_WRITE	Security violation access read/write indicator

22.4.1.58 Security Violation Miscellaneous Information at Address(n) Register (SEC_VIO_MISC_INFO0 - SEC_VIO_MISC_INFO17)

Offset

For a = 0 to 17:

Register	Offset
SEC_VIO_MISC_INFOa	E80h + (a × 4h)

Diagram

Fields

Field	Function
31-12 —	Reserved; the value read from a reserved bit is not defined.
11-8 SEC_VIO_INFO_MASTER	Security violation master number 0000b - M33 Code 0001b - M33 System 0010b - Powerquad 0011b - DSP 0100b - DMA0 0101b - DMA1 0110b - SDMA Instruction 0111b - SDMA Data 1000b - SDIO0 1001b - SDIO1 1010b - HASH 1011b - GPU 1100b - Reserved 1101b - Reserved 1110b - Reserved 1111b - Reserved
7-4 SEC_VIO_INFO_MASTER_SEC_LEVEL	Security Violation Info Master Security Level • Bits [5:4]: master security level and privilege level • Bits [7:6]: anti-pole value for master sec level and privilege level
3-2 —	Reserved; the value read from a reserved bit is not defined.
1 SEC_VIO_INFO_DATA_ACCESS	Security Violation Info Data Access Indicates whether the security violations occurred on a data access or a code access 0b - Code 1b - Data
0 SEC_VIO_INFO_WRITE	Security violation access read/write indicator Indicates whether the security violation occurred on a read or write access 0b - Read access 1b - Write access

22.4.1.59 Security Violation Info Validity for Address(n) Register (SEC_VIO_INFO_VALID)

Offset

Register	Offset
SEC_VIO_INFO_VALID	F00h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved													VIO_IN F...	VIO_IN F...	VIO_IN F...
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...	VIO_I NF...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-19 —	Reserved; the value read from a reserved bit is not defined.
18-0 VIO_INFO_VAL IDn	Violation information valid flag for AHB port n Write 1 to a bit to clear it. 0b - Not valid 1b - Valid

22.4.1.60 GPIO Mask for Port 0 Register (SEC_GPIO_MASK0)

Offset

Register	Offset
SEC_GPIO_MASK0	F80h

Function

GPIO mask for port 0 pins. This register is used to block leakage of secure interface pin states to the non-secure world. Peripherals can be configured as secure, using a SEC_RULE for each peripheral.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_	PIO0_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 PIO _n _PIN _n _SE C_MASK	Mask bit 0b - Masked 1b - Not masked

22.4.1.61 GPIO Mask for Port 1 Register (SEC_GPIO_MASK1)**Offset**

Register	Offset
SEC_GPIO_MASK1	F84h

Function

GPIO mask for port 1 pins. This register is used to block leakage of secure interface pin states to the non-secure world. Peripherals can be configured as secure, using a SEC_RULE for each peripheral.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_	PIO1_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 PIO _n _PIN _n _SE C_MASK	Mask bit 0b - Masked 1b - Not masked

22.4.1.62 GPIO Mask for Port 2 Register (SEC_GPIO_MASK2)**Offset**

Register	Offset
SEC_GPIO_MASK2	F88h

Function

GPIO mask for port 2 pins. This register is used to block leakage of secure interface pin states to the non-secure world. Peripherals can be configured as secure, using a SEC_RULE for each peripheral.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_	PIO2_
W	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...	PI...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 PIO _n _PIN _n _SE C_MASK	Mask bit 0b - Masked 1b - Not masked

22.4.1.63 Secure Interrupt Mask for DSP Register (DSP_INT_MASK0)

Offset

Register	Offset
DSP_INT_MASK0	FA0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX
W	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	PMUX	Reserved				
W	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...	_OU...					
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-5 PMUX_OUTn	Mask bit 0b - Masked 1b - Not masked
4-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.64 Secure Mask Lock Register (SEC_MASK_LOCK)

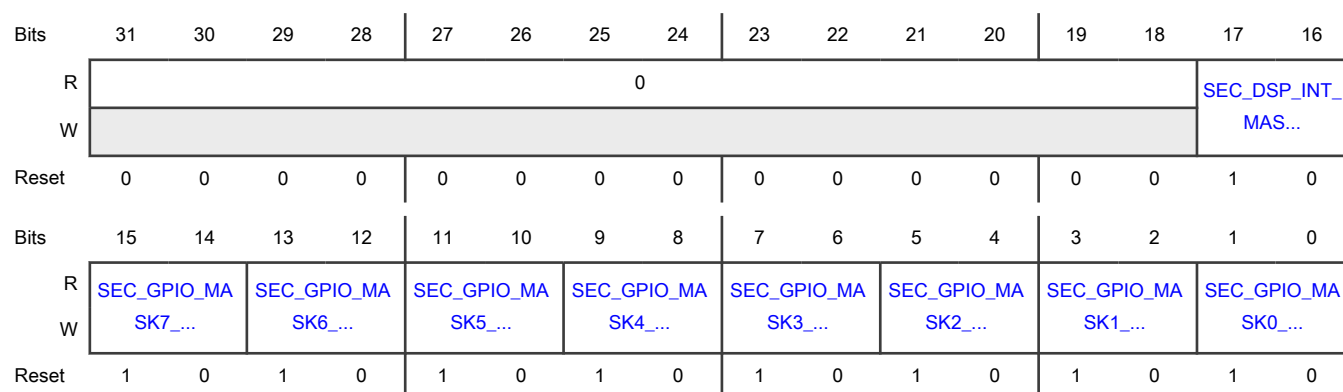
Offset

Register	Offset
SEC_MASK_LOCK	FBCh

Function

For the security of general purpose register access control. Each pair of lock bits can be written once. After changing the default value, the lock bits are locked too.

Diagram



Fields

Field	Function
31-18 —	Reserved; the value read from a reserved bit is not defined.
17-16 SEC_DSP_INT_ MASK_LOCK	SEC_DSP_INT_MASK Lock 00b - Reserved 01b - SEC_DSP_INT_MASK cannot be written 10b - SEC_DSP_INT_MASK can be written 11b - Reserved
15-14 SEC_GPIO_MA SK7_LOCK	SEC_GPIO_MASK7 Lock 00b - Reserved 01b - SEC_GPIO_MASK7 cannot be written 10b - SEC_GPIO_MASK7 can be written 11b - Reserved
13-12 SEC_GPIO_MA SK6_LOCK	SEC_GPIO_MASK6 Lock 00b - Reserved 01b - SEC_GPIO_MASK6 cannot be written 10b - SEC_GPIO_MASK6 can be written 11b - Reserved
11-10 SEC_GPIO_MA SK5_LOCK	SEC_GPIO_MASK5 Lock 00b - Reserved 01b - SEC_GPIO_MASK5 cannot be written 10b - SEC_GPIO_MASK5 can be written 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

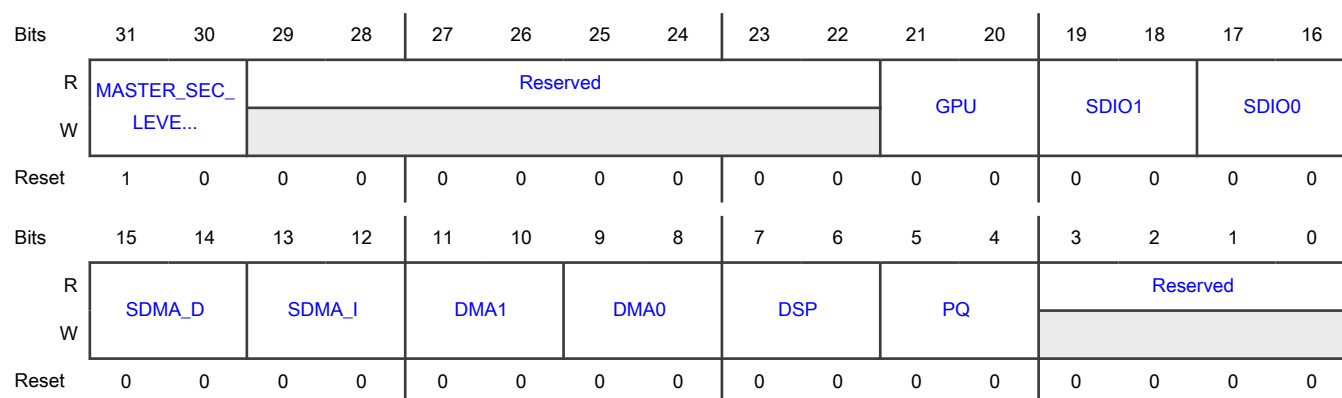
Field	Function
9-8 SEC_GPIO_MA SK4_LOCK	SEC_GPIO_MASK4 Lock 00b - Reserved 01b - SEC_GPIO_MASK4_LOCK cannot be written 10b - SEC_GPIO_MASK4_LOCK can be written 11b - Reserved
7-6 SEC_GPIO_MA SK3_LOCK	Secure GPIO _MASK3 Lock 00b - Reserved 01b - SEC_GPIO_MASK3 cannot be written 10b - SEC_GPIO_MASK3 can be written 11b - Reserved
5-4 SEC_GPIO_MA SK2_LOCK	Secure GPIO _MASK2 Lock 00b - Reserved 01b - SEC_GPIO_MASK2 cannot be written 10b - SEC_GPIO_MASK2 can be written 11b - Reserved
3-2 SEC_GPIO_MA SK1_LOCK	Secure GPIO _MASK1 Lock 00b - Reserved 01b - SEC_GPIO_MASK1 cannot be written 10b - SEC_GPIO_MASK1 can be written 11b - Reserved
1-0 SEC_GPIO_MA SK0_LOCK	Secure GPIO _MASK0 Lock 00b - Reserved 01b - SEC_GPIO_MASK0 cannot be written 10b - SEC_GPIO_MASK0 can be written 11b - Reserved

22.4.1.65 Master Secure Level Register (MASTER_SEC_LEVEL)

Offset

Register	Offset
MASTER_SEC_LEVEL	FD0h

Diagram



Fields

Field	Function
31-30 MASTER_SEC_ LEVEL_LOCK	Master Security Level Lock 00b - Reserved 01b - Lock writing to this register, including these (MASTER_SEC_LEVEL_LOCK) bits 10b - This register can be written 11b - Reserved
29-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 GPU	GPU 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
19-18 SDIO1	SDIO 1 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
17-16 SDIO0	SDIO 0 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-14 SDMA_D	Smart DMA (SDMA) Data 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
13-12 SDMA_I	Smart DMA (SDMA) Instruction 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
11-10 DMA1	DMA 1 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
9-8 DMA0	DMA 0 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
7-6 DSP	DSP 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
5-4 PQ	Power Quad 00b - Non-secure and non-privileged Master 01b - Non-secure and privileged Master 10b - Secure and non-privileged Master 11b - Secure and privileged Master
3-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.66 Master Secure Level Register (MASTER_SEC_ANTI_POL_REG)

Offset

Register	Offset
MASTER_SEC_ANTI_POL_REG	FD4h

Function

The fields in this register must be equal to the same fields in the MASTER_SEC_LEVEL register, but with inverse polarity.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MASTER_SEC_ANTI_POL_LOCK		Reserved								GPU		SDIO1		SDIO0	
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDMA_D		SDMA_I		DMA1		DMA0		DSP		PQ		Reserved			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-30 MASTER_SEC_LEVEL_ANTI_POL_LOCK	Master Security Level Antipole Lock 00b - Reserved 01b - Lock writing to this register, including these (MASTER_SEC_LEVEL_ANTI_POL_LOCK) bits 10b - This register can be written 11b - Reserved
29-22 —	Reserved; the value read from a reserved bit is not defined.
21-20 GPU	GPU Must be equal to NOT(MASTER_SEC_LEVEL[GPU])
19-18 SDIO1	SDIO 1 Must be equal to NOT(MASTER_SEC_LEVEL[SDIO1])
17-16 SDIO0	SDIO 0 Must be equal to NOT(MASTER_SEC_LEVEL[SDIO0])

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-14 SDMA_D	Smart DMA (SDMA) Data Must be equal to NOT(MASTER_SEC_LEVEL[SDMA_D])
13-12 SDMA_I	Smart DMA (SDMA) Instruction Must be equal to NOT(MASTER_SEC_LEVEL[SDMA_I])
11-10 DMA1	DMA 1 Must be equal to NOT(MASTER_SEC_LEVEL[DMA1])
9-8 DMA0	DMA 0 Must be equal to NOT(MASTER_SEC_LEVEL[DMA0])
7-6 DSP	DSP Must be equal to NOT(MASTER_SEC_LEVEL[DSP])
5-4 PQ	Power Quad Must be equal to NOT(MASTER_SEC_LEVEL[PQ])
3-0 —	Reserved; the value read from a reserved bit is not defined.

22.4.1.67 Miscellaneous CPU0 Control Signals Register (CM33_LOCK_REG)

Offset

Register	Offset
CM33_LOCK_REG	FECh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CM33_LOCK_REG_L...															
W	0															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								LOCK_SAU				LOCK_S_MPU			
W	0								LOCK_S_VTOR				LOCK_NS_MPU			
Reset	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0

Fields

Field	Function
31-30 CM33_LOCK_REGISTER_LOCK	Lock CM33 Lock Register After lock, this register (Miscellaneous CPU0 Control Signals Register) will be read-only, and then this register cannot be unlocked. 00b - Reserved 01b - Does not allow writing to this register 10b - Allows writing to this register 11b - Reserved
29-10 —	Reserved
9-8 LOCK_SAU	Lock SAU SAU is the Security Attribution Unit. 00b - Reserved 01b - SAU is locked 10b - SAU can be used 11b - Reserved
7-6 LOCK_S_MPU	Lock Secure MPU 00b - Reserved 01b - Locks Secure MPU 10b - Secure MPU can be used 11b - Reserved
5-4 LOCK_S_VTOR	Lock Secure VTOR 00b - Reserved 01b - Locks Secure VTOR 10b - Secure VTOR can be used 11b - Reserved
3-2 LOCK_NS MPU	Lock Non-Secure MPU 00b - Reserved 01b - Locks Non-Secure MPU 10b - Non-Secure MPU can be used 11b - Reserved
1-0	Lock Non-Secure VTOR 00b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCK_NS_VTOR	01b - Locks Non-Secure VTOR 10b - Non-Secure VTOR can be used 11b - Reserved

22.4.1.68 Secure Control Duplicate Register (MISC_CTRL_DP_REG)

Offset

Register	Offset
MISC_CTRL_DP_REG	FF8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDAU_ALL_NS				DISABLE_SMA_RT_M...				DISABLE_SIMP_LE_...				ENABLE_NS_P_RIV_...			
W																
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Fields

Field	Function
31-16 —	Reserved
15-14 IDAU_ALL_NS	IDAU All Non-Secure IDAU is the Implementation Defined Attribution Unit. 00b - Reserved 01b - IDAU is disabled, which means that all memories are attributed as non-secure memory. 10b - IDAU is enabled (restrictive mode) 11b - Reserved
13-12	Disable Smart Master Strict Mode A Smart master can perform both data and code access (DSP, SDMA) and execute code.

Table continues on the next page...

Table continued from the previous page...

Field	Function
DISABLE_SMART_MASTER_STRICT_MODE	Smart master strict mode only works when ENABLE_SECURE_CHECKING is on. If ENABLE_SECURE_CHECKING is off, then master strict mode is disabled, independently of its value. When enabling ENABLE_SECURE_CHECKING, both strict modes start as enabled (DISABLE_SMART_MASTER_STRICT_MODE=10, can access memories and peripherals at same level only). 00b - Reserved 01b - Can access memories and peripherals at the same level or below that level 10b - Can access memories and peripherals at same level only 11b - Reserved
11-10 DISABLE_SIMPLE_MASTER_STRICT_MODE	Disable Simple Master Strict Mode Simple master strict mode only works when ENABLE_SECURE_CHECKING is on. If ENABLE_SECURE_CHECKING is off, then master strict mode is disabled, independently of its value. When enabling ENABLE_SECURE_CHECKING, both strict modes start as enabled (DISABLE_SMART_MASTER_STRICT_MODE=10, can access memories and peripherals at same level only). A Simple master can perform only data access (other masters). 00b - Reserved 01b - Can access memories and peripherals at the same level or below that level 10b - Can access memories and peripherals at same level only 11b - Reserved
9-8 DISABLE_VIOLATION_ABORT	Disable Violation Abort 00b - Reserved 01b - The violation detected by the secure checker will not cause an abort, but a secure_violation_irq (interrupt request) will still be asserted and serviced by ISR. 10b - The violation detected by the secure checker will cause an abort. 11b - Reserved
7-6 ENABLE_NS_PRIV_CHECK	Enable Non-Secure Privilege Checking 00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved
5-4 ENABLE_S_PRIV_CHECK	Enable Secure Privilege Checking 00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 ENABLE_SECURE_CHECKING	Enable Secure Checking 00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved
1-0 WRITE_LOCK	Write Lock After lock, this register (Secure Control Duplicate Register) will be read-only, and then this register cannot be unlocked. 00b - Reserved 01b - Writes to this register and to the Memory and Peripheral RULE registers are not allowed 10b - Writes to this register and to the Memory and Peripheral RULE registers are allowed 11b - Reserved

22.4.1.69 Secure Control Register (MISC_CTRL_REG)

Offset

Register	Offset
MISC_CTRL_REG	FFCh

Diagram

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

Fields

Field	Function
31-16	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15-14 IDAU_ALL_NS	IDAU All Non-Secure IDAU is the Implementation Defined Attribution Unit. 00b - Reserved 01b - IDAU is disabled, which means that all memories are attributed as non-secure memory. 10b - IDAU is enabled (restrictive mode) 11b - Reserved
13-12 DISABLE_SMART_MASTER_STRICT_MODE	Disable Smart Master Strict Mode A Smart master can perform both data and code access (DSP, SDMA) and execute code. Smart master strict mode only works when ENABLE_SECURE_CHECKING is on. If ENABLE_SECURE_CHECKING is off, then master strict mode is disabled, independently of its value. When enabling ENABLE_SECURE_CHECKING, both strict modes start as enabled (DISABLE_SMART_MASTER_STRICT_MODE=10, can access memories and peripherals at same level only). 00b - Reserved 01b - Can access memories and peripherals at the same level or below that level 10b - Can access memories and peripherals at same level only 11b - Reserved
11-10 DISABLE_SIMPLE_MASTER_STRICT_MODE	Disable Simple Master Strict Mode Simple master strict mode only works when ENABLE_SECURE_CHECKING is on. If ENABLE_SECURE_CHECKING is off, then master strict mode is disabled, independently of its value. When enabling ENABLE_SECURE_CHECKING, both strict modes start as enabled (DISABLE_SMART_MASTER_STRICT_MODE=10, can access memories and peripherals at same level only). A Simple master can perform only data access (other masters). 00b - Reserved 01b - Can access memories and peripherals at the same level or below that level 10b - Can access memories and peripherals at same level only 11b - Reserved
9-8 DISABLE_VIOLATION_ABORT	Disable Violation Abort 00b - Reserved 01b - The violation detected by the secure checker will not cause an abort, but a secure_violation_irq (interrupt request) will still be asserted and serviced by ISR. 10b - The violation detected by the secure checker will cause an abort. 11b - Reserved
7-6	Enable Non-Secure Privilege Checking

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENABLE_NS_PRIV_CHECK	00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved
5-4 ENABLE_S_PRIV_CHECK	Enable Secure Privilege Checking 00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved
3-2 ENABLE_SECURE_CHECKING	Enable Secure Checking 00b - Reserved 01b - Enabled (restrictive mode) 10b - Disabled 11b - Reserved
1-0 WRITE_LOCK	Write Lock After lock, this register (Secure Control Duplicate Register) will be read-only, and then this register cannot be unlocked. 00b - Reserved 01b - Writes to this register and to the Memory and Peripheral RULE registers are not allowed 10b - Writes to this register and to the Memory and Peripheral RULE registers are allowed 11b - Reserved

Chapter 23

PowerQuad DSP Coprocessor and Accelerator (PowerQuad)

23.1 Chip-specific PowerQuad information

Table 359. Reference links to related information

Topic	Related module	Reference
Full description	PowerQuad	PowerQuad
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

23.1.1 Module instances

This device has only one instance of the PowerQuad module, POWERQUAD0.

23.2 Overview

PowerQuad is a DSP Accelerator, designed to assist Cortex-M, working within the Arm ecosystem. The PowerQuad hardware module is designed to accelerate some general DSP computing tasks, such as math, matrix, filter, and transform functions including Fast Fourier Transforms (FFT). Because hardware executes the computing tasks outside the Arm core, tasks run faster and save CPU time.

As technology evolves, so does the need for smarter and lower-power IoT nodes, so the computational performance requirements for local analysis of data is growing. This requirement poses some challenges for the Cortex-M architecture, designed for low power and real-time control. That strength allows a very light layer of software to be able to control I/O and peripherals in a tightly coupled deterministic manner. This same architecture that is advantageous for MCU control, is not optimal for the task of handling vast amounts of data. The purpose of the PowerQuad hardware accelerator is to assist the Cortex-M33 (CM33) core with the handling of digital signal processing.

PowerQuad is a hardware accelerator for common calculations in DSP applications. PowerQuad includes seven internal computation engines:

- Transform
- Transcendental function
- Trigonometry function
- Dual Biquad infinite impulse response (IIR) filter
- Matrix accelerator
- Finite impulse response (FIR) filter
- Coordinate rotation digital computer (CORDIC)

By significantly reducing the runtime and the energy requirements of signal processing algorithms, PowerQuad allows you to run more accurate, elaborate algorithms in an energy-efficient manner.

23.2.1 Block diagram

PowerQuad is a digital signal coprocessing companion to the CM33 core. You can access PowerQuad through the coprocessor interface, the memory mapped interface, or through the CORDIC interface address space.

PowerQuad is primarily based on a quad floating-point multiply accumulator (MAC) compute block. This four-way MAC implements almost all functions supported by the accelerator. The other compute elements in PowerQuad are the FFT radix-8 butterfly engine (for FFT/DCT calculations) and the CORDIC for inverse trigonometric functions (arctan, arctanh).

Since the main compute engine is natively in floating-point format, this simplifies operations that are not tightly bound in terms of input data or result. Having a floating-point format ensures that a certain resolution can be achieved without the need for saturating the checking and/or implementing logic.

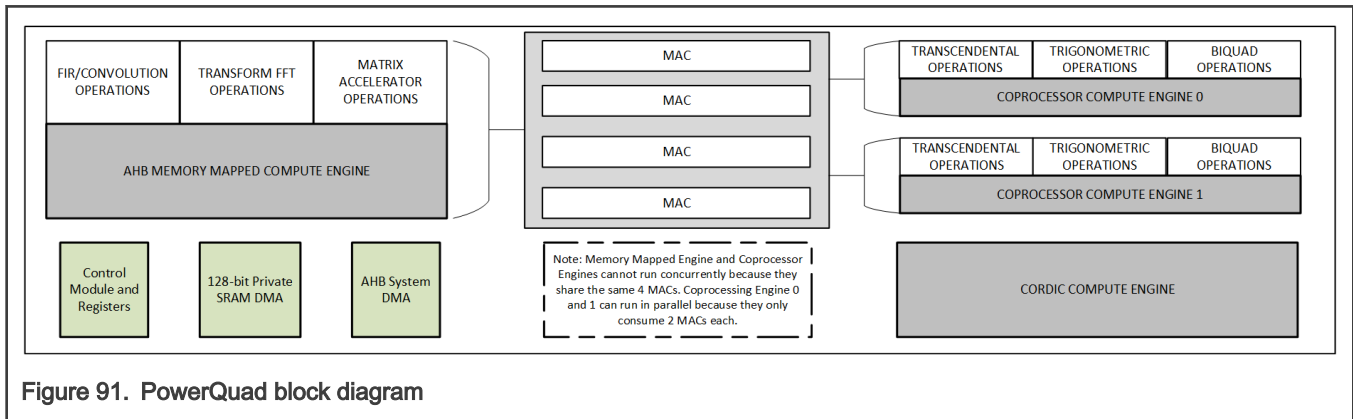


Figure 91. PowerQuad block diagram

23.2.2 Features

The key features of PowerQuad are:

- Communication interfaces:
 - Coprocessor interface from the CM33 core, a register-to-register interaction.
 - AHB as a subordinate interface, use to configure the PowerQuad registers.
 - AHB as a manager interface for PowerQuad to fetch or write data to and from the system memory, when required.
 - 128-bit wide RAM controller that can access up-to 4 RAM banks simultaneously to fetch or write data as needed.
- Four single-precision floating-point multiply accumulator (MAC) units
- Coprocessor engines for tightly coupled opcodes: $\sin(x)$, $\cos(x)$, $\ln(x)$, $e^x(x)$, e^{-x} , $1/(x)$, $1/\sqrt{x}$, $\sqrt{x}$, $\text{biquad}(x)$. Using two MACs, you can run two opcodes in parallel
- Memory-mapped
 - FFT/iFFT/DCT/iDCT operations
 - Matrix operations: add, sub, dot, prod, mult, inverse, transpose, scale
 - Convolution/Correlation/FIR
- Arctan/Arctanh (these operations can be used as the base to compute any CORDIC function)

With the assistance of PowerQuad, the CM33 core is free to perform other tasks. While PowerQuad executes an assigned computation task, the CM33 core can prepare the next PowerQuad task, resulting in a pipeline of PowerQuad tasks.

23.3 Functional description

In a conventional MCU without PowerQuad, the main CPU performs all of the computation work. When using the PowerQuad, work is accelerated and depending on the operation in progress the main CPU can continue to work on other tasks while the PowerQuad finishes processing the operation. For example, transcendental functions are now essentially assembly instructions

for the CM33 core. Other larger functions (such as FFT, or matrix calculations) can be considered as a kind of DMA job, where the core can start them and receive an interrupt or event when the job completes.

Using PowerQuad, you can treat PowerQuad as a coprocessor, performing multiple tasks by rearranging them and reducing the branching to speed up computational processing. The PowerQuad engines compute the assigned functions faster than using the CM33 core alone.

The aim of the PowerQuad coprocessor is to serve the following purposes:

- Provide significant energy reduction for the execution of digital signal processing algorithms
- Significantly accelerate digital signal processing operations compared to the Cortex-M core alone
- Provide hardware acceleration for common DSP functions (mainly from the CMSIS-DSP library)

After applying the clock to PowerQuad, based on the requested operation, one of the computation engines activates. All PowerQuad operations fall into one of three access types. Access PowerQuad through either:

- The coprocessor engines (CP)
- The AHB memory mapped engine (AHB).
- The CORDIC engine

23.3.1 Coprocessor interface

23.3.1.1 Introduction

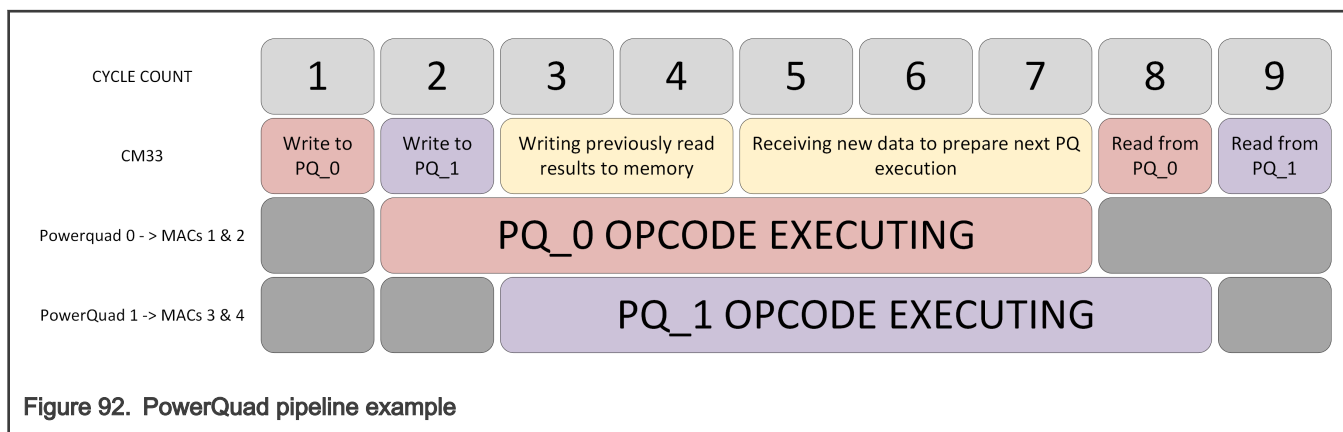
The ARMv8-M architecture (CM33) introduces a coprocessor interface, which allows PowerQuad access through opcodes:

- MCR (Move from coprocessor to Register)
- MRC (Move from Register to coprocessor)

When accessing PowerQuad through the coprocessor interface, executing a correctly formatted coprocessor instruction launches the operation. Executing a second coprocessor instruction retrieves the result. The coprocessor is self-stalling, which means that if you perform a read from a result register, the access stalls until the computation is complete. These take 2 MACS out of the 4 to implement, allowing you to run two operations in parallel.

To achieve the best performance, use vectorized forms of the coprocessor instructions. These typically handle the work at function calls and follow the timing as shown in the figure below. These vectorized forms are already coded by NXP to achieve timing as shown (note that the figure shows just 2 calculations. In the vectorized functions, we perform 4 of these in a row (that is, 8 calculations)).

When submitting the data and/or opcodes to a coprocessor, the CM33 core can continue executing other tasks while the coprocessor computes in parallel. This parallel work ensures that the CM33 core and PowerQuad are not left idle, maximizing the performance of the work done.



In PowerQuad, this interface uses short (several cycles) operations, for which the CM33 core provides data from one of its registers. The result is stored back in the CM33 register. The above diagram represents a snippet in time where the CM33 core and PowerQuad work in parallel. The CM33 core requests that PowerQuad 0 begin to execute an opcode provided. In the following cycle the CM33 core requests that PowerQuad 1 also begin to execute an opcode. Meanwhile, PowerQuad is fully loaded and the CM33 core takes this time to write back any data previously received from PowerQuad. The CM33 core also uses this time to prepare the next PowerQuad requests. Once PowerQuad 0 finishes, the CM33 core fetches the data by reading back the result in one cycle. Then in the next cycle reads back the result from PowerQuad 1. You can repeat this pipelined effort continuously to optimize the workload on the CM33 core while PowerQuad is busy.

When the data comes to PowerQuad in fixed-point or floating-point format, internally PowerQuad handles the data as floating point. The conversion from fixed-point to floating-point format occurs automatically. Prescaling and postscaling options are available to allow fixed-point numbers with fractional precision.

Results of PowerQuad operations can be either fixed-point or floating-point format (conversion from floating-point to fixed-point format, if needed).

The functions handled in this way are:

- Transcendental: $\ln(x)$, e^x , e^{-x} , $1/(x)$, $1/\sqrt{x}$, $\sqrt{x}$, $x1/x2$
- Trigonometric: $\sin(x)$, $\cos(x)$
- Biquad: $\text{biquad}(x)$

For PowerQuad Coprocessing interface examples, see [Construct coprocessor interface operations](#).

23.3.1.2 Transcendental

In mathematics, a transcendental function is a type of function that you cannot express algebraically. A possible solution to simplify the implementation of these algorithms is to create a lookup table. Lookup tables can help to simplify or speed up an inefficient process, however lookup tables take extra space in the memory of the device. A key advantage of PowerQuad is that there is no need for lookup tables, so you do not waste RAM for transcendental functions. All of this work is taken care of by the hardware module.

Another advantage of transcendental function implementation for PowerQuad is that they become actual instructions in the CM33 core. The functions give high precision in a matter of a handful of cycles.

Since PowerQuad can execute 2 of these functions in parallel, the number of cycles consumed matches the number of cycles the CM33 core requires to bring in more data for PowerQuad and write out the results. Therefore, you can fully pipeline PowerQuad running in parallel with the CM33 core and achieve almost single-cycle effective performance for the calculations.

PowerQuad supports the following transcendental functions:

- $\ln(x)$
- e^x
- e^{-x}
- $1/(x)$
- $1/\sqrt{x}$
- $\sqrt{x}$
- $x1/x2$

23.3.1.2.1 Requirements

Input and output data support float, INT16, and INT32.

23.3.1.2.2 Performance

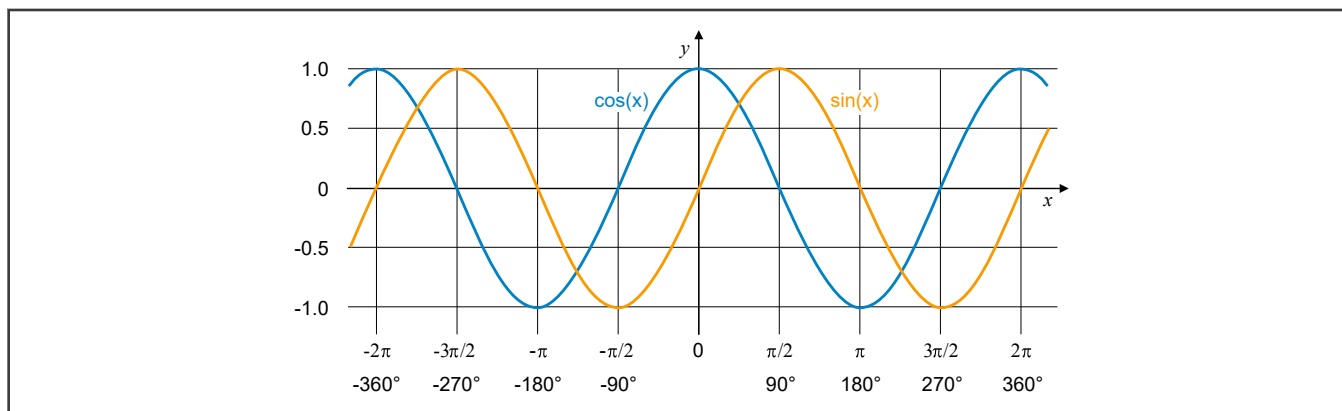
Table 360. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Transcendental	PQ_LnF32	15.0E-6	80.0E-9	187.6	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed Note, the math functions running on CM33 are only supported on math.h
	PQ_SqrtF32	4.2E-6	93.3E-9	45.3	
	PQ_InvSqrtF32	4.2E-6	80.0E-9	52.8	
	PQ_EtoxF32	18.8E-6	86.7E-9	217.2	
	PQ_EtonxF32	18.8E-6	86.7E-9	217.2	
	PQ_LnFixed	15.0E-6	60.0E-9	250.2	
	PQ_SqrtFixed	4.2E-6	73.3E-9	56.9	
	PQ_InvSqrtFixed	4.2E-6	66.7E-9	62.7	
	PQ_EtoxFixed	19.4E-6	60.0E-9	324.0	
	PQ_EtonxFixed	19.4E-6	73.3E-9	264.3	

23.3.1.3 Trigonometry

The trigonometric functions most widely used in modern mathematics are the sine, the cosine, and the tangent. In embedded applications, trigonometric functions are used in robotics, motor control, signal processing among others. It is essential to process data quickly to benefit from the timing aspects of each application.

The Taylor series of $\sin(x)$ and $\cos(x)$ centered about 0 have the property of having best convergence (fewest terms) when closest to 0. By partitioning the angular space into half-quadrants (that is $\pi/2$ or 45 degrees), you can reuse the basic Taylor series for the $\sin(x)$ and $\cos(x)$ in the range $[0, \pi/2]$.



Tangent can be calculated using the equation:

$$\tan(x) = \frac{\sin(x)}{\cos(x)} = \frac{\sin(x)}{\sqrt{1 - \sin^2(x)}}$$

The tangent function is not directly mapped as a PowerQuad function. To calculate the tangent of a given value, you can perform a sine and cosine function in parallel and then perform the division of those results. In addition, you could also do sine and inverse square root, however, this method would consume more cycles than the first option.

23.3.1.3.1 Requirements

Input and output floating-point data support radian units, Q15 and Q31 fixed-point data support normalized values.

For fixed-point inputs, the input value is normalized to range -1 to 1 which represents -pi to pi.

For floating-point inputs, the input value is in radians. This function calculates -pi to pi.

23.3.1.3.2 Performance

Table 361. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Trigonometry	PQ_SinF32	686.7E-9	206.7E-9	3.3	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	PQ_CosF32	720.0E-9	180.0E-9	4.0	
	PQ_SinQ15	413.3E-9	286.7E-9	1.4	
	PQ_CosQ15	453.3E-9	233.3E-9	1.9	
	PQ_SinQ31	493.3E-9	233.3E-9	2.1	
	PQ_CosQ31	480.0E-9	226.7E-9	2.1	

23.3.1.4 Dual Biquad IIR

In signal processing, a digital Biquad filter is a second order recursive linear filter, containing two poles and two zeros.

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{a_0 + a_1z^{-1} + a_2z^{-2}}$$

The coefficients are often normalized such that $a_0 = 1$:

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}$$

High-order IIR filters can become unstable due to quantization of the coefficients. Typically higher-order filters are implemented in cascaded Biquad portions. PowerQuad can execute cascaded filters by saving the internal states of each run and restoring the states as the data is processed.

The Biquad engine uses the coprocessor interface. Use the APIs provided in the SDK, as executing a Biquad filter is composed of several steps. The MCR and MRC opcodes are used sequentially depending on the order of the filter used. A second order recursive digital filter is shown below.

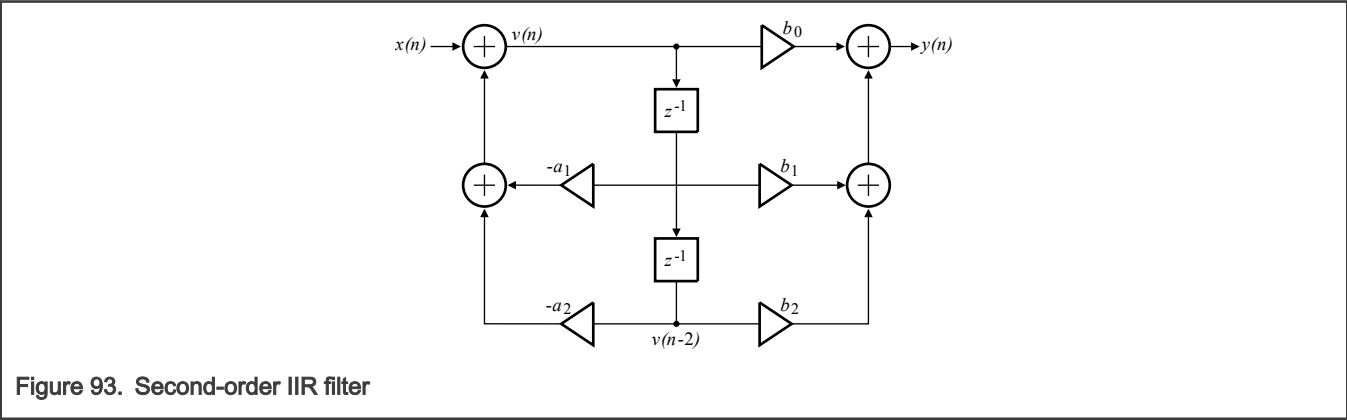


Figure 93. Second-order IIR filter

23.3.1.4.1 Requirements

Input and output data support float, INT16, and INT32.

For single Biquad computations, Biquad state 0 is used.

For cascade Biquad computations, the data flow is input -> Biquad state 1 -> Biquad state 0 -> output.

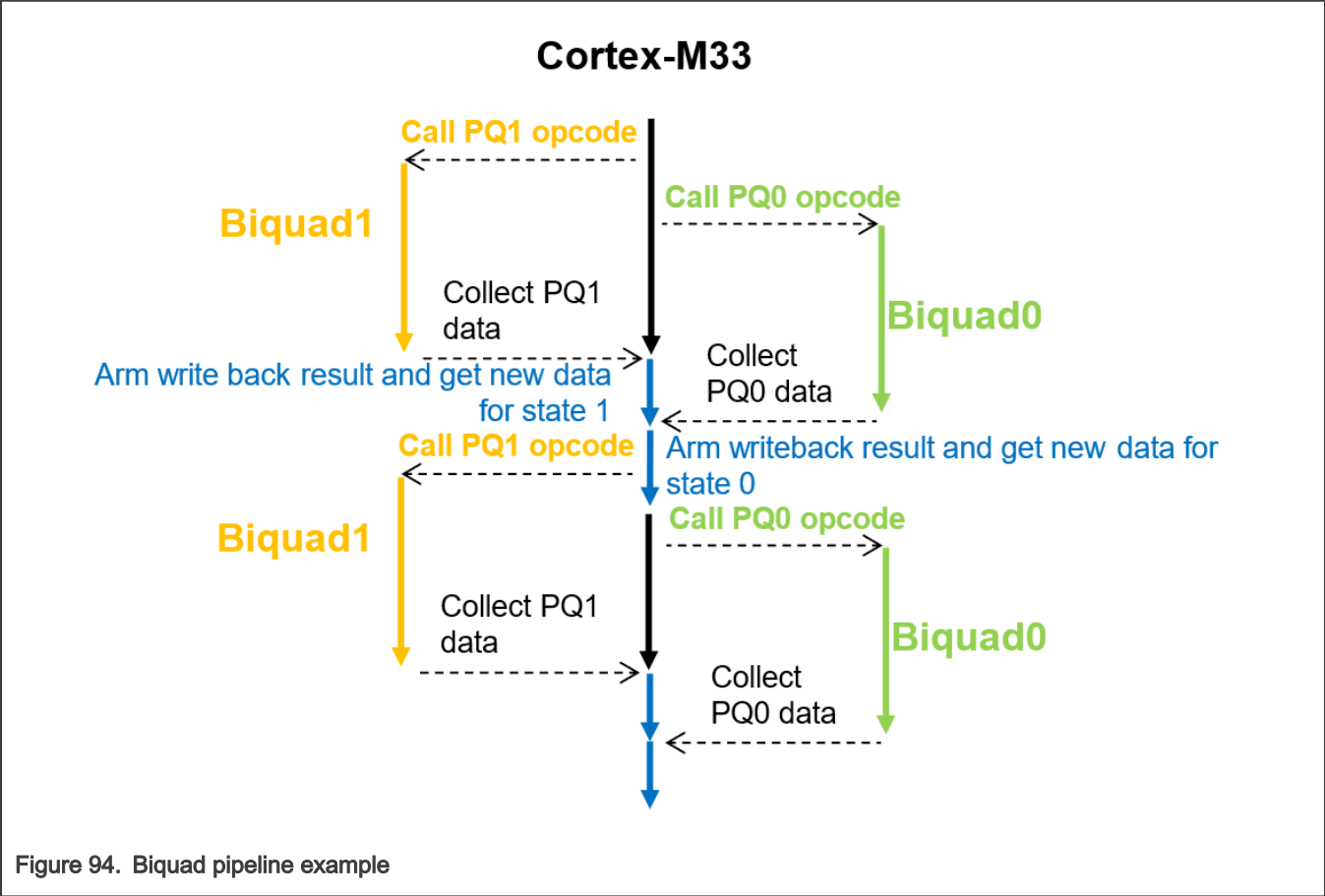


Figure 94. Biquad pipeline example

23.3.1.4.2 Performance

Table 362. Performance using PowerQuad

Function	API	Using PowerQuad (s)	Conditions
Biquad IIR	PQ_BiquadF32	5.6E-6	Frequency 150 MHz
	PQ_BiquadIncrementalF32	4.2E-6	IAR 9.30.1
	PQ_BiquadFixed16	1.5E-6	SDK 2.13.0
	PQ_BiquadFixed32	1.3E-6	Running on Flash
	BiquadCascadeFixed16	4.0E-6	Optimization O3 Speed
	BiquadCascadeFixed32	3.9E-6	SW Math Comparison does not exist at the moment.
	BiquadCascadeF32	89.7E-6	

23.3.2 AHB memory-mapped engine

23.3.2.1 Introduction

PowerQuad handles large data operations in a memory-mapped fashion. To access PowerQuad through this method, write to [Control \(CONTROL\)](#) to launch a memory-mapped operation. When the result is ready, an event or interrupt generates to inform the CM33 core that the result is ready. Alternatively, the CM33 core can also poll [CONTROL\[INST_BUSY\]](#). Retrieve results through one or more AHB reads from the PowerQuad's private RAM or the system's memory.

Some operations must use system memory for intermediate operations or to save context while using the PowerQuad for another operation. Handle these operations through the 32-bit AHB bus. Most memory-mapped operations use the private 128-bit wide memory. Use both the AHB bus and private memory if possible. That is, have one operand coming from system memory and one from private memory, such that you maximize bandwidth and performance. Move operands into private memory by choosing private memory as the result destination pointer. The private memory always keeps the data in the native format (which is floating point in all cases except for FFT, which uses fixed point). In FFT, we recommend that you use private memory for the intermediate calculations, as a temporary region. For other calculations, the recommendation is that one side comes from AHB (system) and the other from private memory. In order to get the operand into private memory, use a matrix scale operation (scale by 1.0) and have an address in private memory as the destination.

PowerQuad is natively 32-bit floating-point, so temporary results in private memory can only be in the same format. The only exception is the FFT engine, which uses fixed-point data. The 4-region memory model configures these parameters:

- Scaling and data format for the operands
- Temporary results
- Final result to fixed-point or floating-point format

With PowerQuad, you can change the format, prescale, or postscale the data on the fly, which is a huge advantage, saving cycles to increase performance.

The CM33 core writes to the control registers of PowerQuad to:

- Select the function
- Provide 4 memory regions with source destination addresses
- Save a temporary scratch address for intermediate results (where applicable such as with FFT and Matrix Inversion)

Poll completion in [CONTROL\[INST_BUSY\]](#) (cleared to '0'), or generate an interrupt/event from the PowerQuad.

The functions handled in this way are:

- Matrix (add, sub, scale, invert, transpose, multiply, dot, product)
- FFT/IFFT/DCT/IDCT (note that DCT is partial in PowerQuad and requires some adjustments)
- FIR/Convolution/Correlation

For PowerQuad AHB memory-mapped interface examples, see [Construct AHB interface operations](#).

For any given function where PowerQuad interacts with data (from AHB system memory or private RAM), a user set configuration for 4 regions gives the addressing and the interface for interacting with the data.

The regions are:

- Output
- Temp
- Input A
- Input B

The PowerQuad control registers allow the user to choose the base address for each of the regions used in the function. Apart from a base address for each region, the user can also choose the format for the data on the inside and the outside. For example, you can configure Input A as INT16 fixed-point input but use it as floating point by PowerQuad. The format in or out can only be chosen if the region used is in AHB space. All data written to or read from private RAM does not undergo any conversion.

The internal or external formats can be any of the following:

- Int16 Fixed point
- Int32 Fixed point
- Floating point

In addition to formatting the input and output, you can also choose a scaling factor for the fixed-point inputs and outputs.

23.3.2.2 Memory Handling

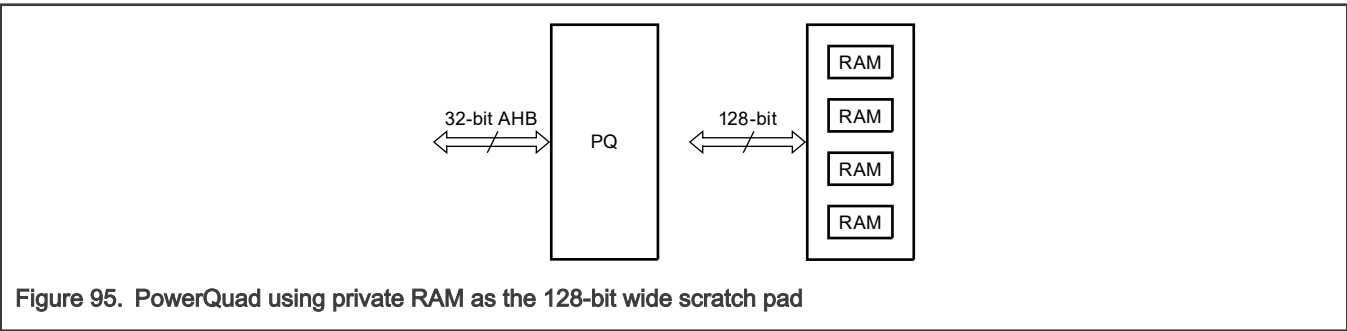
PowerQuad has four memory pointer regions:

Input A pointer	Stores the argument A for the algorithm
Input B pointer	Stores the argument B for algorithm (optional)
Temp pointer	Area used for intermediate calculations
Output pointer	Where the result is stored

Each of the memory regions may be configured in the following manner:

Origin data	32-bit fixed, 16-bit fixed, or 32-bit float
Data used in PowerQuad	All data must be float except in the FFT engine which are fixed.
Scaling	PowerQuad can do scaling by power of 2 for input or output.

Each of these regions may reside in internal memory or may reside in private memory. PowerQuad accesses and collects the data accordingly. It is able to access AHB memory space for the entire address space, except for the range E000_0000h – EFFF_FFFFh. The address space starting with E000_0000h is that through which, PowerQuad can access its private RAM. It is recommended to make use of private memory as this region is only accessible by PowerQuad.



23.3.2.3 Transform

PowerQuad provides discrete Fourier transforms and discrete cosine transforms, implemented with a Radix-8 Butterfly structure Fast Fourier Transform (FFT) engine using fixed-point arithmetic at a resolution of 24 bits. The Radix-8 butterfly structure of the engine is shown below. This implementation both reduces the number of memory accesses and makes full use of the four MACs available in PowerQuad.

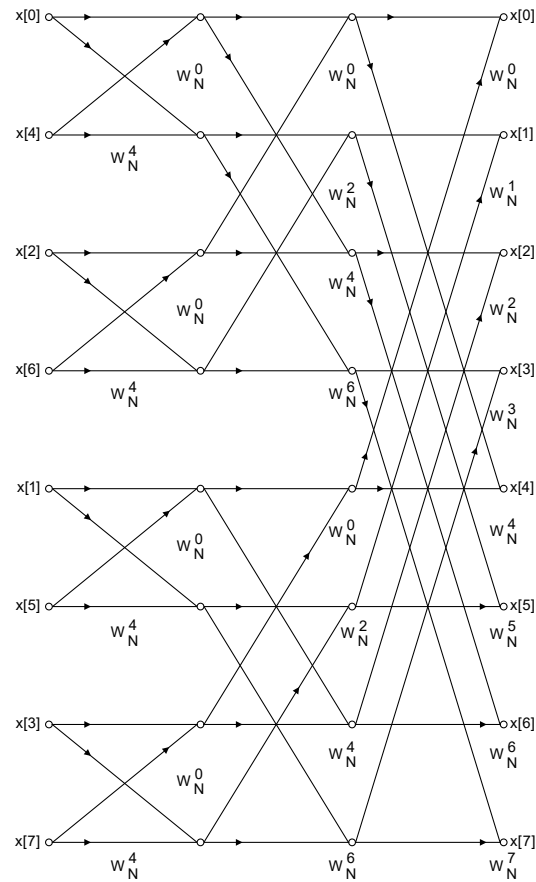


Figure 96. Radix-8 butterfly structure

The discrete Fourier transform (DFT) transforms a sequence of N complex numbers:

$$\{x_n\} := x_0, x_1, \dots, x_{N-1}$$

Into another sequence of complex numbers:

$$\{X_k\} := X_0, X_1, \dots, X_{N-1}$$

Defined by:

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N}$$

With the inverse transform defined as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

The discrete cosine transform is similar to the discrete Fourier transform but uses only real numbers. It transforms a sequence of N real numbers:

$$\{x_n\} := x_0, x_1, \dots, x_{N-1}$$

into another sequence of real numbers:

$$\{X_k\} := X_0, X_1, \dots, X_{N-1}$$

Defined by:

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right], \text{ where } k = 0, \dots, N-1$$

And the inverse defined by:

$$X_k = \frac{1}{2} x_0 + \sum_{n=1}^{N-1} x_n \cos \left[\frac{\pi}{N} n \left(k + \frac{1}{2} \right) \right], \text{ where } k = 0, \dots, N-1$$

FFT and IFFT are fully performed by PowerQuad. In the case of DCT, PowerQuad only performs odd/even mirroring of the original input before performing an FFT on the input. For example, input of [0 1 2 3 4 5 6 7] gets odd/even mirroring that results in [0 2 4 6 7 5 3 1] before entering FFT. To reach a final DCT value, ARM is required to multiply the resulting spectrum with a rotating phasor. For IDCT, ARM needs to multiply the input by a phasor. The PowerQuad IDCT machine then performs an inverse FFT and then the reverse of the odd/even mirroring done by the transform.

23.3.2.3.1 Requirements

Input and output data support INT16 and INT32.

Important aspects of the input data for the FFT engine are:

- Only considers the bottom 27 bits of the input word, therefore prescaling must not exceed this limit to avoid saturation.
- Always scales the input data by 1/N when computing. If an unscaled result is necessary, the input data (in the INPUT A region) must first be multiplied by N. The inverse FFT is also scaled by 1/N, but is correct per the formula, so no scaling treatment is needed.
- Achieves scaling (that is, multiplying by N) at no cost by configuring the prescaling for the input into FFT.
- Supports these lengths for DCT and DFT are N = 16, 32, 64, 128, 256, 512.

NOTE

PowerQuad can perform scaling by N on the way in with no impact to performance.

Real number sequences must be arranged in memory such that if a sequence is as follows:

$$\{x_n\} := x_0, x_1, \dots, x_{N-1}$$

Then the input array in memory is:

$$x[N] = \{x_0, x_1, \dots, x_{N-1}\}$$

Complex number sequences must be arranged in memory such that if a sequence is as follows:

$$\{x_n\} := x_0, x_1, \dots, x_{N-1}, \text{ where } x_n = x_{nreal} + x_{nim}$$

Then the input array in memory is:

$$x[N] = \{x_{0real}, x_{0im}, x_{1real}, x_{1im}, \dots, x_{N-1real}, x_{N-1im}\}$$

For the output sequence, the array is always made up of complex numbers, where the imaginary parts are zero for real number operations.

23.3.2.3.2 Performance

Table 363. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
FFT	rfft_q15	446.6E-6	20.1E-6	22.2	Frequency 150 MHz

Table continues on the next page...

Table 363. Performance using CM33 alone versus with PowerQuad (continued)

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
	rfft_q31	537.5E-6	20.1E-6	26.7	IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	cfft_q15	674.9E-6	22.2E-6	30.4	
	cfft_q31	830.4E-6	22.2E-6	37.4	
	ifft_q15	675.5E-6	22.1E-6	30.5	
	ifft_q31	854.4E-6	22.1E-6	38.6	

23.3.2.4 Matrix

The matrix accelerator engine supports the following operations:

- Addition
- Subtraction
- Multiplication
- Vector Dot-Product
- Inversion
- Transpose
- Scaling

Matrices are composed in the following manner:

In the example matrices below, Matrix A, is composed of 2 rows and 3 columns. Matrix B is composed of 2 rows and 2 columns. Depending on the required matrix operation, the number of rows and columns of each matrix determines if it is possible to proceed.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Adds/Subtracts/product of two matrices:

The number of rows in Matrix A must match the rows in Matrix B. The same applies to the columns of each matrix.

Multiplication of two matrices:

The number of columns in Matrix A must match the number of rows in Matrix B.

Scaling of a matrix:

A matrix is scaled using a floating-point scaler number.

$[R] = k * [M]$, where each element in the matrix is scaled by k. The floating-point scaler number is loaded into the [Miscellaneous \(MISC\)](#) register.

Transpose of a matrix:

The transpose of a matrix is an operator that flips a matrix over its diagonal, that is, it switches the row and column indexes of the matrix by producing another matrix denoted as A^T .

Let A be an $m \times n$ matrix. The transpose of A, denoted A^T , is the $n \times m$ matrix whose columns are the respective rows of A.

Inversion of a matrix:

Matrix inversion is supported only for square matrices. A square matrix is singular only when its determinant is exactly zero. The implementation needs some scratch area in the AHB space. We are not using any of private RAM for this function.

Vector Dot Product:

The dot product is the sum of the products of the corresponding entries of the two sequences of numbers (vectors). The result is one number.

23.3.2.4.1 Requirements

Input and output data support float, INT16, and INT32.

The **Length (LENGTH)** register is remapped when using the matrix engine. It is composed of the concatenation of the columns in matrix_B, the columns in matrix_A, and the rows in matrix_A.

- LENGTH[31:0] = Cols_B[20:16] | Cols_A[12:8] | Rows_A[4:0]

Matrix operations have a maximum dimension as shown below:

Table 364. Matrix operation dimensions

Operation	Maximum size
Addition	16 x 16
Subtraction	
Hadamard Product	
Product	
Transpose	
Scaling	
Inversion	9 x 9
Vector Dot-Product	256 elements

23.3.2.4.2 Performance**Table 365. Performance using CM33 alone versus with PowerQuad**

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Matrix	mat_add_q15	12.2E-6	6.2E-6	2.0	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	mat_add_q31	12.2E-6	6.2E-6	2.0	
	mat_add_f32	19.4E-6	6.2E-6	3.1	
	mat_sub_q15	12.2E-6	6.2E-6	2.0	
	mat_sub_q31	12.2E-6	6.1E-6	2.0	
	mat_sub_f32	19.4E-6	6.2E-6	3.2	
	mat_mult_q15	322.4E-6	19.3E-6	16.7	
	mat_mult_q31	346.9E-6	19.4E-6	17.9	
	mat_mult_f32	352.5E-6	43.7E-6	8.1	

Table continues on the next page...

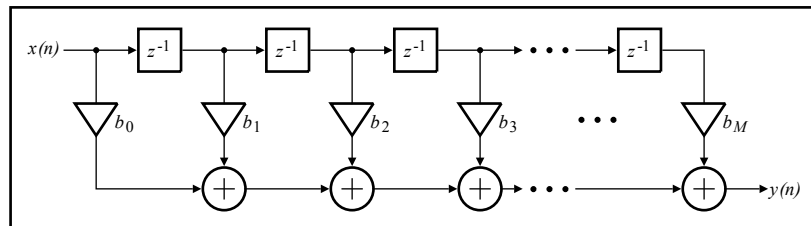
Table 365. Performance using CM33 alone versus with PowerQuad (continued)

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
	mat_inverse_f32	118.4E-6	34.9E-6	3.4	
	mat_trans_q15	10.4E-6	5.1E-6	2.1	
	mat_trans_q31	10.4E-6	5.1E-6	2.0	
	mat_trans_f32	16.8E-6	6.2E-6	2.7	
	mat_scale_q15	17.3E-6	5.1E-6	3.4	
	mat_scale_q31	33.6E-6	5.2E-6	6.5	
	mat_scale_f32	15.6E-6	4.4E-6	3.6	

23.3.2.5 FIR

In digital signal processing, an Nth order discrete time FIR filter does not have feedback, which makes it a very stable filter. However, this type of filter requires many more coefficients to meet a similar outcome of an IIR filter. Due to this requirement, the time it takes to execute a FIR filter is much longer as each coefficient introduces another multiplication in the system. Having PowerQuad as an accelerator of the same operation greatly reduces the impact of having a high-order FIR filter.

The signal flow graph for the general, causal, length $N = M + 1$, finite-impulse-response (FIR) digital filter is shown below.


Figure 97. FIR signal flow

This filter implements the difference equation:

$$y[n] = \sum_{i=0}^M b_i x[n-i], \text{ where } n = 0, \dots, M$$

Here $x[n]$ is the input signal, $y[n]$ is the output signal, and M is the filter order. An M th-order filter has $(M + 1)$ filter taps.

The PowerQuad FIR engine run 4 MAC operations in parallel.

23.3.2.5.1 Requirements

Input and output data support float, INT16, and INT32.

Length (LENGTH) is remapped when using the FIR engine. It is composed of the concatenation of the number of filter taps and the number of samples in the input sequence.

- LENGTH[31:0] = numTaps[31:16] | numSamples[15:0]

Configuring the FIR incremental operations uses the offset from the base address that the FIR operation uses. FIR incremental operation may only be used after FIR operation is used. Prescaling and postscaling the data can be a huge advantage for these operations.

- `PQ_FIR(POWERQUAD, pSrcA, srcALen, pSrcB, srcBLen, pDst, op_type);`
- `PQ_FIRIncremental(POWERQUAD, srcALen, srcBLen, offset);`

For full API examples, check the SDK PowerQuad drivers provided at mcuxpresso.nxp.com.

See the MCUXpresso SDK API Reference Manual included in the SDK document folder SDK.zip/docs.

23.3.2.5.2 Performance

Table 366. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
Filter	<code>fir_q15</code>	20.5E-6	1.9E-6	10.6	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed
	<code>fir_q31</code>	9.9E-6	1.9E-6	5.1	
	<code>fir_f32</code>	16.7E-6	1.8E-6	9.1	
	<code>conv_q15</code>	4.0E-6	1.4E-6	2.8	
	<code>conv_q31</code>	3.9E-6	1.4E-6	2.7	
	<code>conv_f32</code>	4.3E-6	1.4E-6	3.1	
	<code>correl_q15</code>	4.8E-6	1.5E-6	3.2	
	<code>correl_q31</code>	4.0E-6	1.5E-6	2.7	
	<code>correl_f32</code>	4.2E-6	1.4E-6	3.0	

23.3.3 CORDIC engine

CORDIC, also known as Volder's algorithm, is a simple and efficient algorithm to calculate many different functions including trigonometric, hyperbolic, and logarithmic functions; and real and complex multiplications.

CORDIC is helpful in accelerating computations in various applications such as graphics in human-machine interfaces and field-oriented control. PowerQuad adds to the advantages of using CORDIC by accelerating the iterations per cycle.

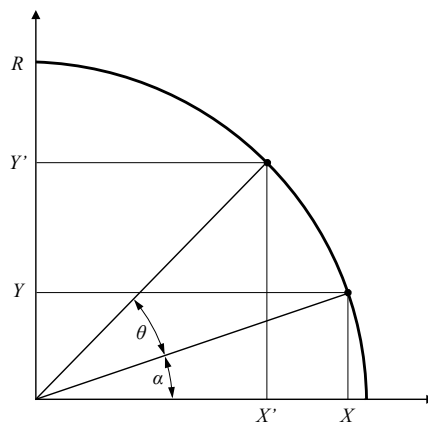


Figure 98. Rotation and vectoring mode of the CORDIC algorithm

In PowerQuad, the CORDIC engine can be directly used for inverse tangent and inverse hyperbolic tangent functions.

CORDIC hardware implements 2 iterations per cycle. The number of iterations is programmable. Although the iterations may be programmable, there is a tradeoff of speed vs. accuracy. CORDIC also allows use cases like calculating phase and magnitude from a real/imaginary pair simultaneously with 2 bits of convergence of output accuracy per cycle. In addition, the CORDIC engine stalls the CPU until the result is ready. Therefore, it is not necessary to poll for events or interrupts.

Functions that can use CORDIC:

- Arctan (x) : 12 cycles for 24-bit accuracy
- Arctanh (x) : 12 cycles for 24-bit accuracy

For PowerQuad CORDIC interface examples, see [CORDIC API examples](#).

23.3.3.1 Requirements

Input and output data support INT16 and INT32.

23.3.3.2 Performance

Table 367. Performance using CM33 alone versus with PowerQuad

Function	API	Using CM33 (s)	Using PowerQuad (s)	Improved Performance	Conditions
CORDIC	PQ_ArctanFixed	19.3E-6	486.7E-9	39.7	Frequency 150 MHz IAR 9.30.1 SDK 2.13.0 Running on Flash Optimization O3 Speed Note, the math functions running on CM33 are only supported on math.h
	PQ_ArctanhFixed	28.1E-6	453.3E-9	62.0	

23.3.4 Clocking

Enable the PowerQuad clock through an AHB Clock Control register in the system control of your chip.

See PowerQuad chip-specific clocking.

23.3.5 Interrupts

Events and Interrupts are enabled in the PowerQuad registers for AHB operations. PowerQuad informs the Cortex-M33 core, based on event or interrupt, upon completion of the chosen algorithm. Events and interrupts are enabled in the PowerQuad registers for AHB interface operations, they cannot be enabled for coprocessor interface operations. PowerQuad informs the CM33 core through the AHB bus, based on an event or interrupt, upon completion of a chosen algorithm.

23.4 External signals

This module has no external signals.

23.5 Initialization

See PowerQuad chip-specific initialization.

1. Enable the PowerQuad clock in the system configuration of your chip.
2. Reset the PowerQuad peripheral in the system configuration of your chip.
3. Enable events and interrupts.

23.6 Application information

This section explores relevant examples for both the coprocessor and AHB interface operations.

23.6.1 Construct coprocessor interface operations

MCR and MRC opcodes are constructed below for each of the supported algorithms in the coprocessor mechanism. The APIs in the SDK drivers internally use the MCR and MRC opcodes. Use these APIs, which are provided in the SDK drivers. To understand how the opcodes are constructed refer to the examples below. The API reference manual can be found in the docs folder of the downloadable SDK package provided at mcuxpresso.nxp.com.

MCR and MRC opcodes have several elements that are defined below. The table describes the inputs from the PowerQuad module for each element.

Table 368. MCR and MRC opcodes

Opcode	Parameters
coproc	CP_PQ
opc1	PQ_INV, PQ_LN, PQ_SQRT, PQ_INVSQRT, PQ_ETOX, PQ_ETONX, PQ_DIV, PQ_SIN, PQ_COS, PQ_BIQ0_CALC, PQ_BIQ1_CALC
Rt	(x) – the input value
CRn	PQ_FLOAT32, PQ_FIXEDPT, PQ_COMP0_ONLY, PQ_COMP1_ONLY
CRm	0
opc2	PQ_TRANS, PQ_TRIG, PQ_BIQUAD, PQ_TRANS, PQ_TRIG_FIXED, PQ_BIQUAD_FIXED

Table 369. Parameter descriptions

Parameters	Description
CP_PQ	PowerQuad Module
PQ_INV	Inverse function, $1/x$
PQ_LN	Natural logarithm, $\ln(x)$
PQ_SQRT	Square root, $\sqrt{x}$
PQ_INVSQRT	Inverse square root, $1/\sqrt{x}$
PQ_ETOX	Exponential function, e^x
PQ_ETONX	Inverse Exponential function, e^{-x}
PQ_DIV	Division, $x1/x2$
PQ_SIN	Sine, $\sin(x)$
PQ_COS	Cosine, $\cos(x)$

Table continues on the next page...

Table 369. Parameter descriptions (continued)

Parameters	Description
PQ_BIQ0_CALC	IIR filter for state 0, biquad0(x)
PQ_BIQ1_CALC	IIR filter for state 1, biquad1(x)
PQ_FLOAT32	Floating point data
PQ_FIXEDPT	Fixed-point data
PQ_COMP0_ONLY	Computation Block 0
PQ_COMP1_ONLY	Computation Block 1
PQ_TRANS	Transcendental engine used for floating-point data
PQ_TRIG	Trigonometric engine used for floating-point data
PQ_BIQUAD	Dual Biquad IIR engine used for floating-point data
PQ_TRANS_FIXED	Transcendental engine used for fixed-point data
PQ_TRIG_FIXED	Trigonometric engine used for fixed-point data
PQ_BIQUAD_FIXED	Dual Biquad IIR engine used for fixed-point data

23.6.1.1 Writing to PowerQuad

Coprocessing operations begin when the MCR operation is executed. Since MCR is an Arm operation, Arm assembler instructions are used directly in the application code. Each type of operation contains different elements to execute properly. See below the different operation types and how they are constructed. APIs have already been defined in the SDK for each of the supported mathematical functions. The following is only for reference.

Table 370. Commands

sin(x)
Floating point: __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRIG)
Fixed point: __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRIG_FIXED)
cos(x)
Floating point: __arm_mcr(CP_PQ, PQ_COS, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRIG)
Fixed point: __arm_mcr(CP_PQ, PQ_COS, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRIG_FIXED)
ln(x)
Floating point: __arm_mcr(CP_PQ, PQ_LN, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)
Fixed point:

Table continues on the next page...

Table 370. Commands (continued)

<code>__arm_mcr(CP_PQ, PQ_LN, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>
e^x
Floating point: <code>__arm_mcr(CP_PQ, PQ_ETOX, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)</code>
Fixed point: <code>__arm_mcr(CP_PQ, PQ_ETOX, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>
e^{-x}
Floating point: <code>__arm_mcr(CP_PQ, PQ_ETONX, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)</code>
Fixed point: <code>__arm_mcr(CP_PQ, PQ_ETONX, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>
$1/x$
Floating point: <code>__arm_mcr(CP_PQ, PQ_INV, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)</code>
Fixed point: <code>__arm_mcr(CP_PQ, PQ_INV, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>
$x1/x2$
The input for division is calculated with the concatenation of the divider and divisor. $x = (x2 \mid (x1 \ll 32));$
Floating point: <code>__arm_mcr(CP_PQ, PQ_FLOAT32 PQ_COMP0_ONLY, (uint64_t)(x), PQ_DIV)</code>
$1/\sqrt{x}$
Floating point: <code>__arm_mcr(CP_PQ, PQ_INVSQRT, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)</code>
Fixed point: <code>__arm_mcr(CP_PQ, PQ_INVSQRT, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>
$\sqrt{x}$
Floating point: <code>__arm_mcr(CP_PQ, PQ_SQRT, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_TRANS)</code>
Fixed point: <code>__arm_mcr(CP_PQ, PQ_SQRT, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_TRANS_FIXED)</code>

Table continues on the next page...

Table 370. Commands (continued)

biquad(x)
Floating point: __arm_mcr(CP_PQ, PQ_BIQ0_CALC, (uint32_t)(x), PQ_FLOAT32 PQ_COMP0_ONLY, 0, PQ_BIQUAD)
Fixed point: __arm_mcr(CP_PQ, PQ_BIQ0_CALC, (uint32_t)(x), PQ_FIXEDPT PQ_COMP0_ONLY, 0, PQ_BIQUAD_FIXED)

23.6.1.2 Reading from PowerQuad

To read back the result produced by the above algorithms, use the following MRC opcodes:

Table 371. Read back commands

Floating point:
__arm_mrc(CP_PQ, 0, PQ_FLOAT32 PQ_COMP0_ONLY, 0, 0)
__arm_mrc(CP_PQ, 1, PQ_FLOAT32 PQ_COMP0_ONLY, 0, 0)
Fixed point:
__arm_mrc(CP_PQ, 0, PQ_FIXEDPT PQ_COMP0_ONLY, 0, 0)
__arm_mrc(CP_PQ, 1, PQ_FIXEDPT PQ_COMP0_ONLY, 0, 0)

23.6.1.3 Coprocessor API example

23.6.1.3.1 Sinusoidal example

Using a sinusoidal equation as an example, let's build the coprocessing operation to accelerate the time it takes to execute, $\sin(3.0)$.

In this first example, the input and output values are floating-point numbers. The input value of 3.0 is in radians.

```
float input = 3.0;
float referenceOutput = 0.141120;
```

As mentioned above, the SDK already provides ready to use APIs for the different mathematical operations. In the case of a sine wave, there are three options that are compatible with CMSIS-DSP operations.

```
arm_sin_f32(float x)
arm_sin_q31(q31_t x)
arm_sin_q15(q15_t x)
```

For floating-point values, the `arm_sin_f32` option is used. This then uses the API from the PowerQuad drivers. The extra variable seen below is used for integer-float conversion during the operation.

```
float32_t arm_sin_f32(float32_t x)
{
    float tmp;
    PQ_SinF32(&x, &tmp);
    return tmp;
}
```

Inside the PowerQuad function, we can see that the floating-point variable is converted to an unsigned 32-bit integer, which is what the MCR operation uses to process the input. This is done by using a simple union structure that defines the equivalent

floating-point and integer variable. The blue arrow is the defined function call to the MCR operation constructed in [Writing to PowerQuad](#).

```
static inline void PQ_SinF32(float *pSrc, float *pDst)
{
    pq_float_t val;

    val.floatX = *pSrc;
    _pq_sin0(val.integerX);
    val.integerX = _pq_readAdd0();
    *pDst = val.floatX;
}

#define _pq_sin0(x) __arm_mcr(CP_PQ, PQ_SIN, (uint32_t)(x), PQ_FLOAT32 | PQ_COMP0_ONLY, 0, PQ_TRIG)
```

The blue arrow in the following image is the defined function to the MRC operation constructed in [Reading from PowerQuad](#). This function reads back the result and converts the value back to floating point before storing in the *pDst pointer.

```
static inline void PQ_SinF32(float *pSrc, float *pDst)
{
    pq_float_t val;

    val.floatX = *pSrc;
    _pq_sin0(val.integerX);
    val.integerX = _pq_readAdd0();
    *pDst = val.floatX;
}

#define _pq_readAdd0() __arm_mrc(CP_PQ, 1, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0)
```

23.6.1.3.2 Tangent example

[Writing to PowerQuad](#) demonstrates that the mapped coprocessing functions for trigonometric applications are sine and cosine. However, using simple math, and other mapped mathematical functions we can build the tangent equation.

Although there are several ways to come to the same tangent solution, note that the chosen equations affect the number of cycles required for the same result. Consider the following three equations:

$$\tan x = \frac{\sin x}{\cos x} \quad \tan x = \frac{\sin x}{\sqrt{1 - \sin^2 x}} \quad \tan x = \frac{\cos x}{\sqrt{1 - \cos^2 x}}$$

All three equations provide the same result. However, the two on the right take longer to execute than the one on the left because of the number of cycles required to execute them. Keep in mind that the two on the right are still much faster than doing the same sequence of events without PowerQuad enabled.

For the option on the left, sine and cosine can be done in parallel since these functions use 2 MACs each. Once the values are ready, perform a division. For the two options on the right, square either sine or cosine first, then do the subtraction. Next take the square root, then finally do a division. All of these steps consume many more cycles than the option on the left. Therefore, to realize the full potential of PowerQuad, choose the simplest route when building a new math function based on the available functions.

Since PowerQuad does not natively support the tangent function and we now know which functions to use for efficiency, we can build the code to run on PowerQuad. We want to run two coprocessor functions in parallel, so we must send the MCR opcodes. One uses computation block 0 and the second uses computation block 1. These are already provided in the SDK API, `fsl_powerquad.h` header file.

```
#define _pq_readAdd0() __arm_mrc(CP_PQ, 1, PQ_FLOAT32 | PQ_COMP0_ONLY, 0, 0)
```

For the purposes of this example, Calculate the tangent of pi/4, which gives us a result of 1, if done correctly. In addition, we use floating-point values as our input and output.

```
float input = (double)(MATH_PI/4.0f);

float finalResult;
```

We use the union structure to convert easily between floating-point and integer values, one for each function. Since the input is the same for the sine and cosine functions used to calculate the tangent, the same value is assigned to both structures.

```

pq_float_t val0, val1, div;

val0.floatX = input;
val1.floatX = input;

```

Then send the commands for each of the opcodes that run on PowerQuad, and the opcodes that read from the resulting registers.

```

_pq_sin0(val0.integerX);
_pq_cos1(val1.integerX);
val0.integerX = _pq_readAdd0();
val1.integerX = _pq_readAdd1();

```

Since the division is done using the intermediate integer values retrieved from both the sine and cosine, they can be used immediately to construct the input of the division and execute the opcode.

```

uint64_t inputDiv = (uint64_t)(val1.integerX) | ((uint64_t)(val0.integerX) << 32U);
_pq_div0(inputDiv);

```

Then read from the resulting register and convert to floating point for the final result.

```

div.integerX = _pq_readMult0();
finalResult = X1.floatX;

```

At this point, the result is 1 for a correctly executed tangent function.

23.6.1.3.3 Biquad cascade example

Having a dual biquad engine allows us to use two different biquad states to execute a chain of second order biquad stages. Where each stage has the same format as described in [Dual Biquad IIR](#). For this example, a 3-stage biquad cascade is shown.

A structure is created to initialize each of the coefficients in each biquad stage. The values vary from application to application, in this case we have the following data. Initialize the COMPREG register with zero, because the register is used for intermediate operations. Importantly, if COMREG has other values, those values can corrupt the result of the operation.

```

pq_biquad_state_t state[3] = {
    {
        .compreg = 0,
        .param =
        {
            .v_n_1 = 0.0f,
            .v_n   = 0.0f,
            .a_1   = 0.2514f,
            .a_2   = 0.5028f,
            .b_0   = 0.2514f,
            .b_1   = -0.1712f,
            .b_2   = 0.1768f,
        },
    },
    {
        .compreg = 0,
        .param =
        {
            .v_n_1 = 0.0f,
            .v_n   = 0.0f,
            .a_1   = 0.2514f,
            .a_2   = 0.5028f,
            .b_0   = 0.2514f,
            .b_1   = -0.1712f,
            .b_2   = 0.1768f,
        },
    },
    {
        .compreg = 0,
        .param =
        {
            .v_n_1 = 0.0f,
            .v_n   = 0.0f,
            .a_1   = 0.2514f,
            .a_2   = 0.5028f,
            .b_0   = 0.2514f,
            .b_1   = -0.1712f,
            .b_2   = 0.1768f,
        },
    },
};

```

The data for the biquad in this example is:

```
float dataForBiquad[16] = {1024.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

There are two possible routes to take when executing a cascaded biquad. The first is to calculate all results at once. The second option is to do incremental calculations.

For the first option, the API provided is:

```

pq_biquad_cascade_df2_instance instance;
PQ_BiquadCascadeDf2Init(&instance, stage, state);
PQ_BiquadCascadeDf2F32(&instance, dataForBiquad, biquadResult, 15);

```

The PQ_BiquadCascadeDf2Init function simply assigns the values for the number of stages in this cascade sequence and the structure of the coefficients for each of those stages. The PQ_BiquadCascadeDf2F32 function takes the input data for the filter and the block size, the result is stored in the output pointer provided.

Internally this function restores and backups the intermediate results produced from each of the states, and performs a biquad operation on each state. This function is done multiple times until all stages have been processed. The Restore and BackUp functions use the GPREG and COMPREG registers for intermediate values.

```
if((stage & 0x01U) == 0x01U)
{
    PQ_BiquadRestoreInternalState(POWERQUAD, 0, states);
    PQ_VectorBiquadDf2F32(pSrc, pDst, (int32_t)blockSize);
    PQ_BiquadBackUpInternalState(POWERQUAD, 0, states);
    states++;
    stage--;
}

while(stage > 0U)
{
    PQ_BiquadRestoreInternalState(POWERQUAD, 1, states);
    states++;
    PQ_BiquadRestoreInternalState(POWERQUAD, 0, states);

    PQ_VectorBiquadCascadeDf2F32(pDst, pDst, (int32_t)blockSize);

    states--;
    PQ_BiquadBackUpInternalState(POWERQUAD, 1, states);
    states++;
    PQ_BiquadBackUpInternalState(POWERQUAD, 0, states);

    states++;
    stage -= 2U;
}
```

When the stage is higher than 0, the operation executed is the PQ_VectorBiquadCascadeDf2F32. This function uses both of the Biquad states. As described in [Dual Biquad IIR](#), a cascade biquad operation uses state 1, then state 0 provides the result. If the stage is 1, then a single biquad operation uses only biquad state 0.

For the incremental option, the function is done with the following APIs:

```
PQ_BiquadCascadeDf2Init(&instance, stage, state);

PQ_BiquadCascadeDf2F32(&instance, dataForBiquad, biquadResult, 8);
PQ_BiquadCascadeDf2F32(&instance, &dataForBiquad[8], &biquadResult[8], 7);
```

The function used for the incremental option is the same as the one used in the full calculation. However, the block size is smaller and partitioned in two segments, with the second function starting where the last one left off. This partitioning is beneficial when you must separate operations and use the PowerQuad for another purpose. You must save the intermediate results to return and run the biquad operation without corrupting the data.

For more details on the internal operations of the biquad filter, review the example code provided in the SDK.

23.6.2 Construct AHB interface operations

Below are some examples demonstrating how registers are configured for some of the supported algorithms in the AHB mechanism. The APIs used in the SDK drivers internally use these configurations, so you do not need to use these configurations directly. To understand how to configure the registers, refer to the examples below. The API reference manual can be found in the docs folder of the downloadable SDK package provided at mcuxpresso.nxp.com.

Understanding the math behind the operation, which needs to be accelerated, facilitates the construction of the AHB interface operation. First, it is necessary to know which algorithm to use, before selecting the correct engine and the proper configurations. In addition to the settings for the algorithm chosen, you must write to the **CONTROL** register to enable the execution. Writing to the CONTROL register indicates to PowerQuad that it can begin processing the information, therefore write to the CONTROL register last. The **CONTROL** register is composed of three elements:

INST_BUSY	0: Not Busy 1: Busy

Table continues on the next page...

Table continued from the previous page...

DECODE_MACHINE	1: CP_MTX – Matrix Engine 2: CP_FFT – Transform Engine 3: CP_FIR – Filter Engine 5: CP_CORDIC – Cordic Engine
DECODE_OPCODE	Opcode specific to decode_machine: Matrix: PQ_MTX_SCALE, PQ_MTX_MULT, PQ_MTX_ADD, PQ_MTX_INV, PQ_MTX_PROD, PQ_MTX_SUB, PQ_MTX_DOTP, PQ_MTX_TRANS Transform: PQ_TRANS_CFFT, PQ_TRANS_IFFT, PQ_TRANS_CDCT, PQ_TRANS_IDCT, PQ_TRANS_RFFT, PQ_TRANS_RDCT Filter: PQ_FIR_FIR, PQ_FIR_CONVOLUTION, PQ_FIR_CORRELATION, PQ_FIR_INCREMENTAL Cordic : CORDIC_ARCTAN, CORDIC_ARCTANH, CORDIC_ITER, CORDIC_MIU, CORDIC_T

The following sections describe different AHB interface operations and how they are constructed individually. Each type of operation can use the same registers but in a different manner.

23.6.2.1 Matrix API example

A simple matrix example is the addition of two matrices. Matrix data is stored in memory row-by-row, arranged like standard C arrays.

So, if two 2x2 integer matrices A and B are:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Then the input data is stored in memory arrays as follows:

- int data_A15[4] = {1, 2, 3, 4};
- int data_B15[4] = {5, 6, 7, 8};

In the same manner, we can expect that our resulting matrix has the same format. In matrix addition, the size of the resulting matrix is the same size as the input matrices.

- int Result[4] = {0};

The SDK drivers provide a structure to help organize the matrices. This structure contains key elements of the matrix. It has the number of rows, columns, and the pointer to where the matrix data is stored.

```
typedef struct
{
    uint16_t numRows;    /**< number of rows of the matrix. */
    uint16_t numCols;    /**< number of columns of the matrix. */
    q15_t *pData;        /**< points to the data of the matrix. */
    arm_matrix_instance_q15;
}
```

Using this structure the three matrices that are needed can be initialized. Since the rows and columns are the same for all of them, we can reuse the variable. The initialization shown below simply assigns the value of the columns and rows to their respective variable in the structure, and points the matrix data to the pointer of the structure.

```
srcRows    = MATRIX_ROWS;
srcColumns = MATRIX_COL;
arm_mat_init_q15(&addMatrixA15, srcRows, srcColumns, A15);
arm_mat_init_q15(&addMatrixB15, srcRows, srcColumns, B15);
arm_mat_init_q15(&addMatrixR, srcRows, srcColumns, Result);
```

For compatibility, we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled. It uses the three matrices that are needed for the addition operation.

```
arm_mat_add_q15(&addMatrixA15, &addMatrixB15, &addMatrixR);
```

This example uses fixed-point 16-bit inputs and outputs. For this we must configure the [OUTFORMAT](#), [INAFORMAT](#), [INBFORMAT](#), and [TMPFORMAT](#) registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_FIX16_CONFIG
POWERQUAD->OUTFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_16Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_Float << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->TMPBASE   = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner. No scaler is necessary. The output format is fixed 16-bit data and the internal engine format is handled using floating point. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

Next, it is necessary to define the value of LENGTH. As described in [Requirements](#), LENGTH is remapped depending on the operation. In this case, we need the number of columns for matrix B15 and both the columns and rows for matrix A15.

- LENGTH[31:0] = Cols_B[20:16] | Cols_A[12:8] | Rows_A[4:0]

So, LENGTH = 0x40404;

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
arm_status arm_mat_add_q15(const arm_matrix_instance_q15 *pSrcA,
    const arm_matrix_instance_q15 *pSrcB,
    arm_matrix_instance_q15 *pDst)
{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    arm_status status;
    uint32_t length;
    PQ_SET_FIX16_CONFIG;

    length = POWERQUAD_MAKE_MATRIX_LEN(pSrcA->numRows, pSrcA->numCols, pSrcB->numCols);
    PQ_MatrixAddition(POWERQUAD, length, pSrcA->pData, pSrcB->pData, pDst->pData);

    /* Wait for the completion */
    PQ_WaitDone(POWERQUAD);

    status = ARM_MATH_SUCCESS;

    /* Return to application */
    return status;
}
```

The API that configures the registers for matrix addition is as follows:

- PQ_MatrixAddition(POWERQUAD, length, data_A, data_B, result);

Table 372. PQ_MatrixAddition parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
length	Composed of the length of the M2 column, M1 row and column. When M1 and M2 are matrix A and B respectively. Length = (length[23:16] = M2 columns length [15:8] = M1 columns length [7:0] = M1 rows);
data_A	Base address where the first matrix is located
data_B	Base address where the second matrix is located
result	Base address where the result matrix is expected to be located

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of Result;
- PowerQuad[INABASE] = address of A15;
- PowerQuad[INBBASE] = address of B15;
- PowerQuad[LENGTH] = length;
- PowerQuad[CONTROL] = (CP_MTX << 4U) | PQ_MTX_ADD;

Lastly, it is important that the **EVENTEN** register is enabled to trigger when execution is completed. Events and Interrupts are generally set up during initialization of PowerQuad.

```

arm_status arm_mat_add_q15(const arm_matrix_instance_q15 *pSrcA,
                          const arm_matrix_instance_q15 *pSrcB,
                          arm_matrix_instance_q15 *pDst)
{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    arm_status status;
    uint32_t length;
    PQ_SET_FIX16_CONFIG;

    length = POWERQUAD_MAKE_MATRIX_LEN(pSrcA->numRows, pSrcA->numCols, pSrcB->numCols);
    PQ_MatrixAddition(POWERQUAD, length, pSrcA->pData, pSrcB->pData, pDst->pData);

    /* Wait for the completion */
    PQ_WaitDone(POWERQUAD);

    status = ARM_MATH_SUCCESS;

    /* Return to application */
    return status;
}

```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```

/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
    __WFE();
}

```


At this point, the result can be read from the Result matrix that we initialized at the beginning of the example.

23.6.2.2 FFT API example

The Complex Fast Fourier Transform(CFFT) is a different AHB interface example that uses the transform engine. There are certain requirements and details that must be considered when using the FFT engine. See [Requirements](#) for more details.

This example uses fixed 32-bit format data. The input data is taken by using PowerQuad functions and using sine and cosine waves. The input data can be generated from many different sources. For ease of use, the example uses the PowerQuad module.

```
/* Two full period sin wave. */
for (uint32_t i = 0; i < CFFT_INPUT_LEN / 2; i++)
{
    input[i * 2]      = (arm_sin_q31(i * (0x80000000 / (CFFT_INPUT_LEN / 2))) / 2) >> 5;
    input[i * 2 + 1] = (arm_cos_q31(i * (0x80000000 / (CFFT_INPUT_LEN / 2))) / 2) >> 5;
}
```

Once the input data is ready, PowerQuad must be configured accordingly. For compatibility we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled.

```
arm_cfft_q31(&instance, input, 0, 1);
```

This example uses fixed-point 32-bit inputs and outputs. For this we must configure the [OUTFORMAT](#), [INAFORMAT](#), [INBFORMAT](#), and [TMPFORMAT](#) registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_FFT_Q31_CONFIG
POWERQUAD->OUTFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_32Bit; \
POWERQUAD->TMPBASE   = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner. There is no scaler necessary. The output format is fixed 32-bit data and the internal engine format is handled using fixed point as well. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
void arm_cfft_q31(const arm_cfft_instance_q31 *S, q31_t *p1, uint8_t ifftFlag, uint8_t bitReverseFlag)
{
    assert(bitReverseFlag == 1U);

    q31_t *pIn      = p1;
    q31_t *pOut      = p1;
    uint32_t length = S->fftLen;

    PQ_SET_FFT_Q31_CONFIG;

    if (ifftFlag == 1U)
    {
        PQ_TransformIFFT(POWERQUAD, length, pIn, pOut);
    }
    else
    {
        PQ_TransformCFFT(POWERQUAD, length, pIn, pOut);
    }

    PQ_WaitDone(POWERQUAD);
}
```

The API that configures the registers for CFFT is as follows:

- `PQ_TransformCFFT(POWERQUAD, N, inputData, cfftResult);`

Table 373. PQ_TransformCFFT parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
N	Length for FFT and DCT, the supported lengths are 16, 32, 64, 128, 256, and 512
inputData	Base address where the input array is located
cfftResult	Base address where the result matrix is expected to be located

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of Result;
- PowerQuad[INABASE] = address of input;
- PowerQuad[LENGTH] = N;
- PowerQuad[CONTROL] = (CP_FFT << 4U) | PQ_TRANS_CFFT;

Lastly, it is important that the [EVENTEN](#) register is enabled to trigger when execution is completed. Events and Interrupts are generally set up in the initialization of the PowerQuad.

```
void arm_cfft_q31(const arm_cfft_instance_q31 *S, q31_t *p1, uint8_t ifftFlag, uint8_t bitReverseFlag)
{
    assert(bitReverseFlag == 1U);

    q31_t *pIn    = p1;
    q31_t *pOut    = p1;
    uint32_t length = S->fftLen;

    PQ_SET_FFT_Q31_CONFIG;

    if (ifftFlag == 1U)
    {
        PQ_TransformIFFT(POWERQUAD, length, pIn, pOut);
    }
    else
    {
        PQ_TransformCFFT(POWERQUAD, length, pIn, pOut);
    }

    PQ_WaitDone(POWERQUAD);
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
    __WFE();
}
```

At this point, the result can be read from the Result matrix that we initialized at the beginning of the example.

23.6.2.3 FIR API example

The FIR engine can compute different processes. One of these signal processing mechanisms is convolution. Convolution is one of the most important and fundamental concepts in signal processing and analysis.

Shown here is the mathematical definition of convolution in a discrete time domain.

$$pDst[n] = pSrcA[n] * pSrcB[n] = \sum_{k=-\infty}^{\infty} pSrcA[k] \bullet pSrcB[n-k]$$

Where $pSrcA[n]$ is the input signal, $pSrcB[n]$ is the impulse response, and $y[n]$ is the output.

For this example, the data used is defined as floating-point values, however this example uses 32-bit fixed-point data.

```
static const float pSrcA[] = {0.01f, 0.02f, 0.03f, 0.04f, 0.05f};
static const float pSrcB[] = {0.02f, 0.04f, 0.06f, 0.08f, 0.10f};
```

The drivers in the SDK provide an easy API to convert from floating-point to fixed-point 32-bit data.

```
inputA[i] = FLOAT_2_Q31(pSrcA[i]);
inputB[i] = FLOAT_2_Q31(pSrcB[i]);
```

Once the input data is defined, the PowerQuad must be configured accordingly. For compatibility, we can use the CMSIS-DSP function call directly, which calls the PowerQuad APIs when enabled.

```
arm_conv_q31(inputA, CONV_A_LEN, inputB, CONV_B_LEN, result);
```

The data uses fixed-point 32-bit inputs and outputs. For this, we must configure the **OUTFORMAT**, **INAFORMAT**, **INBFORMAT**, and **TMPFORMAT** registers. The drivers facilitate this configuration by providing a defined framework.

```
#define PQ_SET_Q31_CONFIG
POWERQUAD->OUTFORMAT = ((uint32_t)(-31) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->INAFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->INBFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_32Bit << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->TMPFORMAT = ((uint32_t)(0) << 8U) | ((uint32_t)kPQ_Float << 4U) | (uint32_t)kPQ_Float; \
POWERQUAD->TMPBASE = 0xE0000000U
```

All of the FORMAT registers are configured in the same manner, except OUTFORMAT and TMPFORMAT. The output format is scaled for 32 bits. In addition, the output and the input formats are fixed 32-bit data. The internal engine format is handled using floating point. The temporary format remains floating point throughout its use. Also, the TMPBASE is defined as 0xE000_0000, which is the reference for private RAM.

As described above, the CMSIS-DSP function, internally calls the PowerQuad function.

```
void arm_conv_q31(const q31_t *pSrcA, uint32_t srcALen, const q31_t *pSrcB, uint32_t srcBLen, q31_t *pDst)
{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    PQ_SET_Q31_CONFIG;

    PQ_FIR(POWERQUAD, pSrcA, (int32_t)srcALen, pSrcB, (int32_t)srcBLen, pDst, PQ_FIR_CONVOLUTION);
    PQ_WaitDone(POWERQUAD);
}
```

The API that configures the registers for FIR is as follows:

- PQ_FIR(POWERQUAD, pSrcA, srcALen, pSrcB, srcBLen, pDst, op_type);

Table 374. PQ_FIR parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
pSrcA	Base address where the input signal is located
srcALen	Length of the input signal
pSrcB	Base address where the impulse response signal is located
srcBLen	Length of the impulse response
pDst	Base address where the result is expected to be located
op_type	Filter operation that is the computer

The register configurations are set as shown below:

- PowerQuad[OUTBASE] = address of pDst;

- PowerQuad[INABASE] = address of pSrcA;
- PowerQuad[INBBASE] = address of pSrcB;
- PowerQuad[LENGTH] = (srcBLen << 16) | srcALen;
- PowerQuad[CONTROL] = (CP_FIR << 4U) | PQ_FIR_CONVOLUTION;

Enable the **EVENTEN** register to trigger when execution is completed. Set up Events and Interrupts during PowerQuad initialization.

```
void arm_conv_q31(const q31_t *pSrcA, uint32_t srcALen, const q31_t *pSrcB, uint32_t srcBLen, q31_t *pDst)
{
    assert(NULL != pSrcA);
    assert(NULL != pSrcB);
    assert(NULL != pDst);

    PQ_SET_Q31_CONFIG;

    PQ_FIR(POWERQUAD, pSrcA, (int32_t)srcALen, pSrcB, (int32_t)srcBLen, pDst, PQ_FIR_CONVOLUTION);
    PQ_WaitDone(POWERQUAD);
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
    _WFE();
}
```

The results are read from the result pointer, initialized in the configuration registers.

23.6.3 CORDIC API examples

The inverse trigonometric functions are implemented using a CORDIC. PowerQuad implements two-iteration per cycle CORDIC. The return value is in the range of -2^{27} to 2^{27} , which represents $-\pi$ to π . The sum of data_A and data_B should not exceed the range of INT32.

This example demonstrates the arctangent of 0.5. Larger input numbers gets higher output accuracy, the result of PQ_ArctanFixed(POWERQUAD, 200000, 100000, kPQ_Iteration_24) is more accurate than PQ_ArctanFixed(POWERQUAD, 2, 1, kPQ_Iteration_24).

The PowerQuad functions can be used directly as there is not support for the arctangent function in the CMSIS-DSP library. The input and output data are handled as 32-bit fixed-point data.

```
PQ_ArctanFixed(DEMO_POWERQUAD, 20000000, 10000000, kPQ_Iteration_24);
```

In this example, the FORMAT registers are not used. Instead we directly provide the inputs to the CORDIC registers of the Powerquad. **CORDIC_X** contains the dividend and **CORDIC_Y** contains the divisor. **CORDIC_Z** contains the result once the execution has finished. To initialize this register write 0.

```
int32_t pq_arctan(int32_t x, int32_t y)
{
    int32_t result = PQ_ArctanFixed(DEMO_POWERQUAD, x, y, kPQ_Iteration_24);
    result += (y < 0 && x < 0) ? 0xf8000000;
    result += (y >= 0 && x < 0) ? 0x08000000;

    return (result);
}
```

The API that configures the registers for arctan using CORDIC is as follows:

- PQ_ArctanFixed(POWERQUAD, data_A, data_B, iteration);

Table 375. PQ_ArctanFixed parameters

Parameter	Description
POWERQUAD	Base address for the PowerQuad hardware module in the device
data_A	Base address where the value of the opposite side
data_B	Base address where the value of the adjacent side
iteration	Value of iterations for rotations in the calculation can be 8, 16, or 24

The register configurations are set as shown below:

- PowerQuad[CORDIC_X] = data_A;
- PowerQuad[CORDIC_Y] = data_B;
- PowerQuad[CORDIC_Z] = 0;
- PowerQuad[CONTROL] = (CP_CORDIC << 4U) | CORDIC_ARCTAN | CORDIC_ITER(iteration);

Enable the [EVENTEN](#) register to trigger when execution is completed. Set up Events and Interrupts during PowerQuad initialization.

```
int32_t PQ_ArctanFixed(POWERQUAD_Type *base, int32_t x, int32_t y, pq_cordic_iter_t iteration)
{
    base->CORDIC_X = (uint32_t)x;
    base->CORDIC_Y = (uint32_t)y;
    base->CORDIC_Z = 0U;
    base->CONTROL = (CP_CORDIC << 4U) | CORDIC_ARCTAN | CORDIC_ITER(iteration);

    PQ_WaitDone(base);
    return (int32_t)base->CORDIC_Z;
}
```

The PQ_WaitDone API polls the INST_BUSY flag to know when the result is ready. During the busy period, the main core is sleeping and wakes once the event arrives. Use the event trigger to save cycles instead of an interrupt trigger that services ISR.

```
/* wait for the completion */
while ((base->CONTROL & INST_BUSY) == INST_BUSY)
{
    __WFE();
}
```

The result can be read from the [CORDIC Input Z \(CORDIC_Z\)](#) register.

23.7 Memory map and register definition

This section includes the PowerQuad module memory map and detailed descriptions of all registers.

NOTE

This module does not generate transfer errors when accessing unimplemented registers in its register space.

23.7.1 PowerQuad register descriptions

23.7.1.1 PowerQuad memory map

POWERQUAD0 base address: 4020_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Output Base (OUTBASE)	32	RW	0000_0000h
4h	Output Format (OUTFORMAT)	32	RW	0000_0000h
8h	Temporary Base (TMPBASE)	32	RW	0000_0000h
Ch	Temporary Format (TMPFORMAT)	32	RW	0000_0000h
10h	Input A Base (INABASE)	32	RW	0000_0000h
14h	Input A Format (INAFORMAT)	32	RW	0000_0000h
18h	Input B Base (INBBASE)	32	RW	0000_0000h
1Ch	Input B Format (INBFORMAT)	32	RW	0000_0000h
100h	Control (CONTROL)	32	RW	0000_0000h
104h	Length (LENGTH)	32	RW	0000_0000h
108h	Coprocessor Prescale (CPPRE)	32	RW	0000_0000h
10Ch	Miscellaneous (MISC)	32	RW	0000_0000h
110h	Cursory (CURSORY)	32	RW	0000_0000h
180h	CORDIC Input X (CORDIC_X)	32	RW	0000_0000h
184h	CORDIC Input Y (CORDIC_Y)	32	RW	0000_0000h
188h	CORDIC Input Z (CORDIC_Z)	32	RW	0000_0000h
18Ch	Error Status (ERRSTAT)	32	RW	0000_0000h
190h	Interrupt Enable (INTREN)	32	RW	0000_0000h
194h	Event Enable (EVENTEN)	32	RW	0000_0000h
198h	Interrupt Status (INTRSTAT)	32	RW	0000_0000h
200h - 23Ch	General Purpose Register Bank n (GPREG0 - GPREG15)	32	RW	0000_0000h
240h - 25Ch	Compute Register Bank n (COMPREG0 - COMPREG7)	32	RW	0000_0000h

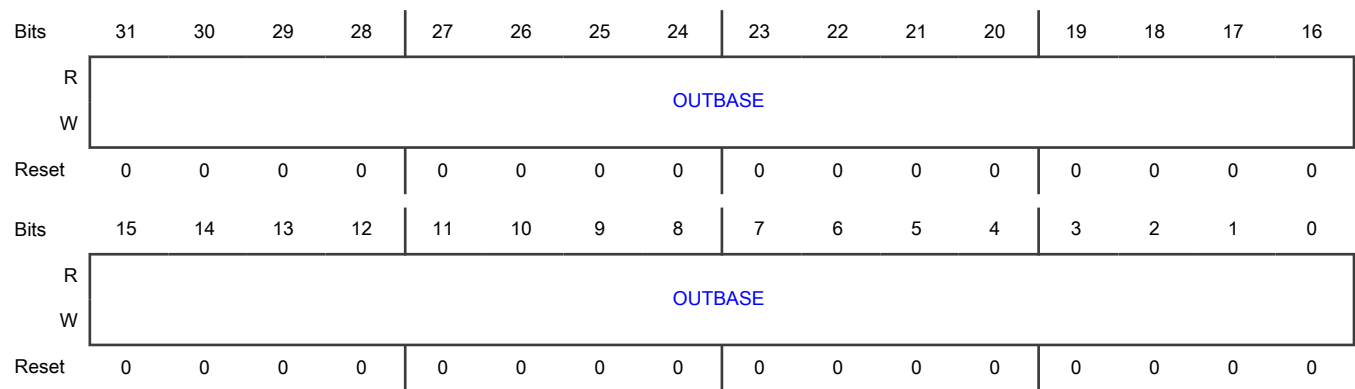
23.7.1.2 Output Base (OUTBASE)

Offset

Register	Offset
OUTBASE	0h

Function

Contains the base address, which points to a location in memory where you want to store the result of the operation.

Diagram**Fields**

Field	Function
31-0 OUTBASE	Output Region Base Address 32-bit register that contains the base address where the result is stored upon completion of the AHB operation.

23.7.1.3 Output Format (OUTFORMAT)**Offset**

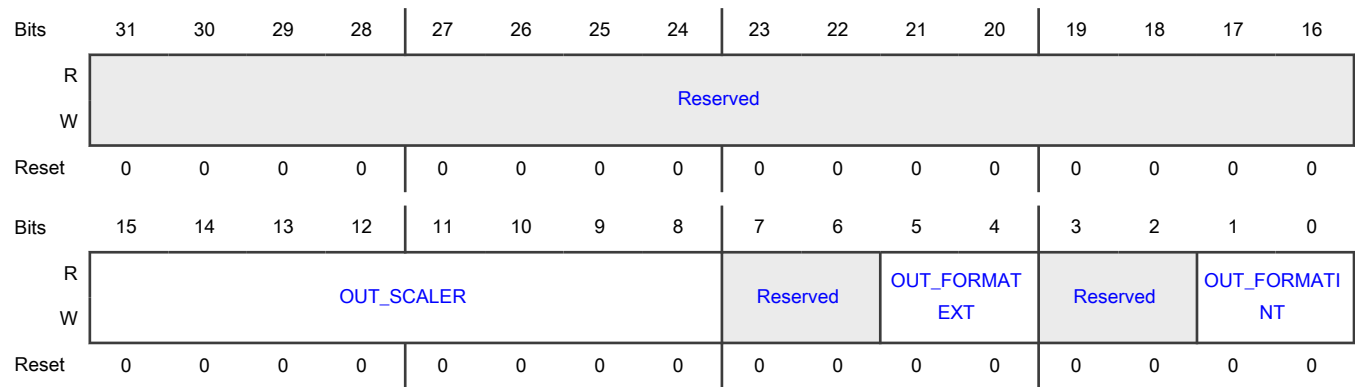
Register	Offset
OUTFORMAT	4h

Function

Used to define the format of the data inside the engine and outside the engine in the OUTBASE location. The scaling factor can also be chosen for fixed-point output.

NOTE

The format definitions can only be used when data regions are in AHB space. Data written to or read from private RAM space does not undergo conversion.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-8 OUT_SCALER	8-bit Scaling Value for Result Data Used when converting from floating=point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 OUT_FORMAT EXT	Output External Format Format of result data, converted when exiting the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 OUT_FORMATI NT	Output Internal Format Format of result data during active operation within the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved

23.7.1.4 Temporary Base (TMPBASE)

Offset

Register	Offset
TMPBASE	8h

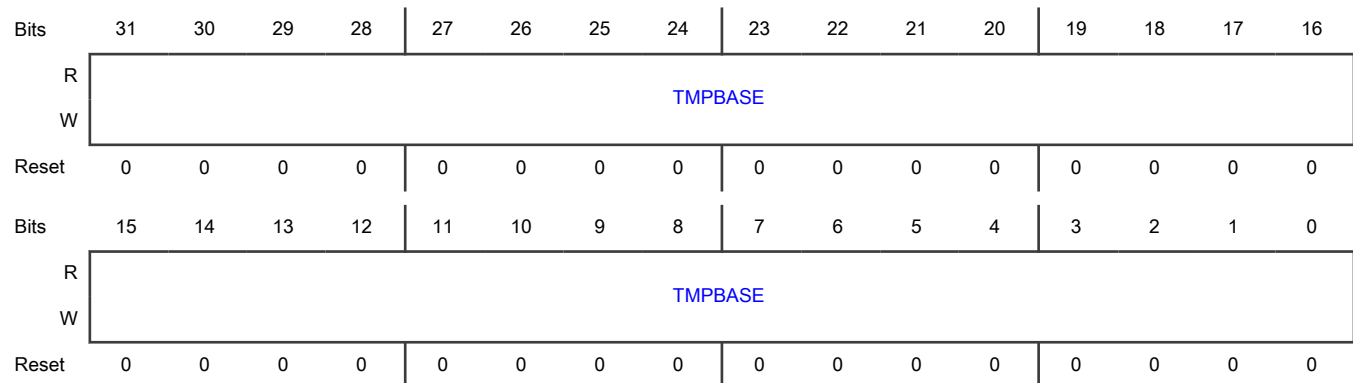
Function

Used to define the location in which temporary data can be stored while operations are being executed. PowerQuad uses its own private RAM for this purpose. The base address for this area is 0xE000_0000.

NOTE

Inverse matrix operations do not use the private RAM area. The TMPBASE address must point to a scratch area in AHB space.

Diagram



Fields

Field	Function
31-0	Base Address for the Temporary Region
TMPBASE	Contains the base address for the temporary data stored during some operations.

23.7.1.5 Temporary Format (TMPFORMAT)

Offset

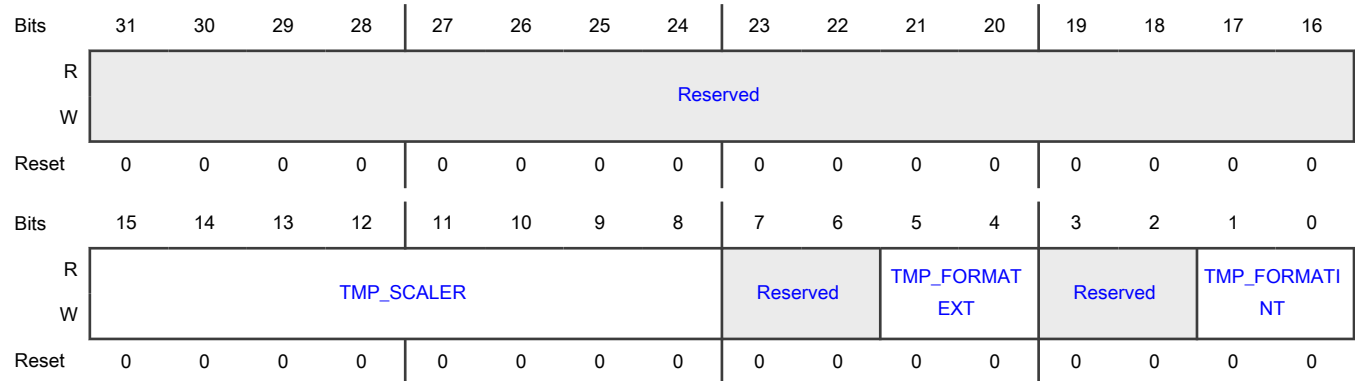
Register	Offset
TMPFORMAT	Ch

Function

Used to define the format of the data inside the engine and outside the engine in the TMPBASE location. The scaling factor can also be chosen for fixed-point output.

NOTE

The format definitions can only be used when data regions are in AHB space. Data written to or read from private RAM space does not undergo conversion.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-8 TMP_SCALER	Scaling Value for Temporary Data. 8-bit scaling value used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 TMP_FORMAT EXT	Temporary External Format Format of temporary data, converted when exiting the engine. NOTE This conversion can only be used for temporary data in AHB space, it cannot be used for temporary data stored in private RAM. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0	Temporary Internal Format Format of temporary data during active operation within the engine.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TMP_FORMATI NT	00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved

23.7.1.6 Input A Base (INABASE)

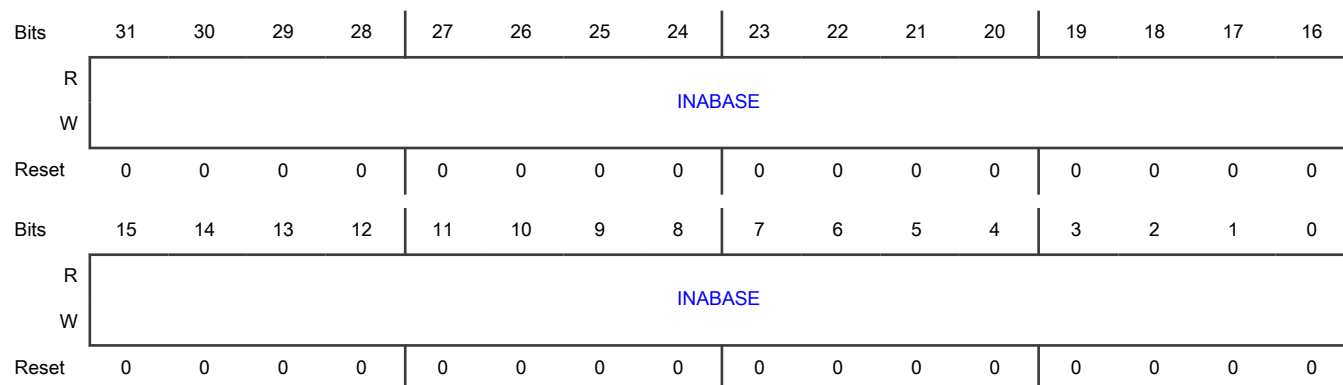
Offset

Register	Offset
INABASE	10h

Function

Contains the base address that points to a location in memory where you store operand A.

Diagram



Fields

Field	Function
31-0 INABASE	Input A Base Contains the base address where operand A is stored prior to executing the AHB operation.

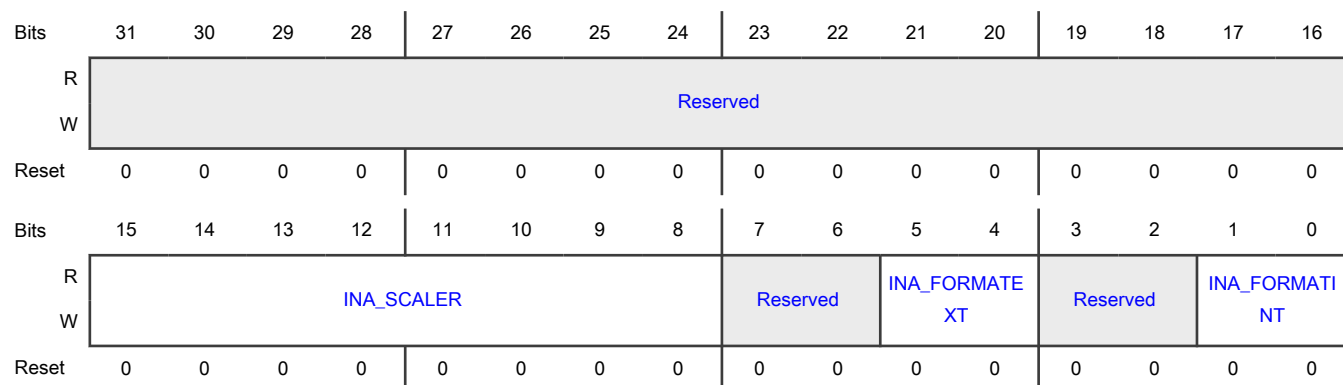
23.7.1.7 Input A Format (INAFORMAT)

Offset

Register	Offset
INAFORMAT	14h

Function

Used to define the format of the data inside the engine and outside the engine in the INABASE location. The scaling factor can also be chosen for fixed-point output.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-8 INA_SCALER	Input A Scaler Value 8-bit scaling value for operand A data. Used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 INA_FORMATE XT	Input A External Format Original format of operand A data prior to entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 INA_FORMATI NT	Input A Internal Format Format of operand A data, converted when entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - F32 32-bit floating-point format
	11b - Reserved

23.7.1.8 Input B Base (INBBASE)

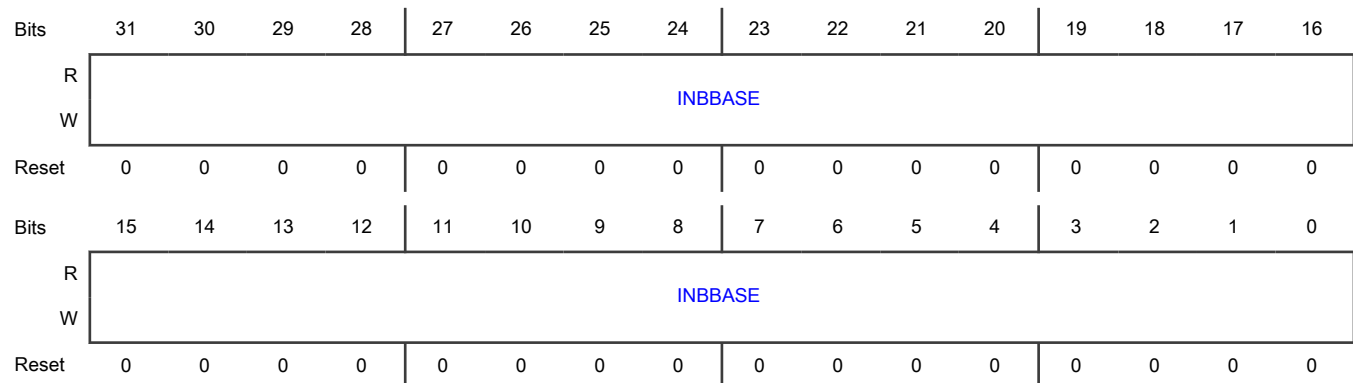
Offset

Register	Offset
INBBASE	18h

Function

Contains the base address that points to a location in memory where you store operand B.

Diagram



Fields

Field	Function
31-0	Input B Base
INBBASE	Contains the base address for where operand B is stored prior to executing the AHB operation.

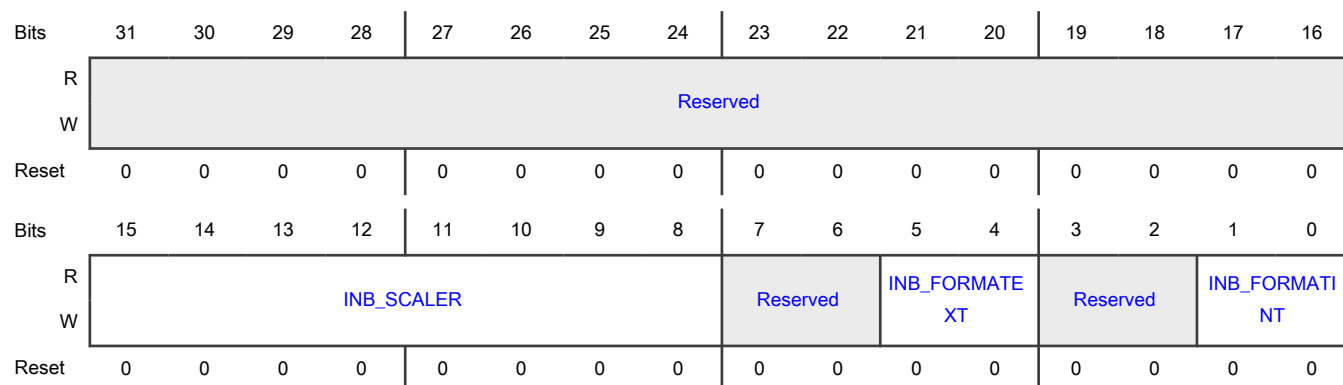
23.7.1.9 Input B Format (INBFORMAT)

Offset

Register	Offset
INBFORMAT	1Ch

Function

Used to define the format of the data inside the engine and outside the engine in the INBBASE location. The scaling factor can also be chosen for fixed-point output.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-8 INB_SCALER	Input B Scaler 8-bit scaling value for operand B data. Used when converting from floating-point to fixed-point Q15 or Q31 formats.
7-6 —	Reserved
5-4 INB_FORMATE XT	Input B External Format Original format of operand B data prior to entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer 10b - F32 32-bit floating-point format 11b - Reserved
3-2 —	Reserved
1-0 INB_FORMATI NT	Input B Internal Format Format of operand B data conversion when entering the engine. 00b - Q15 16-bit fixed-point integer 01b - Q31 32-bit fixed-point integer

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - F32 32-bit floating-point format
	11b - Reserved

23.7.1.10 Control (CONTROL)

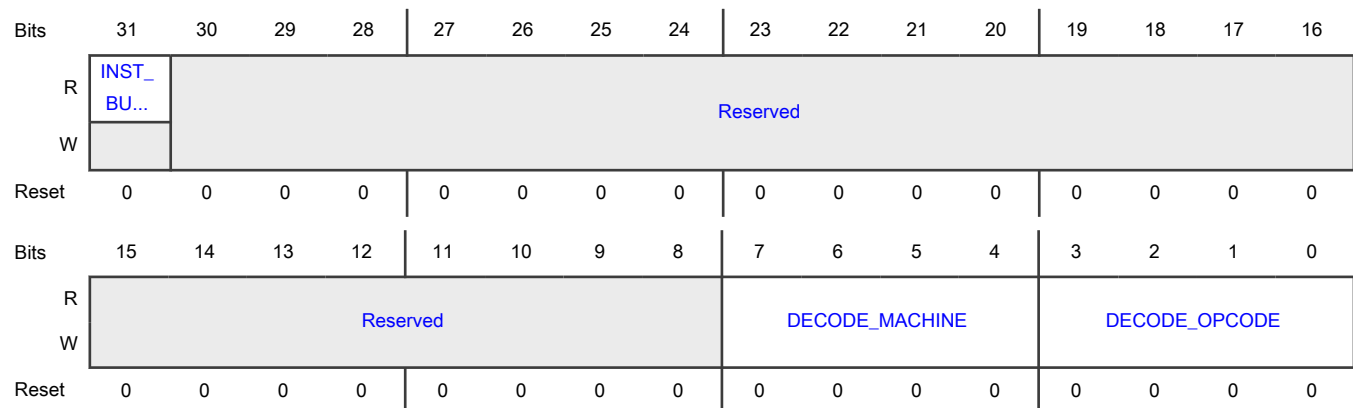
Offset

Register	Offset
CONTROL	100h

Function

Used for AHB operations. You can define the executable operations. When writing to the CONTROL register, the operation begins. A status bit is available for polling to determine if PowerQuad is busy.

Diagram



Fields

Field	Function
31 INST_BUSY	Instruction Busy PowerQuad is busy performing requested operations. 0b - Not busy 1b - Busy
30-8 —	Reserved
7-4	Decode Machine Computation engine to be used for operation.

Table continues on the next page...

Field	Function																																																																	
DECODE_MAC HINE	0000b - Coprocessor 0001b - Matrix engine 0010b - Transform engine 0011b - Filter engine 0100b,0110b-1111b - Reserved 0101b - CORDIC engine																																																																	
3-0 DECODE_OPCODE	Decode Opcode Defines what operation that the engine executes. Each engine can of different operations. The table below describes possible operations for each engine: Table 376. Decode Opcodes <table><tr><th></th><th>CP_MTX</th><th>CP_FFT</th><th>CP_FIR</th><th>CP_CORDIC</th></tr><tr><td>0000b</td><td>-</td><td>PQ_TRANS_CFFT</td><td>PQ_FIR_FIR</td><td>-</td></tr><tr><td>0001b</td><td>PQ_MTX_SCALE</td><td>PQ_TRANS_IFFT</td><td>PQ_FIR_CONVOLUTION</td><td>CORDIC_ARCTAN</td></tr><tr><td>0010b</td><td>PQ_MTX_MULT</td><td>PQ_TRANS_CDCT</td><td>PQ_FIR_CORRELATION</td><td>-</td></tr><tr><td>0011b</td><td>PQ_MTX_ADD</td><td>PQ_TRANS_IDCT</td><td>-</td><td>CORDIC_ARCTANH</td></tr><tr><td>0100b</td><td>PQ_MTX_INV</td><td>PQ_TRANS_RFFT</td><td>PQ_FIR_INCREMENTAL</td><td>-</td></tr><tr><td>0101b</td><td>PQ_MTX_PROD</td><td>-</td><td>-</td><td>-</td></tr><tr><td>0110b</td><td>-</td><td>PQ_TRANS_RDCT</td><td>-</td><td>-</td></tr><tr><td>0111b</td><td>PQ_MTX_SUB</td><td>-</td><td>-</td><td>-</td></tr><tr><td>1000b</td><td>-</td><td>-</td><td>-</td><td>-</td></tr><tr><td>1001b</td><td>PQ_VEC_DOTP</td><td>-</td><td>-</td><td>-</td></tr><tr><td>1010b</td><td>PQ_MTX_TRAN</td><td>-</td><td>-</td><td>-</td></tr><tr><td>1011b - 1111b</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr></table>		CP_MTX	CP_FFT	CP_FIR	CP_CORDIC	0000b	-	PQ_TRANS_CFFT	PQ_FIR_FIR	-	0001b	PQ_MTX_SCALE	PQ_TRANS_IFFT	PQ_FIR_CONVOLUTION	CORDIC_ARCTAN	0010b	PQ_MTX_MULT	PQ_TRANS_CDCT	PQ_FIR_CORRELATION	-	0011b	PQ_MTX_ADD	PQ_TRANS_IDCT	-	CORDIC_ARCTANH	0100b	PQ_MTX_INV	PQ_TRANS_RFFT	PQ_FIR_INCREMENTAL	-	0101b	PQ_MTX_PROD	-	-	-	0110b	-	PQ_TRANS_RDCT	-	-	0111b	PQ_MTX_SUB	-	-	-	1000b	-	-	-	-	1001b	PQ_VEC_DOTP	-	-	-	1010b	PQ_MTX_TRAN	-	-	-	1011b - 1111b	Reserved	Reserved	Reserved	Reserved
	CP_MTX	CP_FFT	CP_FIR	CP_CORDIC																																																														
0000b	-	PQ_TRANS_CFFT	PQ_FIR_FIR	-																																																														
0001b	PQ_MTX_SCALE	PQ_TRANS_IFFT	PQ_FIR_CONVOLUTION	CORDIC_ARCTAN																																																														
0010b	PQ_MTX_MULT	PQ_TRANS_CDCT	PQ_FIR_CORRELATION	-																																																														
0011b	PQ_MTX_ADD	PQ_TRANS_IDCT	-	CORDIC_ARCTANH																																																														
0100b	PQ_MTX_INV	PQ_TRANS_RFFT	PQ_FIR_INCREMENTAL	-																																																														
0101b	PQ_MTX_PROD	-	-	-																																																														
0110b	-	PQ_TRANS_RDCT	-	-																																																														
0111b	PQ_MTX_SUB	-	-	-																																																														
1000b	-	-	-	-																																																														
1001b	PQ_VEC_DOTP	-	-	-																																																														
1010b	PQ_MTX_TRAN	-	-	-																																																														
1011b - 1111b	Reserved	Reserved	Reserved	Reserved																																																														

23.7.1.11 Length (LENGTH)

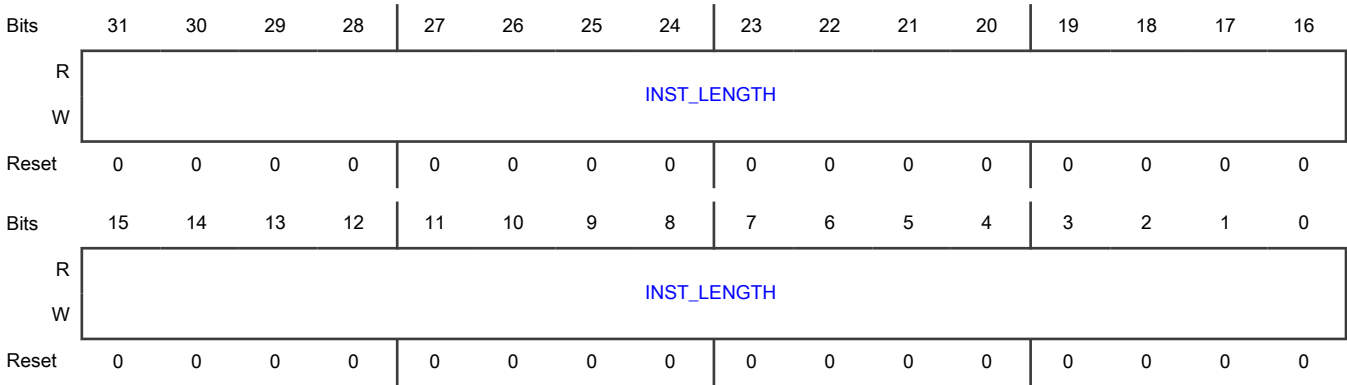
Offset

Register	Offset
LENGTH	104h

Function

Length register

Diagram



Fields

Field	Function
31-0 INST_LENGTH	Instruction length The INST_LENGTH field is remapped according to the operating engine in use. The value of this field provides the dimensions of the operands. For Matrix Operations: LENGTH[31:0] = Cols_B[20:16] Cols_A[12:8] Rows_A[4:0] For FIR Operations: LENGTH[31:0] = numTaps[31:16] numSamples[15:0] For other operations: LENGTH[31:0] is the value of size of the input operand. NOTE Maximum row/column is 16 by 16, regardless of 5-bit allowance. For vector operations, the maximum size is 256 (vector operations have only one dimension for length).

23.7.1.12 Coprocessor Prescale (CPPRE)

Offset

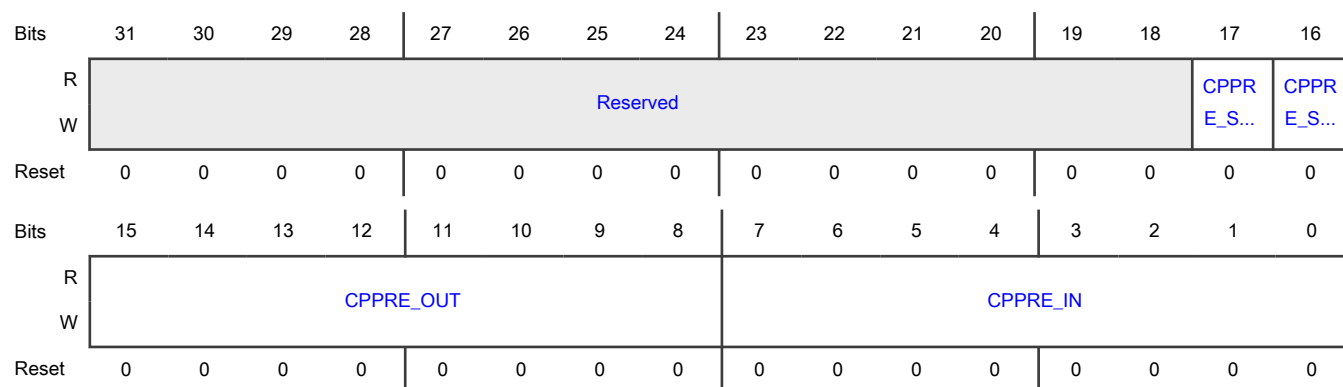
Register	Offset
CPPRE	108h

Function

Used to define the format of the input and output data used via the coprocessor bus interface. In addition, you can set the saturation for the output.

NOTE

CPPRE does not affect data used via the AHB bus interface.

Diagram**Fields**

Field	Function
31-18 —	Reserved
17 CPPRE_SAT8	Saturation 8 Output value saturation is applied at 'n' bits, when 'n' is either 8 bits or 16 bits. 0b - 8 bits 1b - 16 bits
16 CPPRE_SAT	Saturation Sub-32 bit saturation. Enforces saturation for the output data. If enabled, CPPRE_SAT8 defines saturation. 0b - No saturation 1b - Forces sub-32 bit saturation
15-8 CPPRE_OUT	Postscaling Output Contains the coprocessor postscaling value for fixed-point output data with fractional precision.
7-0 CPPRE_IN	Prescaling Input Contains the coprocessor scaling value for fixed-point input data with fractional precision.

23.7.1.13 Miscellaneous (MISC)

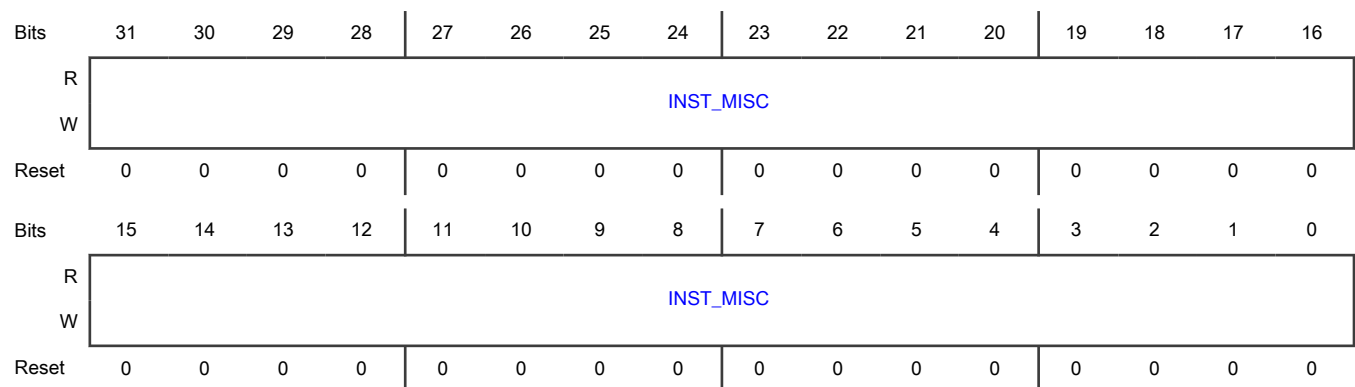
Offset

Register	Offset
MISC	10Ch

Function

Used to store the scaling value used for matrix scale operations. Each element of the matrix scales by the value of this register.

Diagram



Fields

Field	Function
31-0	Scaling Factor
INST_MISC	32-bit value that scales the elements of a matrix.

23.7.1.14 Cursory (CURSORY)

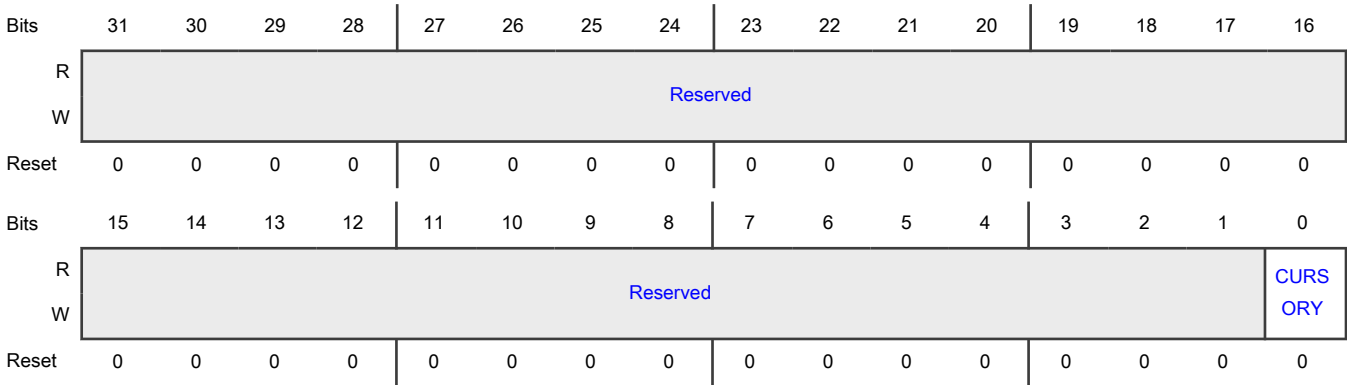
Offset

Register	Offset
CURSORY	110h

Function

Enables PowerQuad to reduce the resolution of results by zeroing out the LSB bits of the input and output of the floating-point MACs. Reducing the resolution reduces power consumption. Use when PowerQuad does not require greater accuracy.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CURSORY	Cursory Mode Cursory mode allows control of the input and output resolution to the MACs used by PowerQuad engines. 0b - Disable cursory mode, full floating-point accuracy (24-bit mantissa + 2 bits before rounding). 1b - Enable cursory Mode, 16-bit mantissa (bottom bits are zeroed for inputs and outputs of MACs).

23.7.1.15 CORDIC Input X (CORDIC_X)

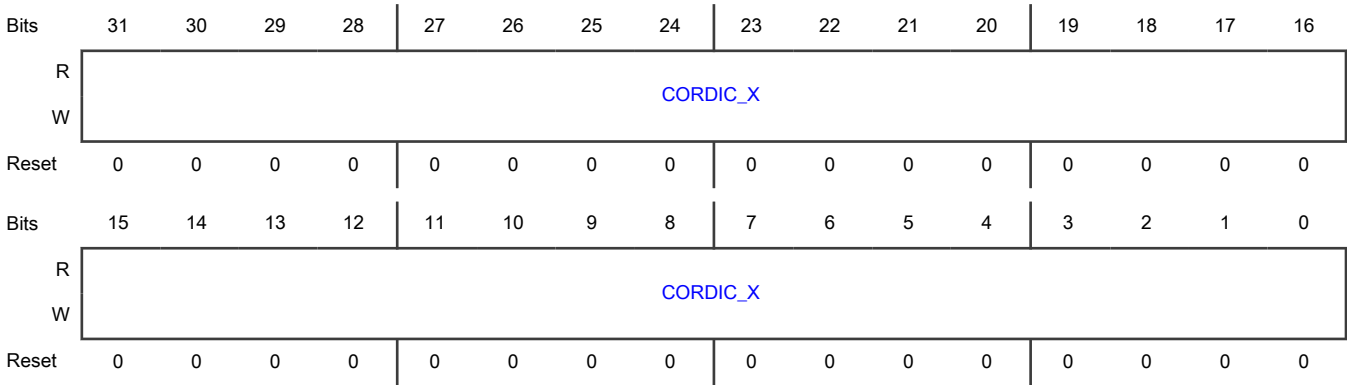
Offset

Register	Offset
CORDIC_X	180h

Function

Used for CORDIC-based algorithms. Contains the value of the first operand in the operation.

Diagram



Fields

Field	Function
31-0	CORDIC Input X
CORDIC_X	Fixed-point operand, X, for CORDIC-based operations.

23.7.1.16 CORDIC Input Y (CORDIC_Y)

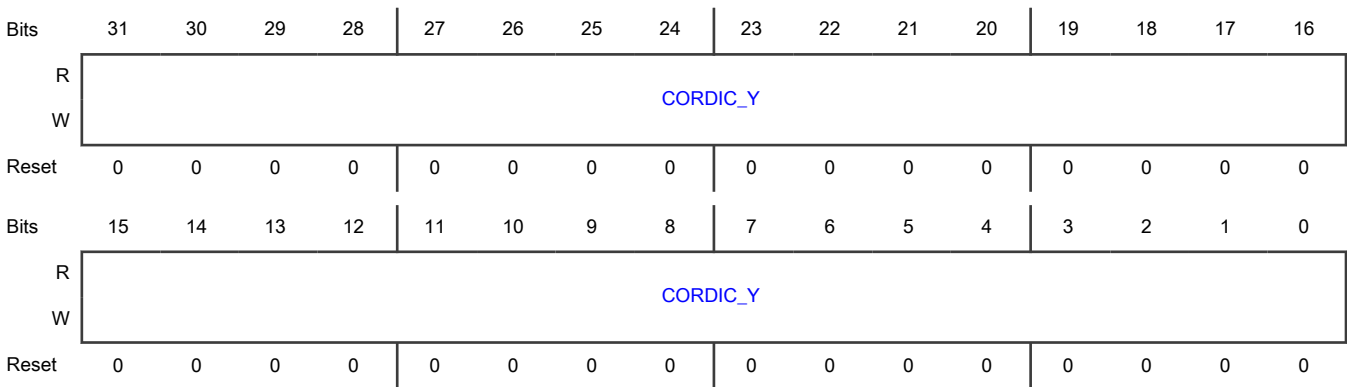
Offset

Register	Offset
CORDIC_Y	184h

Function

Used for CORDIC-based algorithms. Contains the value of the second operand in the operation.

Diagram



Fields

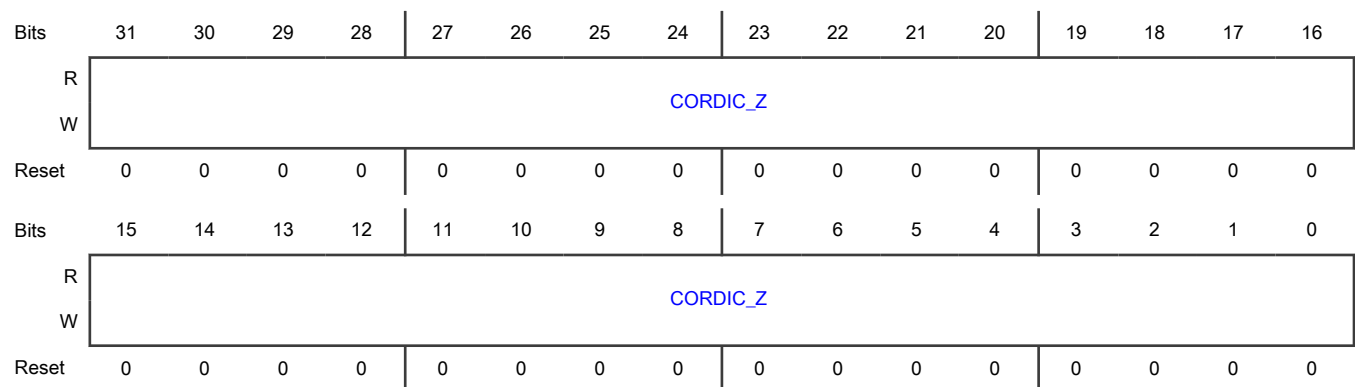
Field	Function
31-0 CORDIC_Y	CORDIC Input Y Fixed-point operand, Y, for CORDIC-based operations.

23.7.1.17 CORDIC Input Z (CORDIC_Z)**Offset**

Register	Offset
CORDIC_Z	188h

Function

Used for CORDIC-based algorithms. Contains the value of the result in the operation.

Diagram**Fields**

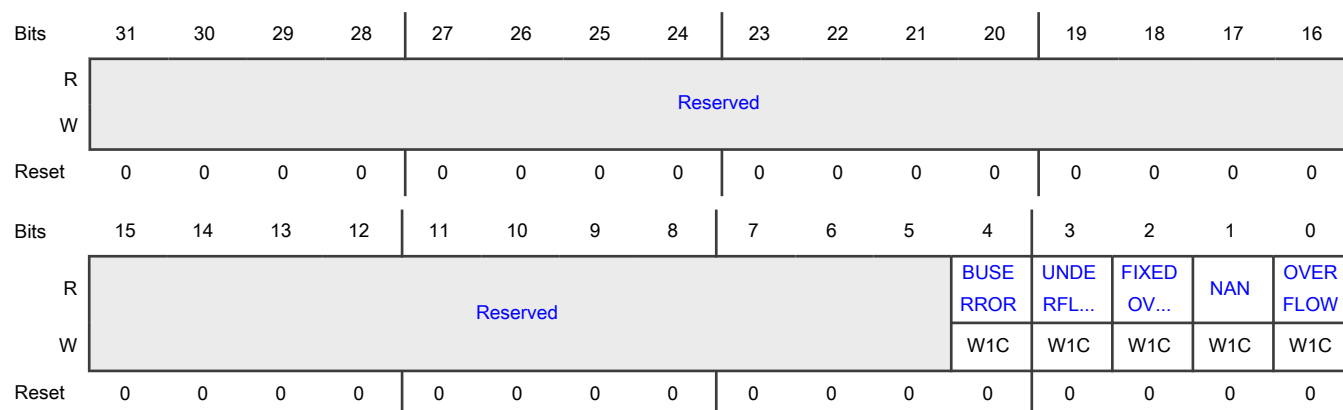
Field	Function
31-0 CORDIC_Z	CORDIC Input Z Fixed-point result data, Z, for CORDIC-based operations.

23.7.1.18 Error Status (ERRSTAT)**Offset**

Register	Offset
ERRSTAT	18Ch

Function

Provides 5 sticky status bits that capture an error during a PowerQuad operation. All bits are 'Write 1 to Clear'(W1C).

Diagram**Fields**

Field	Function
31-5 —	Reserved
4 BUSERROR	Bus Error An AHB master interface bus error has occurred. 0b - No error 1b - Error on bus
3 UNDERFLOW	Underflow Error occurs when the input is so small that it has been truncated to 0. 0b - No error 1b - Error on underflow
2 FIXEDOVERFLOW	Fixed-point Overflow Error occurs when there is more data than can be processed. This bit refers to fixed-point values. 0b - No error 1b - Error on fixed-point overflow
1 NAN	Floating-Point Not-a-Number (NaN) Error occurs when the data has been corrupted or is illegal. 0b - No error 1b - Error on floating-point NaN
0	Floating-point Overflow

Table continues on the next page...

Table continued from the previous page...

Field	Function
OVERFLOW	Error occurs when there is more data than can be processed. This bit refers to floating-point values. 0b - No error 1b - Error on floating-point overflow

23.7.1.19 Interrupt Enable (INTREN)

Offset

Register	Offset
INTREN	190h

Function

Determines which conditions trigger an interrupt, based on the set status of the ERRSTAT register and on successful completion of operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								INTR_	Reserved		INTR_	INTR_	INTR_	INTR_	INTR_
W									CO...			BE...	UF...	FL...	NAN	OF...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved
7 INTR_COMP	Interrupt on Instruction Completion 0b - Disable interrupt 1b - Enable interrupt
6-5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

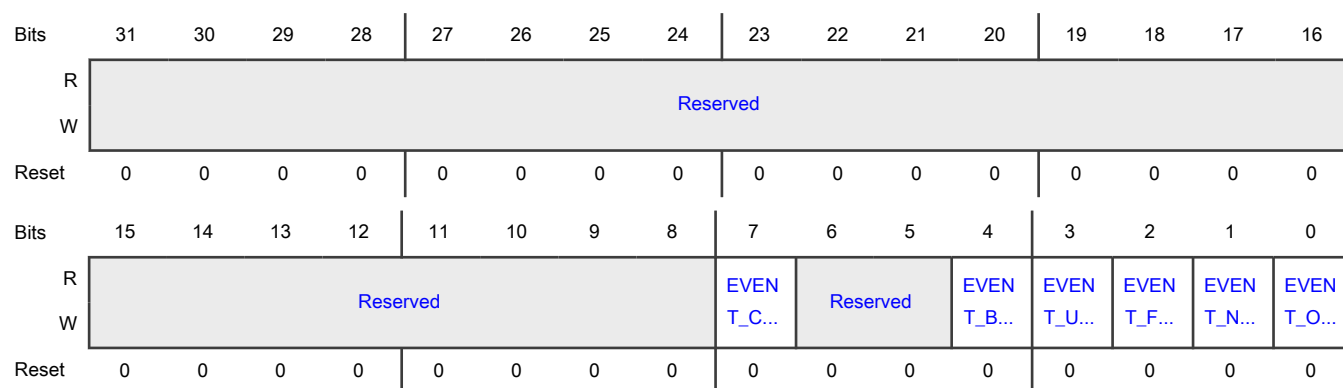
Field	Function
4 INTR_BERR	Interrupt on AHBM Bus Error 0b - Disable interrupt 1b - Enable interrupt
3 INTR_UFLOW	Interrupt on Underflow 0b - Disable interrupt 1b - Enable interrupt
2 INTR_FIXED	Interrupt on Fixed-point Overflow 0b - Disable interrupt 1b - Enable interrupt
1 INTR_NAN	Interrupt Floating-point NaN 0b - Disable interrupt 1b - Enable interrupt
0 INTR_OFLOW	Interrupt Floating-point Overflow 0b - Disable interrupt 1b - Enable interrupt

23.7.1.20 Event Enable (EVENTEN)**Offset**

Register	Offset
EVENTEN	194h

Function

Determines which conditions trigger an event based on the set status of the ERRSTAT register and on successful completion of operation.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7 EVENT_COMP	Event Trigger on Instruction Completion 0b - Disable event trigger 1b - Enable event trigger
6-5 —	Reserved
4 EVENT_BERR	Event Trigger on AHBM Bus Error 0b - Disable event trigger 1b - Enable event trigger
3 EVENT_UFLOW	Event Trigger on Underflow 0b - Disable event trigger 1b - Enable event trigger
2 EVENT_FIXED	Event Trigger on Fixed-point Overflow 0b - Disable event trigger 1b - Enable event trigger
1 EVENT_NAN	Event Trigger on Floating-Point NaN 0b - Disable event trigger 1b - Enable event trigger
0 EVENT_OFLOW	Event Trigger on Floating-point Overflow 0b - Disable event trigger 1b - Enable event trigger

23.7.1.21 Interrupt Status (INTRSTAT)

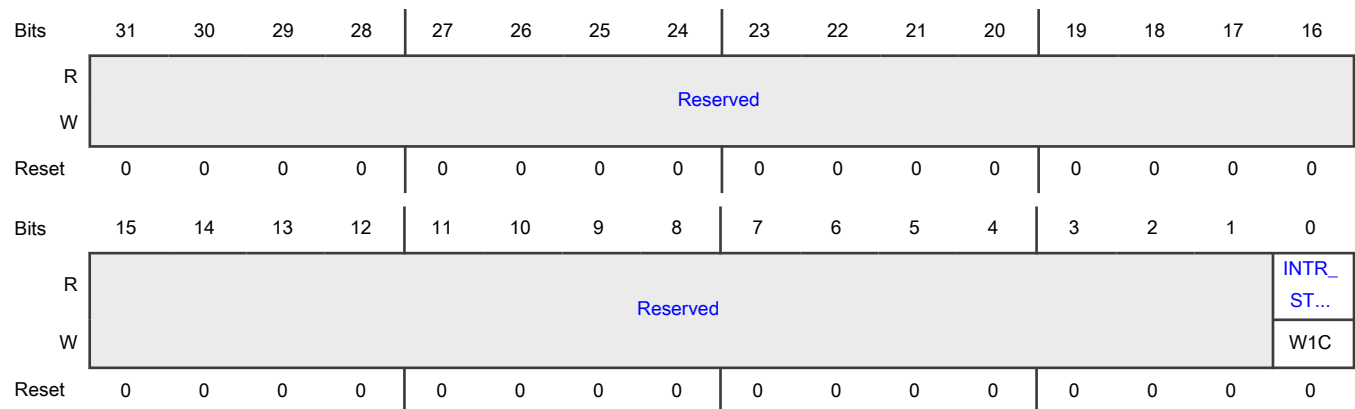
Offset

Register	Offset
INTRSTAT	198h

Function

Notifies on asserted trigger condition. A write of any value clears the Interrupt Status bit.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 INTR_STAT	Interrupt Status 0b - No new interrupt 1b - Interrupt captured

23.7.1.22 General Purpose Register Bank n (GPREG0 - GPREG15)

Offset

For n = 0 to 15:

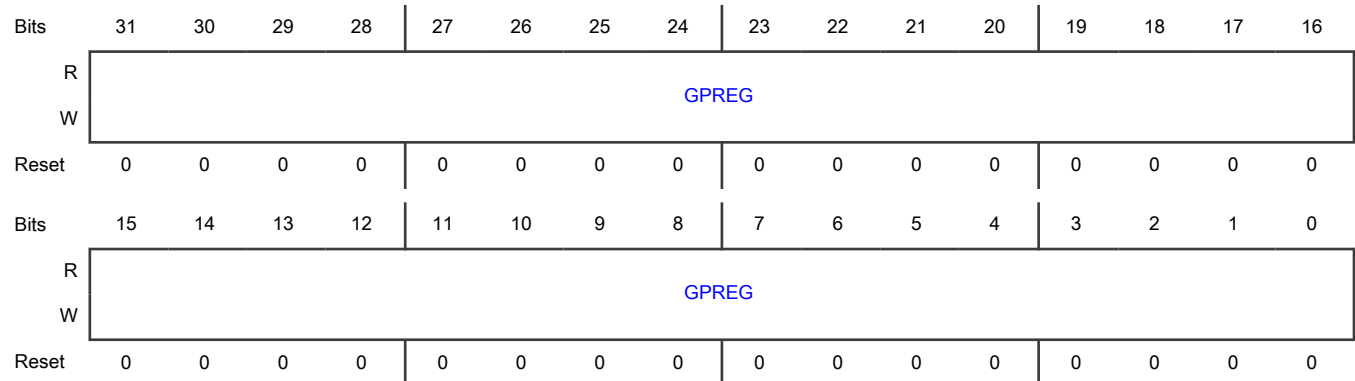
Register	Offset
GPREGn	200h + (n × 4h)

Function

The General Purpose Register Bank is $n \times 32$ bits. Used within the PowerQuad MAC engines. The values seen here are the staging and result registers of the adders and multipliers. These banks are made accessible so that you can back up the registers, when needed, to use PowerQuad for a different thread, then restore these saved values.

These registers are used by the Dual Biquad IIR engine and the Transform engine for intermediate data.

Diagram



Fields

Field	Function
31-0	General Purpose Bank
GPREG	Intermediate staging and result data can be pulled and restored as needed. However, it is important to consider when data is being manipulated, because it can corrupt the results of an operation.

23.7.1.23 Compute Register Bank n (COMPREG0 - COMPREG7)

Offset

For $n = 0$ to 7:

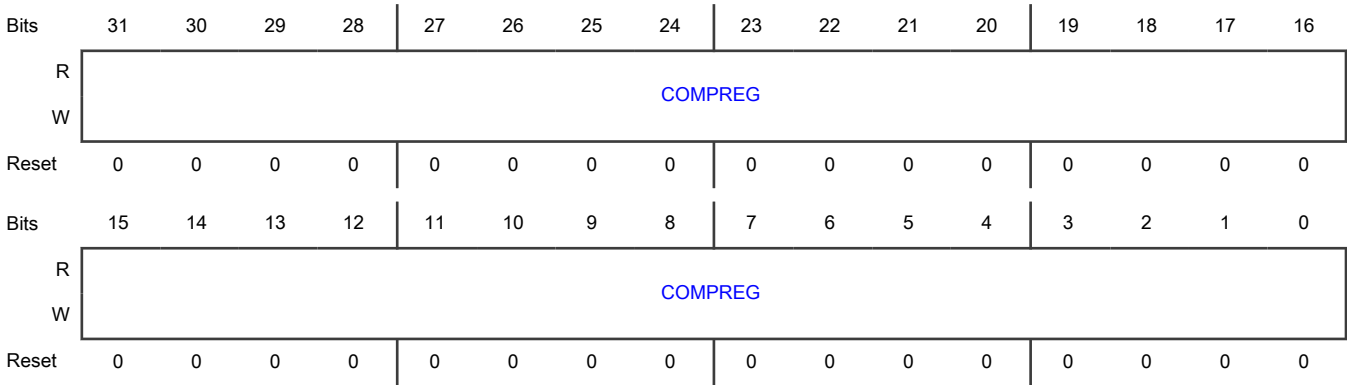
Register	Offset
COMPREGn	$240h + (n \times 4h)$

Function

The Compute Register Bank is $n \times 32$ bits. Used within the PowerQuad MAC engines. The values seen here are the staging or intermediate data used in FFT or Biquad operations. These banks are made accessible so that you can back up the registers, when needed, to use PowerQuad for a different thread. Then restore these saved values.

These registers are only used by the Dual Biquad IIR engine.

Diagram



Fields

Field	Function
31-0 COMPREG	Compute bank Intermediate staging and result data can be pulled and restored as needed. However it is important to consider when data is being manipulated, because it can corrupt the results of an operation.

Chapter 24

Casper

24.1 Chip-specific Casper information

Table 377. Reference links to related information

Topic	Related module	Reference
Full description	Casper	Casper
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

24.1.1 Module instances

This device has only one instance of the Casper module, CASPER0.

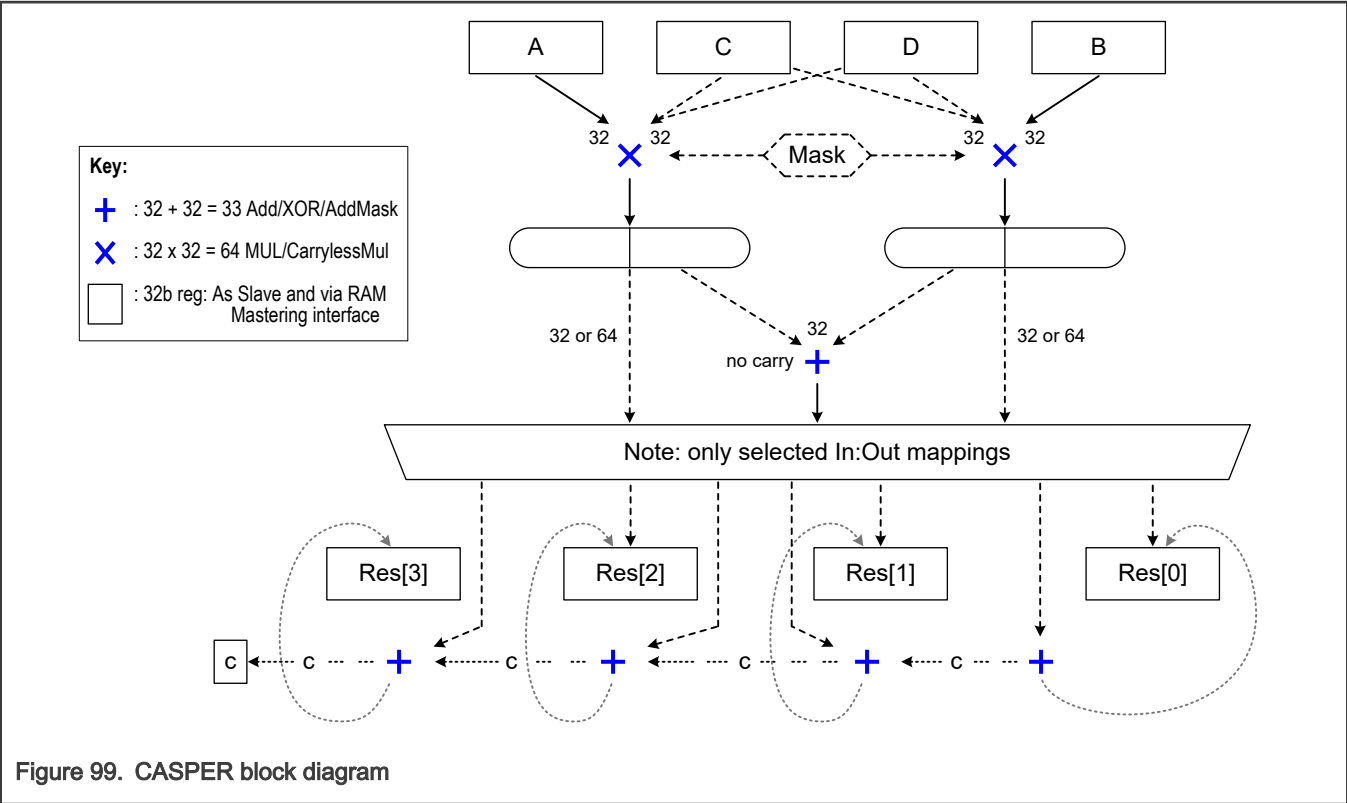
24.2 Overview

NOTE

API descriptions and other contents will be added to this chapter in a future release.

The Cryptographic Accelerator (CASPER) engine provides acceleration of asymmetric cryptographic algorithms. Using CASPER in conjunction with hardware blocks for hashing and symmetric cryptography leads to higher performance. Supported crypto functions are implemented in the SDK (Software Development Kit) and the Mbed™ TLS examples use the CASPER peripheral for computations.

24.2.1 Block diagram



24.2.2 Features

In a conventional MCU without CASPER, the core performs the computation work. Using CASPER frees up the core to do other tasks while CASPER does the computation.

CASPER helps with the acceleration of RSA, DH, and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:

- RSA modular exponentiation
- ECC scalar multiplication, point on curve check
- ECDSA signature generation and verification

Casper includes arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.

The CASPER engines serve to compute the assign functions faster than the core. This table shows a performance comparison for the various functions:

Table 378. CASPER performance improvements

Operation	Algorithm	Software-only version Arm® Cortex™-M33 @100 MHz	With CASPER acceleration Cortex-M33 @100Mhz	Performance improvement
Signing	ECDSA-secp256r1	258.5 ms	57.7 ms	4.48 times

Table continues on the next page...

Table 378. CASPER performance improvements (continued)

Operation	Algorithm	Software-only version Arm® Cortex™-M33 @100 MHz	With CASPER acceleration Cortex-M33 @100Mhz	Performance improvement
Verification	ECDSA-secp256r1	500 ms	58.59 ms	8.53 times
Key exchange	ECDHE-secp256r1	469 ms	92.59 ms	5.07 times
Key exchange	ECDH-secp256r1	250 ms	46.88 ms	5.33 times

24.3 Functional description

24.3.1 CASPER operation

The chip has user-specified portion of RAM for scratch pad purposes.

The CASPER module consist of:

- 4 x 32-bits (ABCD) Data registers, feeding the two multipliers.
- A mask register used for creating an XOR mask for unmasking ABCD and masking output for side channel protection
- The Multiplier has a special 1st *sum* mechanism to add the inner products back to complete a 64 x 64 multiply, which is much faster than a separated adder
- 4 (RES[0-3]) Results registers, which can be used with four adders, and do Add-Mask and XOR operations
- Special access to 2 RAMs (up to 8 kB) in parallel
 - The block can access these 2 RAM banks simultaneously, allowing for 2 (64 bits at a time) operations to work in parallel
- Interleaving RAMs (i.e., one is for the even words and one is for the odd words) allowing for 64 bit word pairs (128 bit) wide to be accessed every time
- A clock enable model so that when the speed of the clock is faster than the pipe, the clock enables support for an MCP (multi-cyclic path) approach to completion of the operations

24.3.2 CASPER co-processor operation

Arm v8-M architecture (Cortex-M33 core) introduces a co-processor interface allowing CASPER access via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Using these opcodes, up to two registers can be transferred between the core and CASPER.

When submitting the data and/or opcodes to a co-processor, the core can continue executing other tasks while the co-processor computes in parallel.

The CRm register index is used as the word address, with CRn as the *bank*.

For example, RES2 is offset 0x38 in memory, and CRm = (0x38 >> 2) = 0xE. It allows access to registers between offset 0x00 and 0x3C with CRn = 0x0. To access the higher registers, the CRn register can be set to 1, 2, or 3. For example, MASK can be accessed using CRn = 1, and CRm = 8. It can be computed as 0x60 >> 2 = 0x18, where the lower nibble goes into CRm, and the upper nibble into CRn.

The MCRR instruction allows access to pairs, including CTRL0/CTRL1, A/B, C/D, RES0/RES1, RES2/RES3 only. The lower index of the pair is used in CRm, with MCR.

24.3.3 CASPER AHB operation

A bus slave (AHB) and Cortex-M33 coprocessor interface so the application can access the control and data/results registers as needed.

The application may setup the operation, the iteration count, and the offset registers (offsets into the RAMs) optimized to allow two words (for example STRD or STM with memory, MCRR with CP) to configure and start in one write operation.

24.3.4 CASPER modes

The CASPER modes table lists descriptions of modes.

- Coarsely integrated operand scanning = CIOS
- Most Significant Word = MSW
- Least Significant Word = LSW

Table 379. CASPER modes

Mode	Name	Description	Comments
0x01	MUL6464_NOSUM	Walking 1 or more of J loop, doing $w[j]=ab*cd[j]$ base on $64 \times 64=128$	Writes out results, but does not read in to add
0x02	MUL6464_SUM	Walking 1 or more of J loop, doing $c,w[j]=w[j]+ab*cd[j]$ base on $64 \times 64=128$	Sums by reading result word ($w[j]$) and adding before writing back. This does not read the final 2 words before writing, since it is assumed they are farthest reached ($w[i+j]$)
0x03	MUL6464_FULLSUM	Walking 1 or more of J loop, doing $c,w[j]=w[j]+ab*cd[j]$ base on $64 \times 64=128$, sum all of w	Reads all of w, including the MSWs, it can have a carry in the carry bit Includes 1st loop half of CIOS multiply use (before reduce)
0x04	MUL6464_REDUCE	Walking 1 or more of J loop, doing $c,w[j-1]=w[j]+m*cd[j]$ base on $64 \times 64=128$, but skip 1st write. Note that m is in ab and is $w[0]*Np$	• Does not compute m into ab, need to this first. • The reduction pass of a CIOS double J loop. So, it writes to the preceding result double-word, except the 1st time, where it throws away the low-order 64 bits. • The full Montgomery use is FULLSUM 1st (1st J loop), then processor computes 1st product result of $w[0]*Np64$ (modulus $N \backslash Q$ as a 64 bit value) • This operation follows same as FULLSUM except it skips 1st write so it can write to previous
0x08	ADD64	ADD with ABOFF, and in/out RESOFF base on $c,r=r+a+c$	Uses 64 bits at a time, producing 65 bit output. Final carry in the carry bit.
0x09	SUB64	SUBTRACT with ABOFF, and in/out RESOFF base on $r=r-a$	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using $r=r+(\sim a+1)$

Table continues on the next page...

Table 379. CASPER modes (continued)

Mode	Name	Description	Comments
0x0C	RSUB64	SUBTRACT with ABOFF, and in/out RESOFF base on $r=r-a$	Uses 64 bits at a time, and if a is larger, final borrow is implicit, but carry bit set if borrowed. Solved using $r=r+(\sim a+1)$
0x0A	DOUBLE64	ADD to self with RESOFF base on $c, r=r+r+c$	Doubles a value, same as $*2$ or $<<1$ Uses 64 bits at a time, producing a 65 bit output, with final carry in the carry bit.
0x0B	XOR64	XOR with ABOFF, and in/out RESOFF base on $r=r^{\wedge}a$	Uses 64 bits at a time, producing 64 bit output. No carry bit.
0x14	COPY	Copy from ABOFF to RESOFF using 64 bits at a time	
0x16	FILL	Fill RESOFF with value in A and B, 64 bits at a time	
0x17	ZERO	Fill RESOFF WITH 0s, 64 bits at a time	

24.4 Memory map and register definition

This section includes the CASPER module memory map and detailed descriptions of all registers.

NOTE

This module does not generate transfer errors when accessing unimplemented registers in its register space.

24.4.1 CASPER register descriptions

24.4.1.1 CASPER memory map

CASPER0 base address: 4020_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control 0 (CTRL0)	32	RW	0000_0000h
4h	Control 1 (CTRL1)	32	RW	0000_0000h
Ch	Status (STATUS)	32	RW	0000_0000h
10h	Interrupt Enable Set (INTENSET)	32	RW	0000_0000h
14h	Interrupt Enable Clear (INTENCLR)	32	RW	0000_0000h
18h	Interrupt status (INTSTAT)	32	R	0000_0000h
20h	A Register (AREG)	32	RW	0000_0000h
24h	B Register (BREG)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
28h	C Register (CREG)	32	RW	0000_0000h
2Ch	D Register (DREG)	32	RW	0000_0000h
30h	Result Register 0 (RES0)	32	RW	0000_0000h
34h	Result Register 1 (RES1)	32	RW	0000_0000h
38h	Result Register 2 (RES2)	32	RW	0000_0000h
3Ch	Result Register 3 (RES3)	32	RW	0000_0000h
60h	Mask (MASK)	32	RW	0000_0000h
64h	Remask (REMASK)	32	RW	0000_0000h
80h	Lock (LOCK)	32	RW	0000_0000h

24.4.1.2 Control 0 (CTRL0)

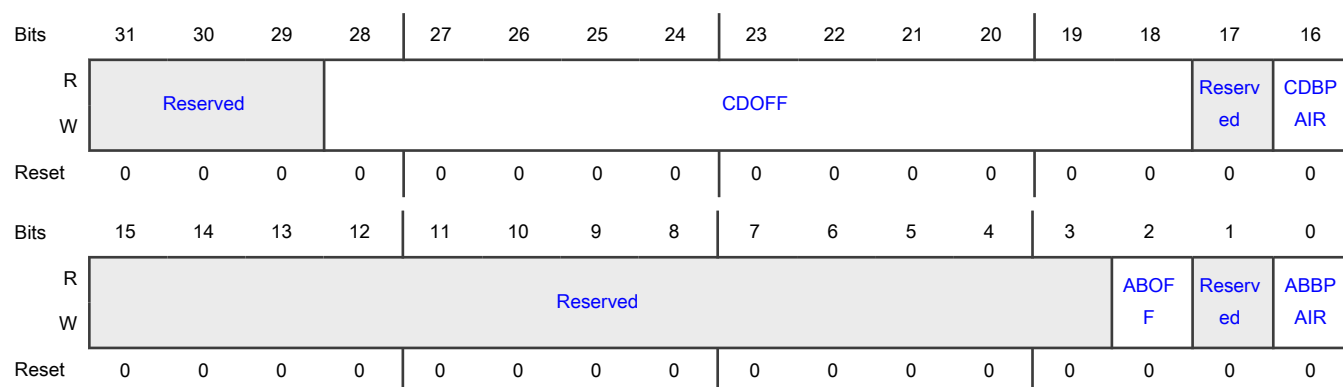
Offset

Register	Offset
CTRL0	0h

Function

Contains the offsets of AB and CD in the RAM.

Diagram



Fields

Field	Function
31-29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
28-18 CDOFF	CD Offset Word or DWord Offset of CD, with D at [2]=0 and C at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed with 32 bit operation. Ideally not in the same RAM as the AB values.
17 —	Reserved
16 CDBPAIR	CDOFF Bank Pair Which bank-pair the offset CDOFF is within. This must be 0 if only 2-up 0b - Bank-pair 0 (1st) 1b - Bank-pair 1 (2nd)
15-3 —	Reserved
2 ABOFF	AB Offset Word or DWord Offset of AB values, with B at [2]=0 and A at [2]=1 as far as the code sees (normally will be an interleaved bank so only sequential to AHB). Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the CD values if 4-up.
1 —	Reserved
0 ABBPAAIR	ABOFF Bank Pair Which bank-pair the offset ABOFF is within. This must be 0 if only 2-up. 0b - Bank-pair 0 (1st) 1b - Bank-pair 1 (2nd)

24.4.1.3 Control 1 (CTRL1)

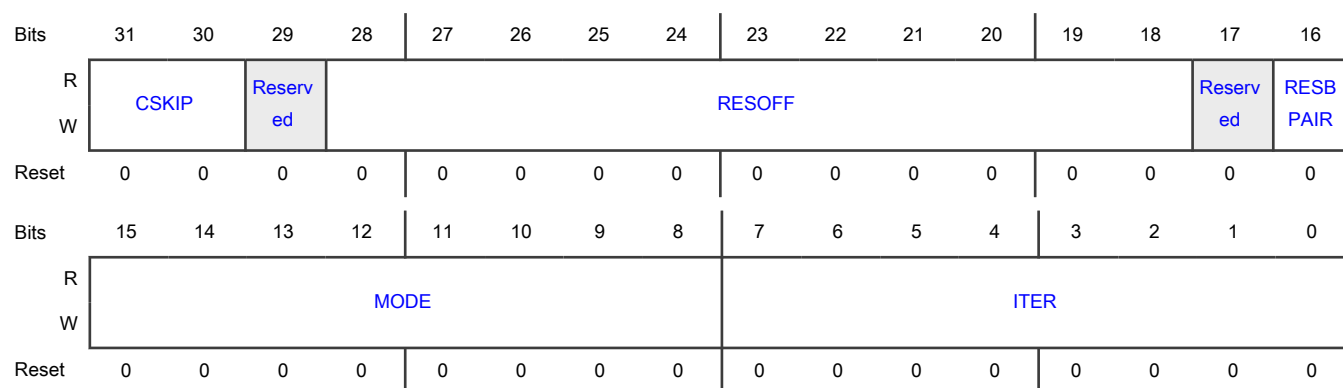
Offset

Register	Offset
CTRL1	4h

Function

Contains the opcode mode, iteration count, and result offset (in RAM) and also launches the accelerator. Note: with CP version: CTRL0 and CTRL1 can be written in one go with MCRR.

Diagram



Fields

Field	Function
31-30 CSKIP	Skip Rules on Carry Skip rules on Carry if needed. This operation will be skipped based on Carry value (from previous operation) if not 0: 00b - No Skip 01b - Skip if Carry is 1 10b - Skip if Carry is 0 11b - Set CTRL0FF to CDOFF and Skip
29 —	Reserved
28-18 RESOFF	Result Offset Word or DWord Offset of result. Word offset only allowed if 32 bit operation. Ideally not in the same RAM as the AB and CD values
17 —	Reserved
16 RESBPAIR	RESOFF Bank Pair Which bank-pair the offset RESOFF is within. This must be 0 if only 2-up. Ideally this is not the same bank as ABBPAIR (when 4-up supported). 0b - Bank-pair 0 (1st) 1b - Bank-pair 1 (2nd)
15-8 MODE	Mode Operation mode to perform. A write 0 means Accelerator is inactive. A write other than 0 means accelerator is active.
7-0 ITER	Iteration Counter The Iteration Counter is the number_cycles – 1. Write 0 means does one cycle – does not iterate.

24.4.1.4 Status (STATUS)

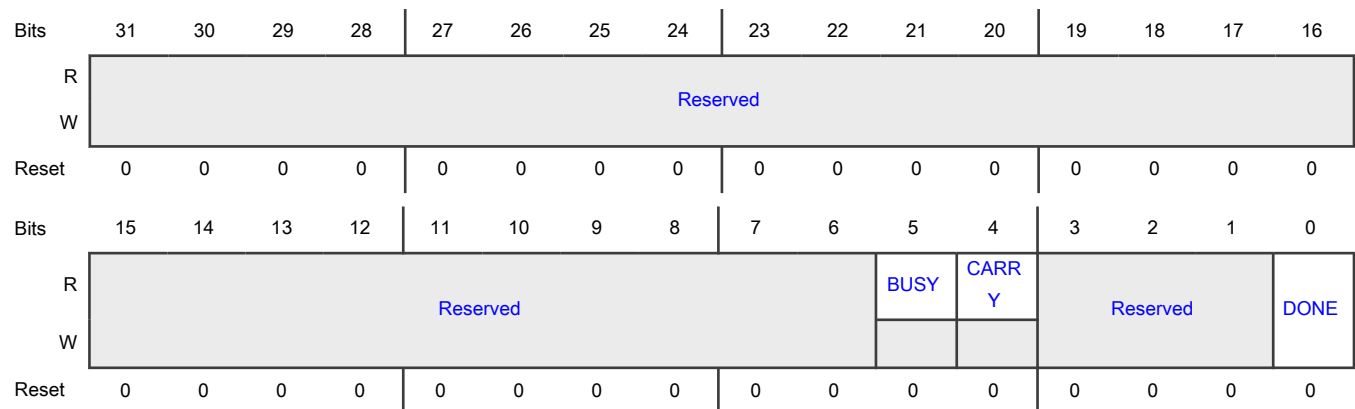
Offset

Register	Offset
STATUS	Ch

Function

Indicates operational status and would contain the carry bit if used.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 BUSY	Busy Indicates if the accelerator is busy performing an operation 0b - Not busy – is idle 1b - Is busy
4 CARRY	Carry Last carry value if operation produced a carry bit 0b - Carry was 0 or no carry 1b - Carry was 1
3-1 —	Reserved
0	Done

Table continues on the next page...

Table continued from the previous page...

Field	Function
DONE	Indicates if the accelerator has finished an operation. Write 1 to clear, or write CTRL1 to clear. 0b - Busy or just cleared 1b - Completed last operation

24.4.1.5 Interrupt Enable Set (INTENSET)

Offset

Register	Offset
INTENSET	10h

Function

Sets interrupts

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 DONE	Done Set if the accelerator should interrupt when done. 0b - Do not interrupt when done 1b - Interrupt when done

24.4.1.6 Interrupt Enable Clear (INTENCLR)

Offset

Register	Offset
INTENCLR	14h

Function

Clears interrupts

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DONE
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved
0 DONE	Done Written to clear an interrupt set with INTENSET. 0b - If written 0, ignored 1b - If written 1, do not interrupt when done.

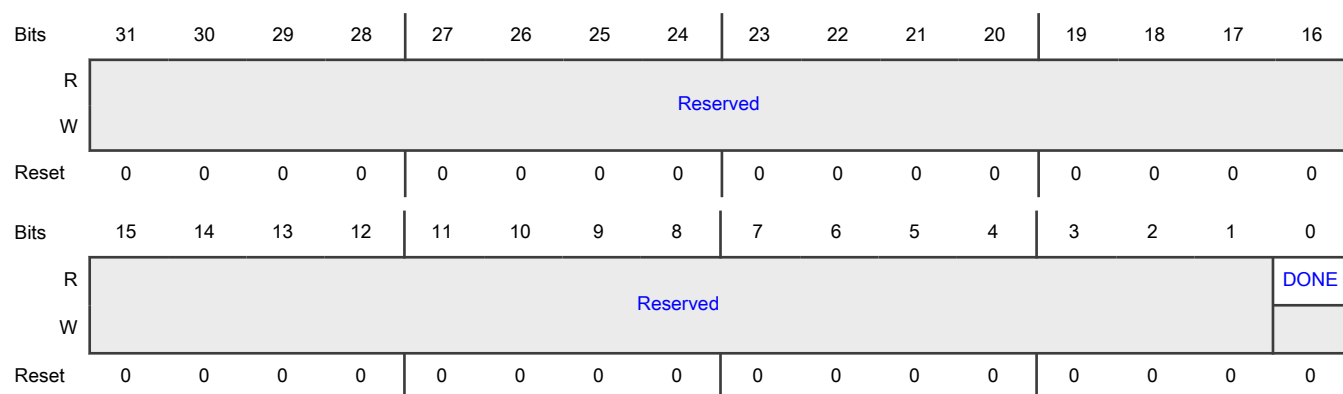
24.4.1.7 Interrupt status (INTSTAT)

Offset

Register	Offset
INTSTAT	18h

Function

Interrupt status bits (mask of INTENSET and STATUS)

Diagram**Fields**

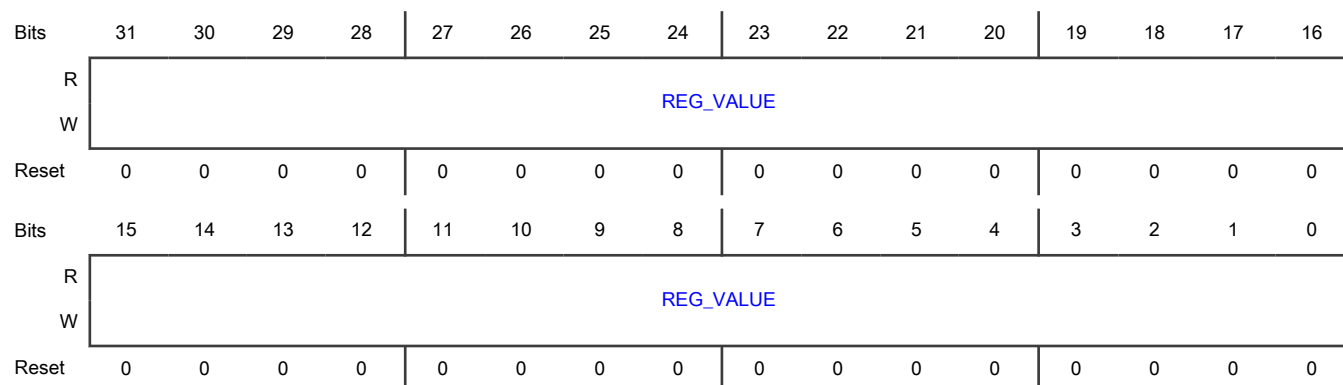
Field	Function
31-1 —	Reserved
0 DONE	If set, interrupt is caused by accelerator being done. 0b - Not caused by accelerator being done 1b - Caused by accelerator being done

24.4.1.8 A Register (AREG)**Offset**

Register	Offset
AREG	20h

Function

AREG is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram

Fields

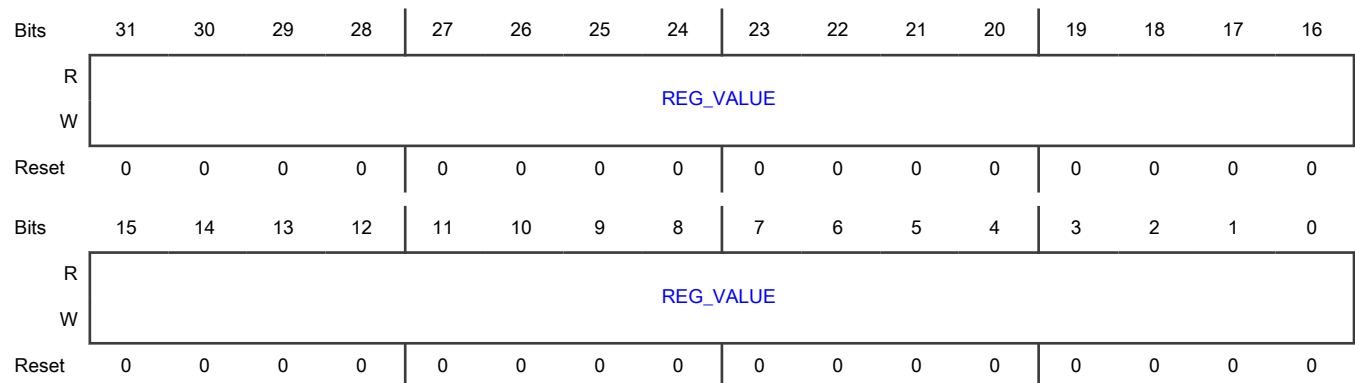
Field	Function
31-0	Register Value
REG_VALUE	Register to be fed into Multiplier.

24.4.1.9 B Register (BREG)**Offset**

Register	Offset
BREG	24h

Function

BREG is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram**Fields**

Field	Function
31-0	Register Value
REG_VALUE	Register to be fed into Multiplier.

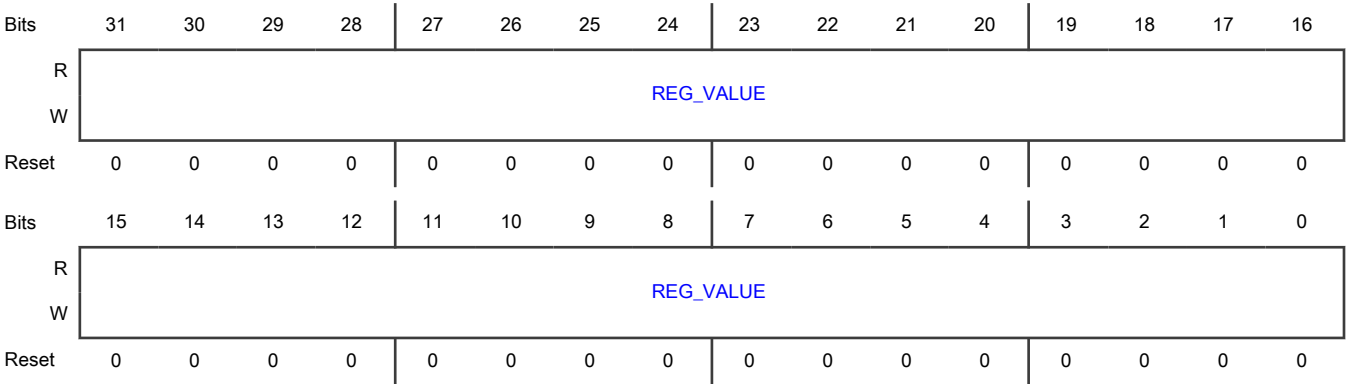
24.4.1.10 C Register (CREG)**Offset**

Register	Offset
CREG	28h

Function

CREG is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram



Fields

Field	Function
31-0	Register Value
REG_VALUE	Register to be fed into Multiplier.

24.4.1.11 D Register (DREG)

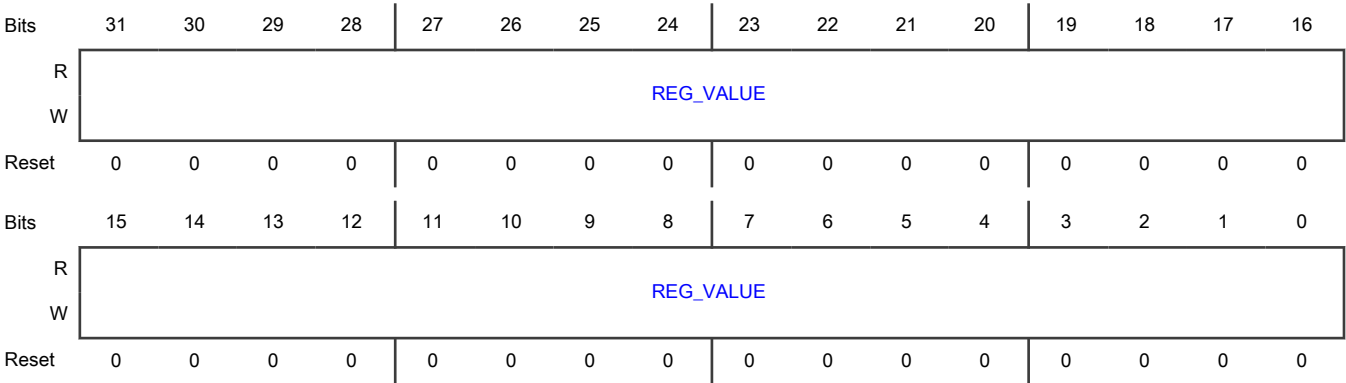
Offset

Register	Offset
DREG	2Ch

Function

DREG is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram



Fields

Field	Function
31-0	Register Value
REG_VALUE	Register to be fed into Multiplier.

24.4.1.12 Result Register 0 (RES0)

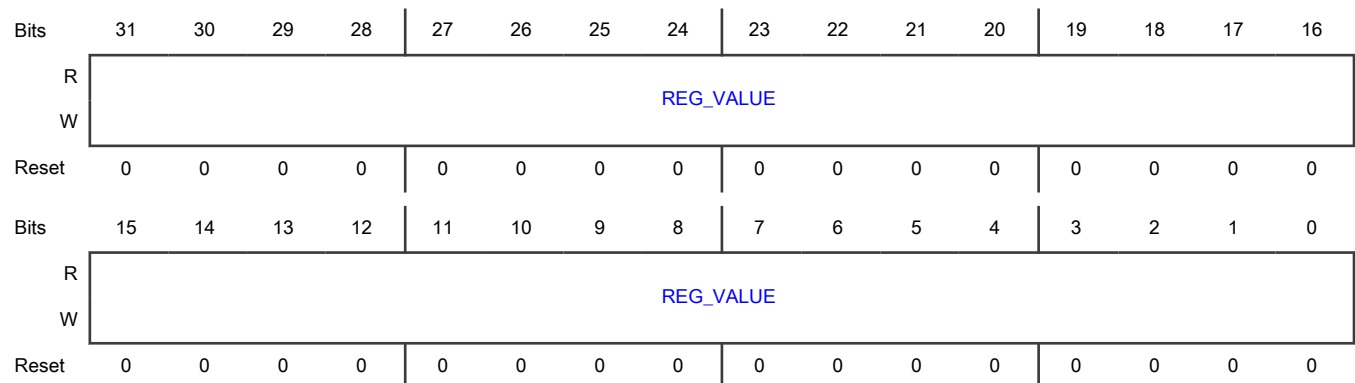
Offset

Register	Offset
RES0	30h

Function

RES0 is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram



Fields

Field	Function
31-0	Register Value
REG_VALUE	Register to hold working result (from multiplier, adder/xor, etc).

24.4.1.13 Result Register 1 (RES1)

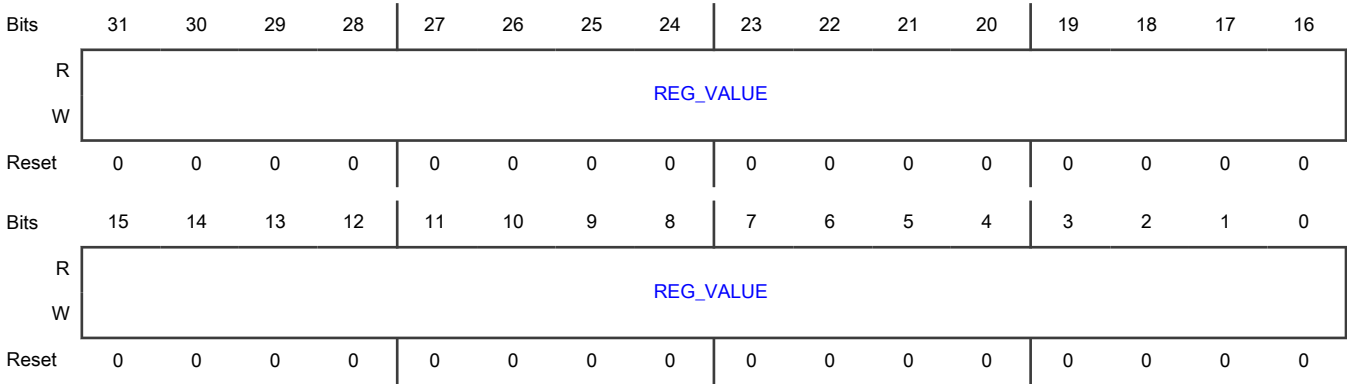
Offset

Register	Offset
RES1	34h

Function

RES1 is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram



Fields

Field	Function
31-0 REG_VALUE	Register to hold working result (from multiplier, adder/xor, etc).

24.4.1.14 Result Register 2 (RES2)

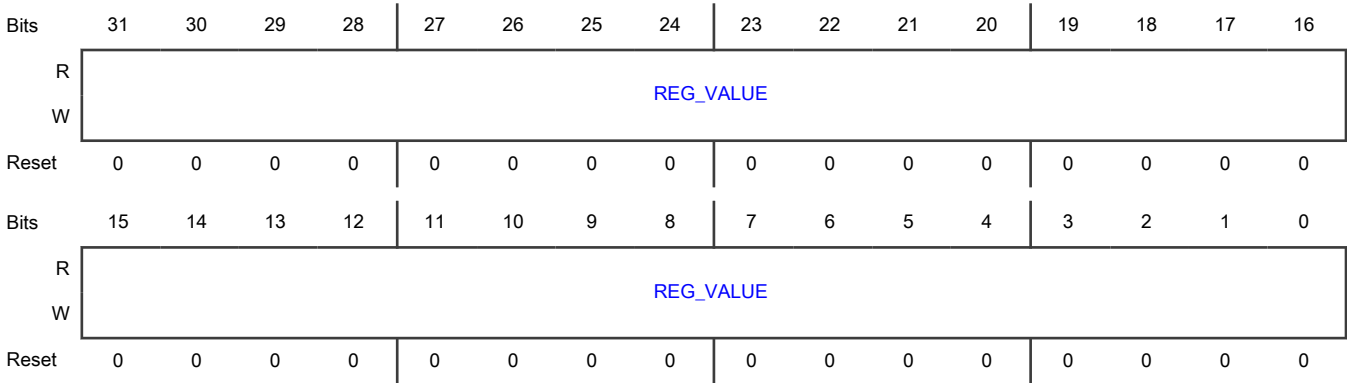
Offset

Register	Offset
RES2	38h

Function

RES2 is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram



Fields

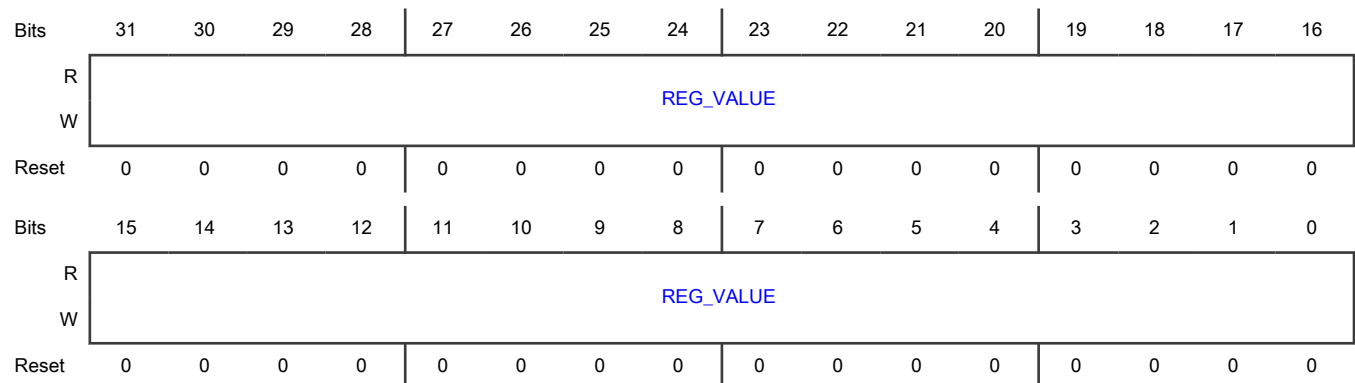
Field	Function
31-0 REG_VALUE	Register Value Register to hold working result (from multiplier, adder/xor, etc).

24.4.1.15 Result Register 3 (RES3)**Offset**

Register	Offset
RES3	3Ch

Function

RES3 is not normally written or read by the application, but is available when the accelerator is not busy.

Diagram**Fields**

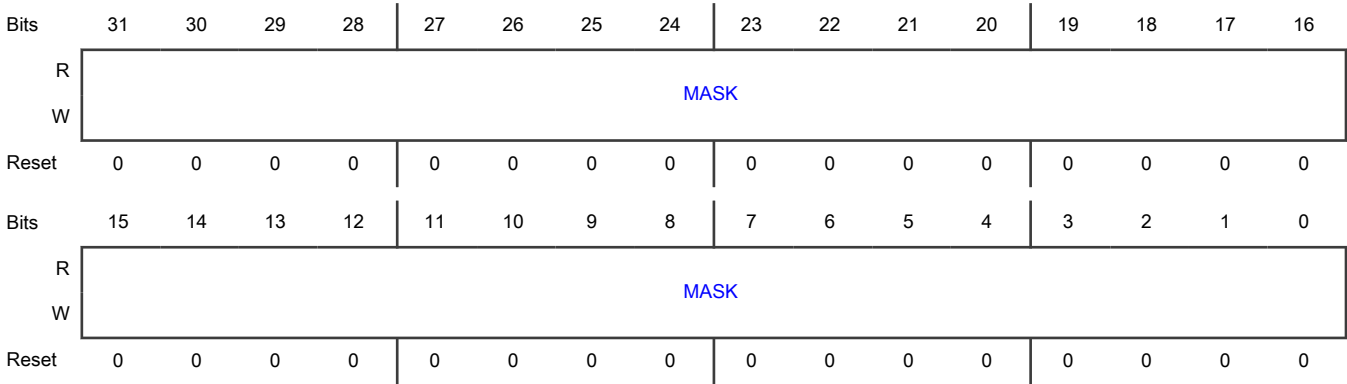
Field	Function
31-0 REG_VALUE	Register to hold working result (from multiplier, adder/xor, etc).

24.4.1.16 Mask (MASK)**Offset**

Register	Offset
MASK	60h

Function
Optional mask register

Diagram



Fields

Field	Function
31-0	Mask
MASK	Mask to apply as a side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values

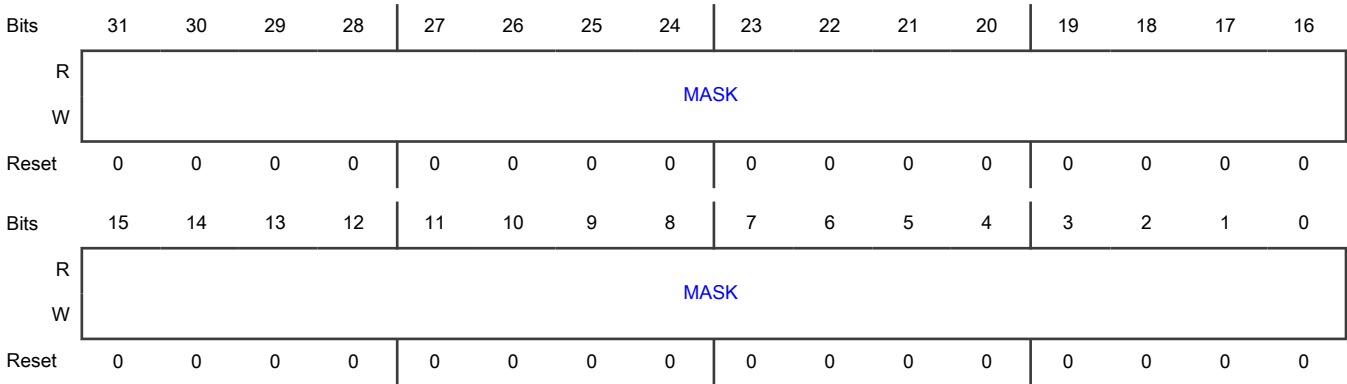
24.4.1.17 Remask (REMASK)

Offset

Register	Offset
REMASK	64h

Function
Optional re-mask register

Diagram



Fields

Field	Function
31-0 MASK	Mask Mask to apply as a side channel countermeasure. 0: No mask to be used. N: Mask to XOR onto values

24.4.1.18 Lock (LOCK)

Offset

Register	Offset
LOCK	80h

Function

Security lock register

Diagram

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

Fields

Field	Function
31-17 —	Reserved
16-4 KEY	Key Must be written as 0x73D to change the register. 0_0111_0011_1101b - Key Value. If set during write, will allow lock or unlock.
3-1	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
0 LOCK	Lock Reads back with security level locked to, or 0. Writes as 0 to unlock, 1 to lock. 0b - Unlock 1b - Lock to current security level

Chapter 25

Physically Unclonable Function (PUF)

25.1 Chip-specific PUF controller information

Table 380. Reference links to related information

Topic	Related module	Reference
Full description	PUF	PUF
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

25.1.1 Chip-Specific initialization

Initial configuration of the PUF can be accomplished as follows:

- Enable the clock to the PUF in the CLKCTL0_PSCCTL0 register. This enables the register interface and the peripheral function clock.
- Clear the PUF peripheral reset in the RSTCTL0_PRSTCTL0 register by writing to the RSTCTL0_PRSTCTL0_CLR register.
- The PUF provides an interrupt to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable this in the SYSCTL0_STARTEN1 register.

25.1.2 Module instances

This device has only one instance of the PUF module, PUF0.

25.2 Overview

The Physically Unclonable Function (PUF) controller provides secure key storage without injecting or provisioning a device unique PUF root key.

25.2.1 Features

The PUF controller has the following features:

- Key strength of 256-bits.
 - The PUF constructs 256-bit strength device unique PUF root key using the digital fingerprint of a device derived from SRAM and error correction data called Activation Code (AC). The AC is generated during the enrollment process and must be stored on an external non-volatile memory device in the system.
- Generation, storage, and reconstruction of keys.
- Key sizes from 64-bits to 4096-bits.
 - PUF controller allows storage of keys, generated externally or on chip, of sizes 64-bits to 4096-bits.
 - PUF controller combines keys with a digital fingerprint of the device to generate key codes. These key codes should be provided to the controller to reconstruct the original key. They can be stored on an external non-volatile memory device in the system.
- Key output via a dedicated hardware interface or through a register interface.

- PUF controller allows assigning a 4-bit index value for each key while generating key codes. Keys that are assigned index value zero are output through HW bus, accessible to AES and OTFAD block only. Keys with non-zero index are available through APB register interface.
- 32-bit APB interface.
- Programmable feature that can block an index from generating new key codes.

25.3 Functional description

25.3.1 PUF controller and key management

25.3.2 Key storage/management

The Key Management module supports storing an AES Key (KEY0) and three OTFAD Keys (KEY1, KEY2, KEY3). These keys are fed into their respective IPs via a dedicated hard-wired interface. Since these keys are from Index-0, they are inaccessible by the software interface. The OTFAD requires a 128-bit key. The device supports up to three regions, therefore, three separate 128-bit keys are made available through the Key Management block. The AES key can be 128 bits, 192 bits or 256 bits in length.

PUF keys for AES and OTFAD, if already loaded, are retained during Deep-sleep and Power-down modes but not retained during Deep-power down modes. The CTRL, CFG, KEYLOCK, KEYENABLE, KEYRESET, IDXBLK_L/_H, IDXBLK_L_DP/_H_DP registers are preserved and not reset during Power-down mode.

This module supports blocking of access to a set of indexes such that they cannot be used anymore for key generation or retrieval until next reset.

25.3.3 Key loading procedure

To load KEYn for use by the AES or OTFAD, use the following procedure:

1. Write the enable value, KEYRESET[KEYn] = 0x2, to clear the associated KEYn hold and KEYn_SHIFT_STATUS registers.
2. Write the enable value, KEYENABLE[KEYn] = 0x2. Ensure that only the intended KEYENABLE[KEYn] field is enabled; if multiple keys are enabled, then no key will be enabled.
3. For added security protection, write a random mask value to the KEYMASKn register.
4. Issue the Get Key command to the PUF, requesting the desired key with KEYINDEX = 0, so that the key is presented on the dedicated hardware interface to the Key Management module. See [Get key](#) and [Pseudocode get key function](#). It is assumed that PUF initialization and start have already been performed before issuing a Get Key command. The requested key will be loaded into the KEYn hold register, which is only visible to the AES or OTFAD.
5. If required, write the disable value, KEYLOCK[KEYn] = 0x1, to prevent any further changes to KEYn.

25.3.4 SRAM PUF power control

The SRAM used for the PUF is a separate block of memory that is only accessible by the PUF controller. All self-test is built into the PUF logic and is not accessible through any chip test modes. The PUF SRAM has a power switch to ensure there is clean start-up power and is controlled via the PUF PWRCTRL register. If a previously powered PUF SRAM is turned off by clearing the RAMON bit (PWRCTRL[RAMON] = 0), up to a 400 ms second delay is required before the PUF SRAM can be powered again by setting the RAMON bit (PWRCTRL[RAMON] = 1). The required delay depends on temperature conditions and the delay is highest only at the worst corners. ROM API implements a special scheme where the first attempt is made with a smaller delay and if that fails, a 400 ms delay is applied in the next iteration.

25.3.5 PUF controller operations

The PUF controller supports the following operations:

1. Enroll: The controller retrieves the Startup Data (SD) from the memory (SRAM), derives a digital fingerprint, generates the corresponding Activation Code (AC) and sends it to the Client Design (CD) to be stored externally. Perform this step only once for each device.

There is a configuration register CFG that can block further enrollment.

2. Start: The AC generated during the enroll operation and the SD are used to reconstruct the digital fingerprint. This is done after every power-up and reset.
3. Generate Key: The controller generates an unique key and combines it with the digital fingerprint to output a key code. Each time a Generate Key operation is executed a new unique key is generated.
4. Set Key: The digital fingerprint generated during the Enroll/Start operations and the key provided by the Client Design (CD) are used to generate a Key Code. This KC can be stored externally. Perform this operation only once for each key.
5. Get Key: The digital fingerprint generated during the Start operation and the KC generated during a Set Key operation are used to retrieve a stored key. Perform this operation every time a key is needed.

25.3.5.1 Order of operations

After power-up or reset the PUF controller starts in one of the three Init states, depending on its previous state. It first initializes itself (indicated by STAT[BUSY] = 1).

When initialization is finished, the PUF controller can be moved to one of the cold or warm states. After power-up an enroll or a start operation can be performed.

NOTE

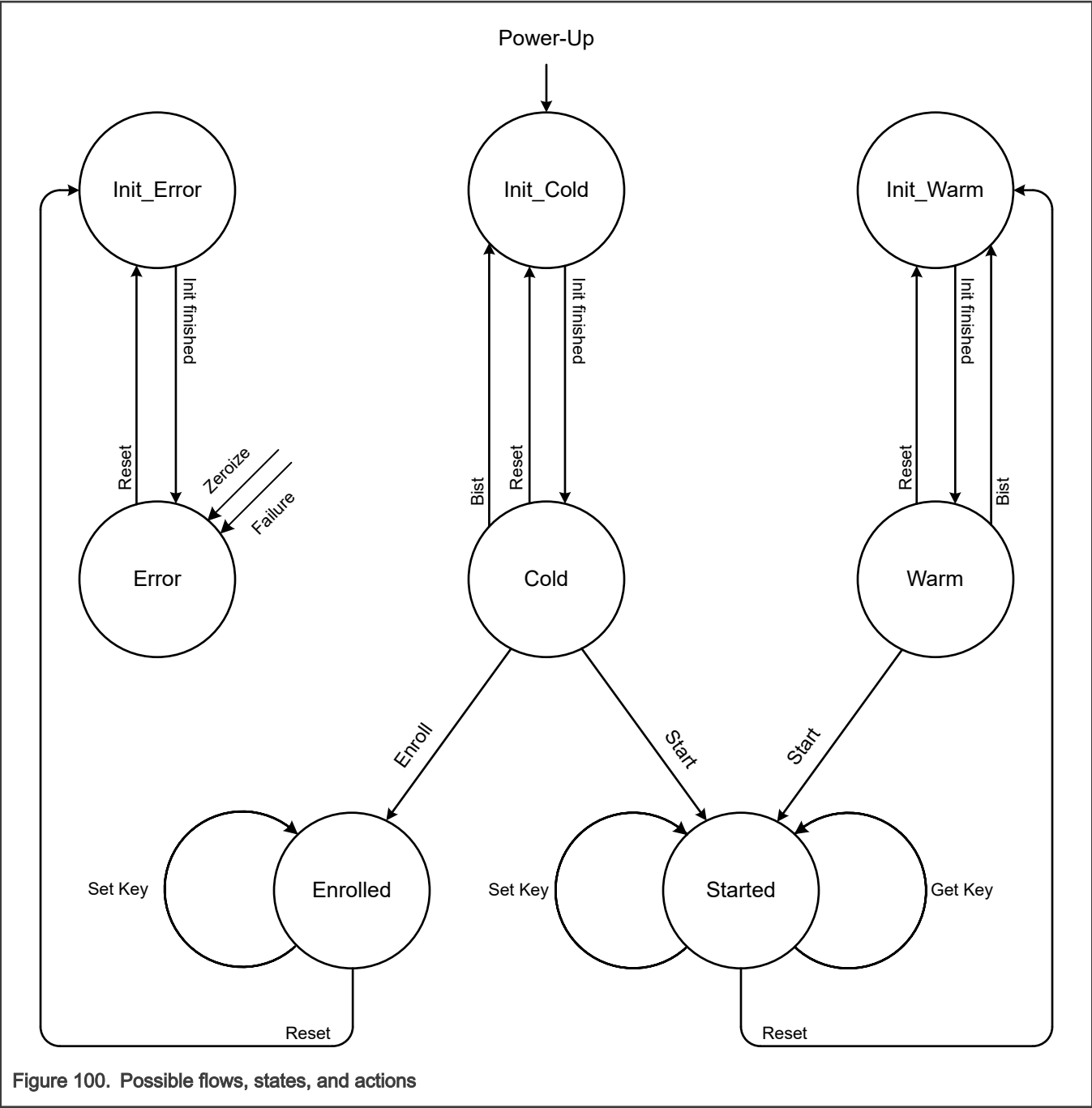
The enroll operation can only be performed when BLOCKENROLL = 0. If an error occurs during enrollment, the PUF controller goes to an error state.

After enrollment, only the Set Key operations can be performed. These operations can be performed repeatedly. When the device is reset from the enrolled state the PUF controller goes to the error state and no new operations can be performed. New operations can be performed only after re-powering the device.

After the start operation Set Key and Get Key operations can be performed. The Set Key and Get Key operations can be performed repeatedly. When the device is reset the Start operation must be performed again, before performing Get Key and Set Key operations.

NOTE

The Generate Key and Set Key operations can only be performed when BLOCKSETKEY = 0.



In case of a ZEROIZE operation through the control register or a failure (for example, a wrong activation code) the PUF controller goes to the error state, which erases all internal critical security parameters and disables communication with the PUF. The only way to leave this state is by repowering the device.

The indicative length of each operation is shown in the following table, assuming that data is available when requested and data can be accepted when presented by the PUF controller. The number of clock cycles may vary because of internal runtime variations, even when running the same operation with the same data.

Table 381. Number of clock cycles per operation

Operation	Number of clock cycles
Initialization	46.2 k
Enroll	17.1 k
Start	37.0 k
Generate Key (128-bit)	1.8 k
Generate Key (256-bit)	1.8 k
Set Key (128-bit)	2.0 k
Set Key (256-bit)	2.0 k
Get Key (128-bit)	2.1 k
Get Key (256-bit)	2.1 k

25.3.5.2 Activation code size

The size of the Activation Code (AC size) is:

AC size = 9536 bits (1192 bytes)

25.3.5.3 Key and code sizes

Keys are protected using the digital fingerprint, which has a 256-bit key strength. Longer keys can be stored for cryptographic purposes. For example, ECC keys up to 512-bits or RSA keys up to 4096-bit can be stored safely.

NOTE

Keys generated by the PUF controller are by construction randomly generated. This means that generated keys cannot be used for cryptographic algorithms that require keys with a specific mathematical structure, which is typical for public key schemes like RSA. In such cases, an externally Generated key should be used and stored as a user key.

The Key Code size (KC size in bits) depends on the key size and can be calculated as:

$KC\ size = 160 + \text{ceiling}_{256}(Key\ size)$

$\text{ceiling}_{256}()$ rounds up to the next multiple of 256

The following table specifies the KEYSIZE values to use for the supported key sizes, and the size of the related PUF-generated KC.

Table 382. Coding of KEYSIZE

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
000001	1	64	416	52
000010	2	128	416	52
000011	3	192	416	52

Table continues on the next page...

Table 382. Coding of KEYSIZE (continued)

KEYSIZE (5:0)	Value	Key size (bits)	KC size (bits)	KC size (bytes)
000100	4	256	416	52
000101	5	320	672	84
000110	6	384	672	84
000111	7	448	672	84
001000	8	512	672	84
001001	9	576	928	116
001010	10	640	928	116
001011	11	704	928	116
001100	12	768	928	116
001101	13	832	1184	148
001110	14	896	1184	148
001111	15	960	1184	148
010000	16	1024	1184	148
...	...	...		
100000	32	2048	2208	276
...	...	...		
110000	48	3072	3232	404
...	...	...		
000000	64	4096	4256	532

25.3.5.4 Key indexing

With the KEYIDX bits a key can be assigned a specific index value. This is done during the Set Key and Generate Key operations. The value of KEYIDX is part of the Key Code.

During key reconstruction the index defined in the Key Code is output on the KEYOUTINDEX[KEYOUTIDX] bits, which can be used to send the key to a specific target.

Keys with key index 0 are sent to the AES key interface. Keys with other indexes are sent to the KEYOUTPUT register.

Example:

Assume a key is intended to be used by a software AES encryption module that has number 0xA assigned to it. Before the Set Key operation the KEYINDEX[KEYIDX] field are set to 0xA. Then the Set Key operation is started and the key is passed to the PUF. The resulting Key Code (KC) is stored.

When the key is required, the Get Key operation is started and the KC is passed to PUF. The key index related to this KC appears in the KEYOUTINDEX[KEYOUTIDX] field and the resulting key appears in the KEYOUTPUT register. Using the KEYOUTIDX value the key can be sent to the AES target with number 0xA.

25.3.5.5 Key code header

The first 32 bits of the Key Code comprise a header with information about the type of key it represents. It includes three fields: Type, Index, and Size. The unused bits are always 0.

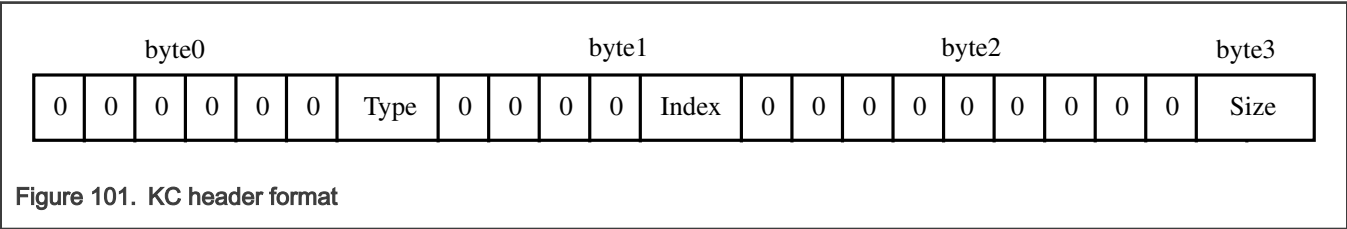
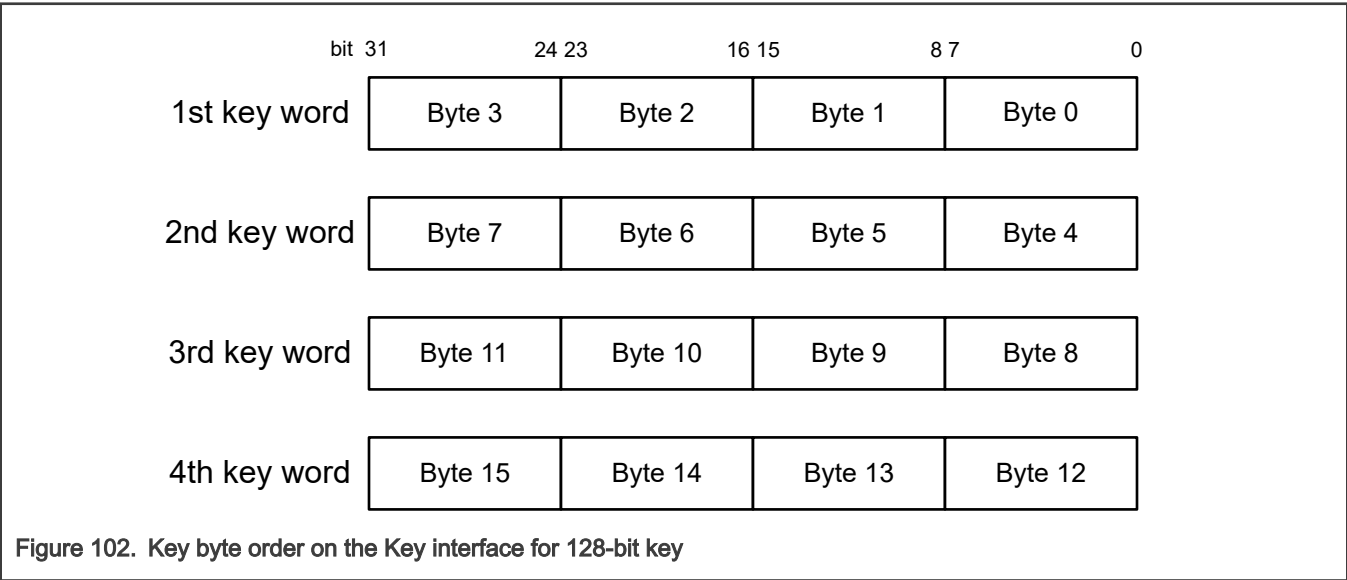


Table 383. KC header field description

Name	Description
Type	Define key type 11: Reserved 10: Reserved 01: Generate key 00: User key
Index	Value of KEYIDX at the moment of the Set Key or Generate Key operation
Size	Value of KEYSIZE at the moment of the Set Key or Generate Key command

25.3.5.6 Key byte order on the APB interface

The first word contains the first bytes and the second word the next bytes. The following figure shows the byte order.



25.3.5.7 Enroll

During the Enroll operation the SRAM startup values are read. Based on these a device-specific digital fingerprint is derived and the related activation code (AC) is generated. The following steps are performed:

1. Software gives the Enroll command to PUF by setting CTRL[ENROLL] = 1.
2. PUF reads the startup values and makes the AC available through CODEOUTPUT register.
3. After the operation is finished, STAT[BUSY] = 0 and STAT[SUCCESS] = 1, signaling that the enroll operation has completed successfully.

25.3.5.8 Start

During the start operation the SRAM startup values and the activation code are read. Based on these the PUF reconstructs the digital fingerprint. The following steps are performed:

1. Client Design gives the Start command to PUF through START.
2. PUF reads the startup values and requests for the AC through CODEINPUT.
3. After the operation is finished, STAT[BUSY] = 0 and STAT[SUCCESS] = 1, signaling that the Start operation has completed successfully. When STAT[ERROR] = 1, instead of STAT[SUCCESS] = 1, the provided activation code does not match the device. In this case the PUF is in the Error state. See [Figure 100](#) and [Error response](#).

25.3.5.9 Generate key

During the Generate key operation, the key size and key index are defined first and a device-specific key is generated. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Software sets the KEYINDEX[KEYIDX] and KEYSIZE[KEYSIZE] fields to their required values.
2. Software gives the Generate Key command to the PUF controller using CTRL[GENERATEKEY].
3. The PUF controller makes the KC available through the CODEOUTPUT register.
4. After the operation is finished, STAT[BUSY] = 0 and STAT[SUCCESS] = 1, signaling that the Generate key operation has completed successfully.

25.3.5.10 Set key

During this operation, the key size and key index are defined the first time that the user key is read. Based on this the device-specific Key Code (KC) is generated. The following steps are performed:

1. Set KEYINDEX[KEYIDX] and KEYSIZE[KEYSIZE] to their required values.
2. Give the Set User Key command to the PUF controller using CTRL[SETKEY].
3. Issue the request for the user key using KEYINPUT[KEYIN].
4. The KC is available through the CODEOUTPUT register.
5. After the operation is finished, STAT[BUSY] = 0 and STAT[SUCCESS] = 1, signaling that the Set Key operation has completed successfully.

NOTE

Keys with key index 0 are sent to the AES Key interface. When the user key index is 0, write the user key to the KEYINPUT register Least Significant word first. User Keys with other indexes should be written to the KEYINPUT register with Most Significant word first.

25.3.5.11 Get key

During Get key operation the key code is read. The key code includes the key index and key size; the values for KEYIDX and KEYSIZE are ignored. The following steps are performed:

1. Issue the Get Key command using the CTRL[GETKEY].
2. The key is available using the interface indicated in [Table 384](#). See [Key indexing](#) for more information on using KEYOUTIDX bits.
3. After the operation is finished STAT[BUSY] = 0 and STAT[SUCCESS] = 1, indicating that the Get key operation has completed successfully. If STAT[ERROR] = 1 instead of STAT[SUCCESS] = 1, the provided Key code does not match the device. In this case no key is provided and the PUF controller goes to the Error state. See [Figure 100](#) and [Error response](#).

NOTE

All key bits produced as defined in the KC must be consumed. If fewer key bits are consumed as defined in the KC, the PUF controller stays busy until the remaining bits are consumed.

Table 384. Key target interfaces per key index

Value of KEYINDEX during set key	Key output on
0	Dedicated interface: KEYINDEX = 0
Other	PUF Key Output (KEYOUTPUT) register

25.3.5.12 Zeroize

When CTRL[ZEROIZE] = 1, all internal critical security parameters are erased and the PUF controller goes to the error state. See [Figure 100](#). The device performs no new operations until repowered.

25.3.5.13 Error response

When an error other than an APB error is detected, all internal critical security parameters are erased and the PUF controller goes to the error state.

When the error occurs during a command (for example, when a wrong activation code or key code is given) or during initialization (for example, a reset was given after Zeroize instead of a repower), STAT[ERROR] = 1 and STAT[BUSY] = 0, independent of the state of the command signals.

25.3.5.14 Key index blocking

When an index blocking register is programmed, key output from the blocked index is not possible. However, the key code from the blocked index is still readable.

Index 1 - 7 and Index 8 - 15 are grouped together in the Index Block registers. Once index settings are done, the lock should be applied to disable modification until the next system reset.

25.4 Initialization

Initial configuration of the PUF can be accomplished as follows:

- Enable the clock to the PUF
- Clear the PUF peripheral reset
- Enable interrupts for wake-up from Deep-sleep mode

25.5 Software development

This section provides pseudocode drivers that implement the basic functionality to help software development. This is not intended to be optimal code. The code is based on the commands described in [PUF controller operations](#).

The software that interfaces with the PUF controller must read its status and drive the control bits. Also, it must provide input data to the PUF controller and accept output data from the PUF controller.

This section provides high-level code that can be used as a starting point for the development of the driver code. The example code uses status polling to control the flow.

NOTE

The status polling method is used for clarity. For more efficient operation, an interrupt-driven architecture is recommended.

After reset (with or without a power cycle of the PUF SRAM) the PUF controller is initialized, indicated by busy asserted. The system waits for initialization to be finished before it starts issuing commands. Use the function `wait_for_init` for this.

The code includes a ZEROIZE function, which can be used in case software detects a reason to delete sensitive data. For example, when an ERROR status is returned by the PUF controller functions.

The Function parameters table defines the parameters of the functions. The Data access functions table defines the data access functions; these must be supplied by the system.

Key formats are defined in section [Key byte order on the APB interface](#). It is assumed that the data and key are stored in memory using this format.

Table 385. Function parameters

Parameter	Description
ACdata	Pointer to a data structure that can store or contains the AC data and the current location in the data. When ACdata is generated it should be stored in some kind of NVM. When ACdata is requested it should be read from NVM.
KCdata	Pointer to a data structure that can store or contains the KC data and the current location in the data. When KCdata is generated it should be stored in some kind of NVM. When KCdata is requested it should be read from NVM.
KeyData	Pointer to a data structure that can store the Key data and the current location in the data.
KeySize	Size of the key in bits.
KeyIndex	Index for which the key is targeted.

Table 386. Data access functions

Variable	Description
Initialize(Target)	Empties the target data structure.
Get_data(Data, Source)	Retrieves the next data word from the Source structure, puts it in Data, and removes it from the head.
Append_data(Data, Target)	Appends the data in data to the end of target.

25.5.1 Pseudocode wait for initialization function

```
status wait_for_init() {
// wait until initialization has finished
while (*STAT & BUSY != 0) {} // check that initialization has passed
if (*STAT & (SUCCESS | ERROR) != SUCCESS) {
return ERROR
}
return OK
}
```

25.5.2 Pseudocode enroll function

```
status enroll(ACdata) {
    // clear the ACdata storage
    initialize(ACdata)
    // check if Enroll is allowed
    if (*ALLOW & ALLOWENROLL == 0) {
        return NOT_ALLOWED
    }
    // begin Enroll
    *CTRL = ENROLL
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
    }
    // while busy read AC
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEOUTAVAIL != 0) {
            tempData = *CODEOUTPUT
            append_data(tempData, ACdata)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

25.5.3 Pseudocode start function

```
status start(ACdata) {
    // check if Start is allowed
    if (*ALLOW & ALLOWSTART == 0) {
        return NOT_ALLOWED
    }
    // begin Start
    *CTRL = START
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {
    }
    // while busy send AC
    while (*STAT & BUSY != 0) {
        if (*STAT & CODEINREQ != 0) {
            get_data(tempData, ACdata)
            *CODEINPUT = tempData
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}
```

25.5.4 Pseudocode generate key function

```
status set_ik(KCdata, KeyIndex, KeySize) {
    // clear the KCdata storage
    initialize(KCdata)
    // check if Set Key is allowed
    if (*ALLOW & ALLOWSETKEY == 0) {
        return NOT_ALLOWED
    }
    // program the key size and index
    *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
    *KEYINDEX = KeyIndex
    // begin Set Key
    *CTRL = GENERATEKEY
}
```

```

// wait till command is accepted
while (*STAT & (BUSY | ERROR) == 0) {
}
// while busy read KC
while (*STAT & BUSY != 0) {
    if (*STAT & CODEOUTAVAIL != 0) {
        tempData = *CODEOUTPUT
        append_data(tempData, KCdata)
    }
} // while
// check result
if (*STAT & SUCCESS == 0) {
    return ERROR
}
return OK
}

```

25.5.5 Pseudocode set key function

```

status set_uk(KCdata, KeyIndex, UKdata) {
    // clear the KCdata storage
    initialize(KCdata)
    // check if Set Key is allowed
    if (*ALLOW & ALLOWSETKEY == 0) {
        return NOT_ALLOWED
    }
    // detect key size
    KeySize = length_in_bits
    // program the key size and index
    *KEYSIZE = KeySize >> 6 // convert to 64-bit blocks
    *KEYINDEX = KeyIndex
    // begin Set Key
    *CTRL = SETUSERKEY
    // wait till command is accepted
    while (*STAT & (BUSY | ERROR) == 0) {}
    // while busy write key and read KC
    while (*STAT & BUSY != 0) {
        if (*STAT & KEYINREQ != 0) {
            get_data(tempData, keyData)
            *KEYINPUT = tempData
        }
        if (*STAT & CODEOUTAVAIL != 0) {
            tempData = *CODEOUTPUT
            append_data(tempData, KCdata)
        }
    } // while
    // check result
    if (*STAT & SUCCESS == 0) {
        return ERROR
    }
    return OK
}

```

25.5.6 Pseudocode get key function

```

status get_key(KCdata, KeyIndex, KeyData) {
    // clear the KeyData storage
    initialize(KeyData)
    // put unused value in KeyIndex
    // Indicates key transfer via dedicated key interface
    KeyIndex = 255
    // check if Get Key is allowed
    if (*ALLOW & ALLOWGETKEY == 0) {
        return NOT_ALLOWED
    }
    // begin Get Key
    *CTRL = GETKEY
}

```

```

// wait till command is accepted
while (*STAT & (BUSY | ERROR) == 0) {
}
// while busy send KC, read key
while (*STAT & BUSY != 0) {
    if (*STAT & CODEINREQ != 0) {
        get_data(tempData, KCdata)
        *CODEINPUT = tempData
    }
    if (STAT & KEYOUTAVAIL != 0) {
        KeyIndex = *KEYOUTINDEX
        tempData = *KEYOUTPUT
        append_data(tempData, KeyData)
    }
} // while
// check result
if (*STAT & SUCCESS == 0) {
    return ERROR
}
return OK
}

```

25.5.7 Pseudocode Zeroize function

```

status zeroize() {
    // zeroize command is always allowed
    *CTRL = ZEROIZE
    // check that command is accepted
    if ((*STAT & ERROR == 0) || (*ALLOW != 0)) {
        return ERROR
    }
    return OK
}

```

25.6 Memory map and register definition

This section includes the PUF module memory map and detailed descriptions of all registers.

25.6.1 PUF Controller register descriptions

25.6.1.1 PUFCTRL memory map

PUF0 base address: 4000_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PUF Control (CTRL)	32	RW	0000_0000h
4h	PUF Key Index (KEYINDEX)	32	RW	0000_0000h
8h	PUF Key Size (KEYSIZE)	32	RW	0000_0000h
20h	PUF Status (STAT)	32	R	0000_0001h
28h	PUF Allow (ALLOW)	32	R	0000_0000h
40h	PUF Key Input (KEYINPUT)	32	W	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
44h	PUF Code Input (CODEINPUT)	32	W	0000_0000h
48h	PUF Code Output (CODEOUTPUT)	32	R	0000_0000h
60h	PUF Key Output Index (KEYOUTINDEX)	32	R	0000_0000h
64h	PUF Key Output (KEYOUTPUT)	32	R	0000_0000h
DCh	PUF Interface Status and Clear (IFSTAT)	32	RW	0000_0000h
100h	PUF Interrupt Enable (INTEN)	32	RW	0000_0000h
104h	PUF Interrupt Status (INTSTAT)	32	RW	0000_0000h
108h	PUF Power Control (PWRCTRL)	32	RW	0000_0001h
10Ch	PUF Configuration (CFG)	32	RW	0000_0000h
200h	Key Lock (KEYLOCK)	32	RW	0000_00AAh
204h	Key Enable (KEYENABLE)	32	RW	0000_0055h
208h	Key Reset (KEYRESET)	32	RW	0000_0000h
20Ch	Index Block Low (IDXBLK_L)	32	RW	8000_AAAAh
210h	Index Block High Duplicate (IDXBLK_H_DP)	32	RW	8000_AAAAh
214h - 220h	Key Mask a (KEY0MASK - KEY3MASK)	32	W	0000_0000h
254h	Index Block High (IDXBLK_H)	32	RW	8000_AAAAh
258h	Index Block Low Duplicate (IDXBLK_L_DP)	32	RW	8000_AAAAh

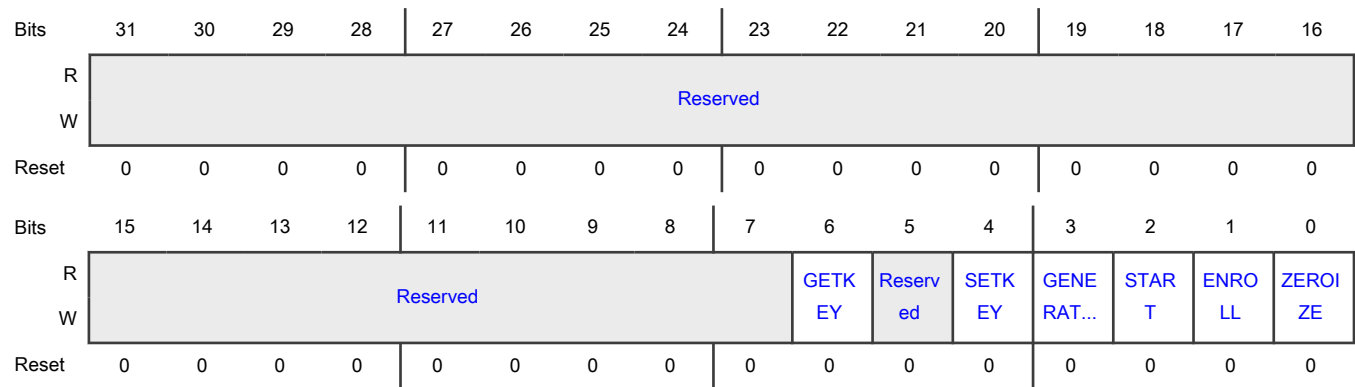
25.6.1.2 PUF Control (CTRL)

Offset

Register	Offset
CTRL	0h

Function

The PUF Control register defines which command must be executed next. The bits automatically revert to 0. Only one command bit may be written with 1 at a time, with the exception of ZEROIZE. Writing ZEROIZE with 1 takes precedence over all other commands.

Diagram**Fields**

Field	Function
31-7 —	Reserved. Read value is 0, only 0 should be written.
6 GETKEY	Get Key Begin Get Key operation
5 —	Reserved
4 SETKEY	Set Key Begin Set User Key operation
3 GENERATEKEY	Set Intrinsic Key Begin Set Intrinsic Key operation
2 START	Start Begin Start operation
1 ENROLL	Enroll Begin Enroll operation
0 ZEROIZE	Zeroize Begin Zeroize operation for Quiddikey and go to Error state

25.6.1.3 PUF Key Index (KEYINDEX)

Offset

Register	Offset
KEYINDEX	4h

Function

The PUF Key Index register defines the key index for the next set key operation.

Diagram

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

Fields

Field	Function
31-4 —	Reserved. Read value is 0, only 0 should be written.
3-0 KEYIDX	Key index for Set Key operations

25.6.1.4 PUF Key Size (KEYSIZE)

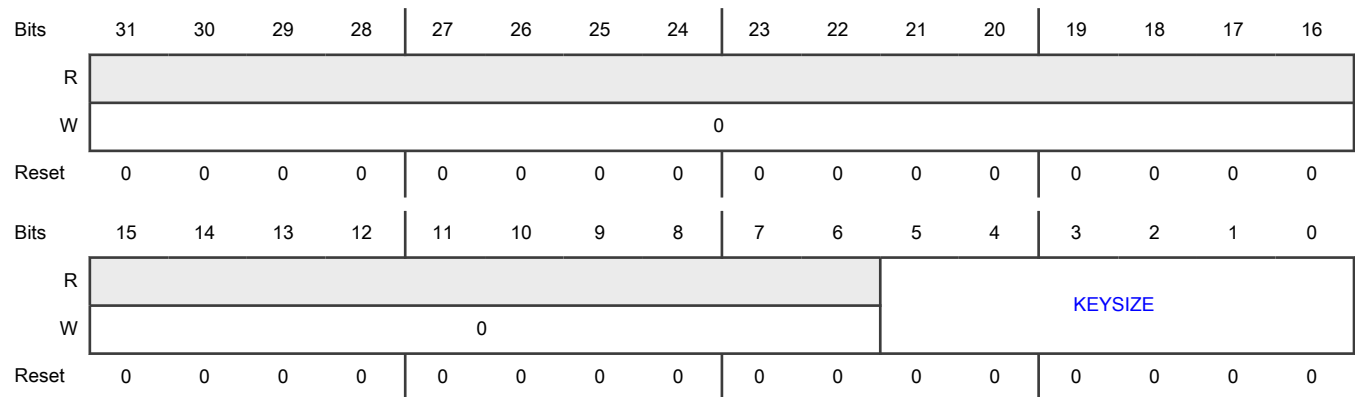
Offset

Register	Offset
KEYSIZE	8h

Function

The PUF Key Size register defines the key size for the next set key operation.

Coding of the KEYSIZE field is defined in [Key and code sizes](#).

Diagram**Fields**

Field	Function
31-6 —	Reserved. Read value is 0, only 0 should be written.
5-0 KEYSIZE	Key Size for Set Key operations

25.6.1.5 PUF Status (STAT)**Offset**

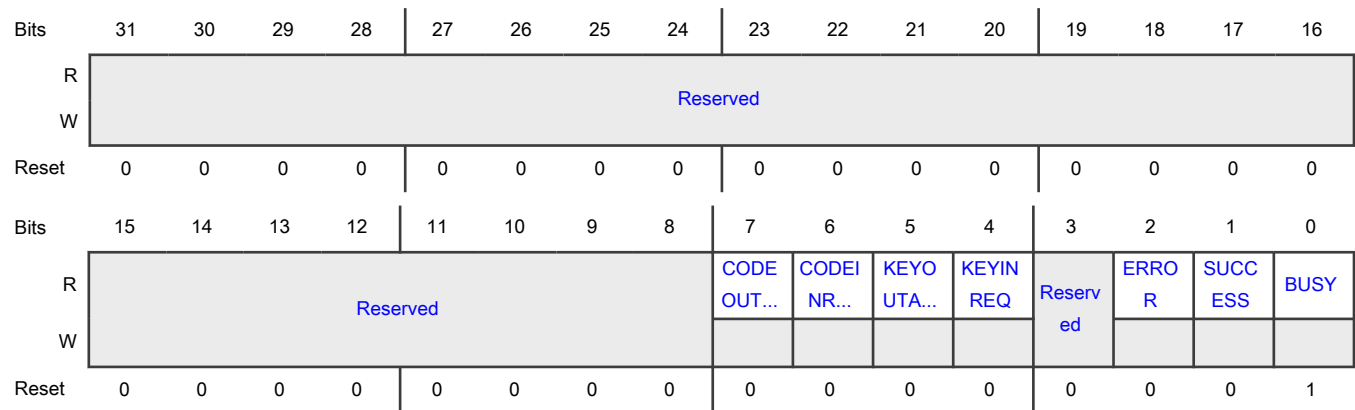
Register	Offset
STAT	20h

Function

The PUF Status register indicates the current status of the PUF and indicates which data is requested or available.

The indicated reset value is present immediately after reset. After the PUF finishes initialization, BUSY = 0 and depending on the state of the PUF, either SUCCESS = 1 or ERROR = 1. See [Order of operations](#).

Diagram



Fields

Field	Function
31-8 —	Reserved
7 CODEOUTAVAIL	Code Out Available Next part of AC/KC is available
6 CODEINREQ	Code In Request Request for next part of AC/KC
5 KEYOUTAVAIL	Key Out Available Next part of key is available
4 KEYINREQ	Key In Request Request for next part of key
3 —	Reserved
2 ERROR	Error PUF is in the Error state and no operations can be performed
1 SUCCESS	Success Last operation was successful
0 BUSY	Busy Indicates that operation is in progress

25.6.1.6 PUF Allow (ALLOW)

Offset

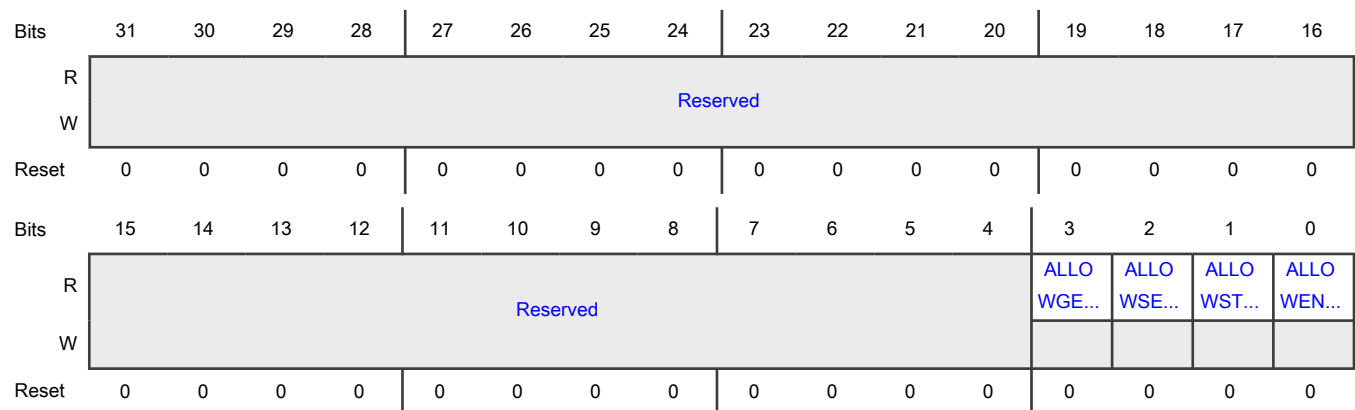
Register	Offset
ALLOW	28h

Function

The PUF Allow register indicates which operations are currently allowed.

The indicated reset value is present immediately after reset. After the PUF finishes initialization, one or more bits of this register goes to 1.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 ALLOWGETKEY	Allow Get Key Get Key operation is allowed
2 ALLOWSETKEY	Allow Set Key Set Key and Generate Key operations are allowed
1 ALLOWSTART	Allow Start Start operation is allowed
0 ALLOWENROLL	Allow Enroll Enroll operation is allowed

25.6.1.7 PUF Key Input (KEYINPUT)

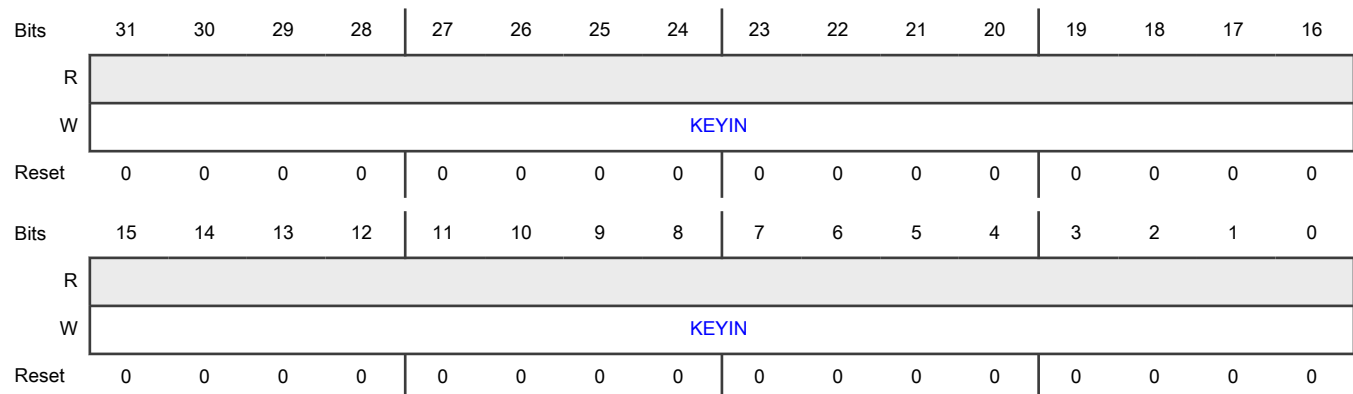
Offset

Register	Offset
KEYINPUT	40h

Function

The PUF reads the key that must be stored during the Set Key operation using the PUF Key Input register.

Diagram



Fields

Field	Function
31-0	Key Input Data
KEYIN	KEYIN must only be written when STAT[KEYINREQ] = 1.

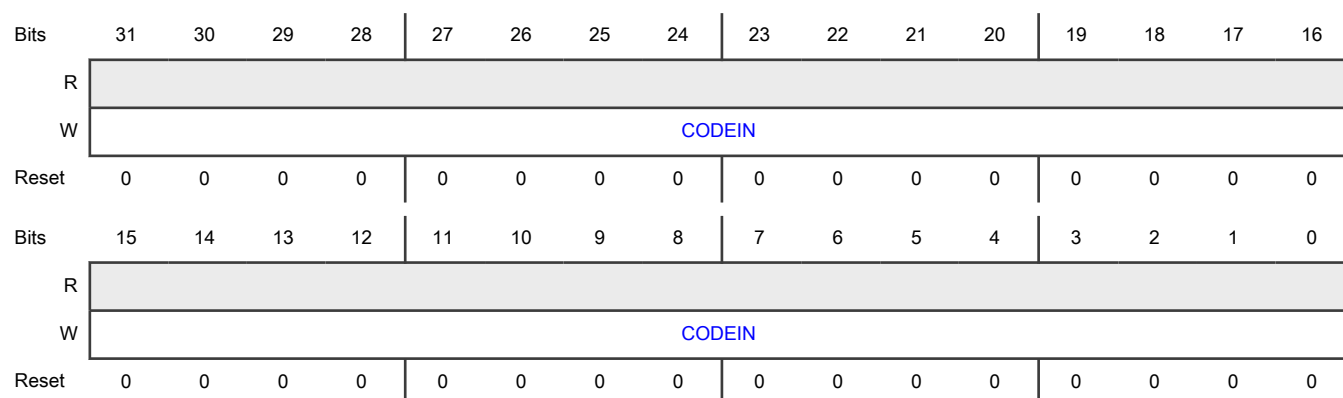
25.6.1.8 PUF Code Input (CODEINPUT)

Offset

Register	Offset
CODEINPUT	44h

Function

The PUF reads the AC (in case of a start operation) or the KC (in case of a Get Key operation) using the PUF Code Input register.

Diagram**Fields**

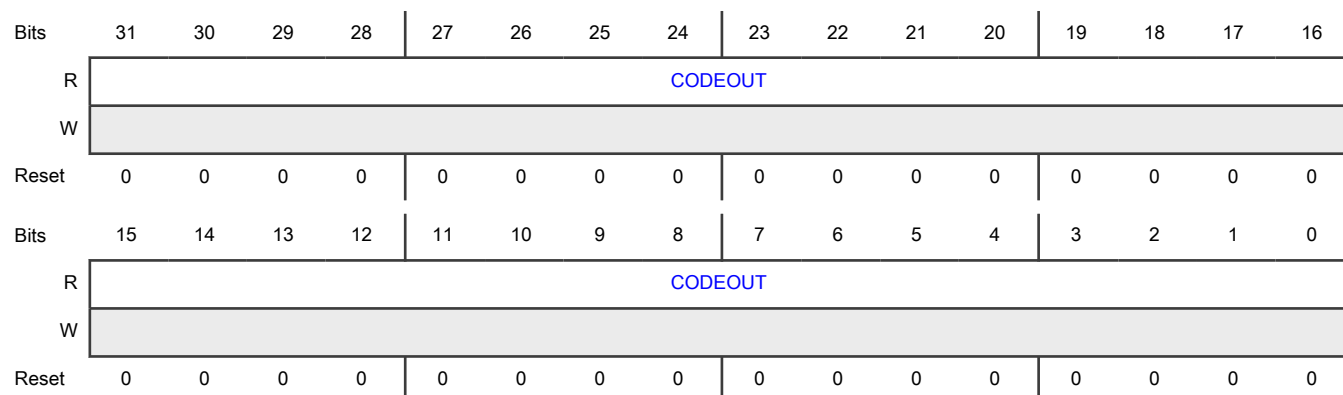
Field	Function
31-0	AC/KC Input Data
CODEIN	CODEIN must only be written when STAT[CODEINREQ] = 1.

25.6.1.9 PUF Code Output (CODEOUTPUT)**Offset**

Register	Offset
CODEOUTPUT	48h

Function

The PUF provides the AC (in case of an enroll operation) or KC (in case of a Set Key or Generate Key operation) using the PUF Code Output register.

Diagram

Fields

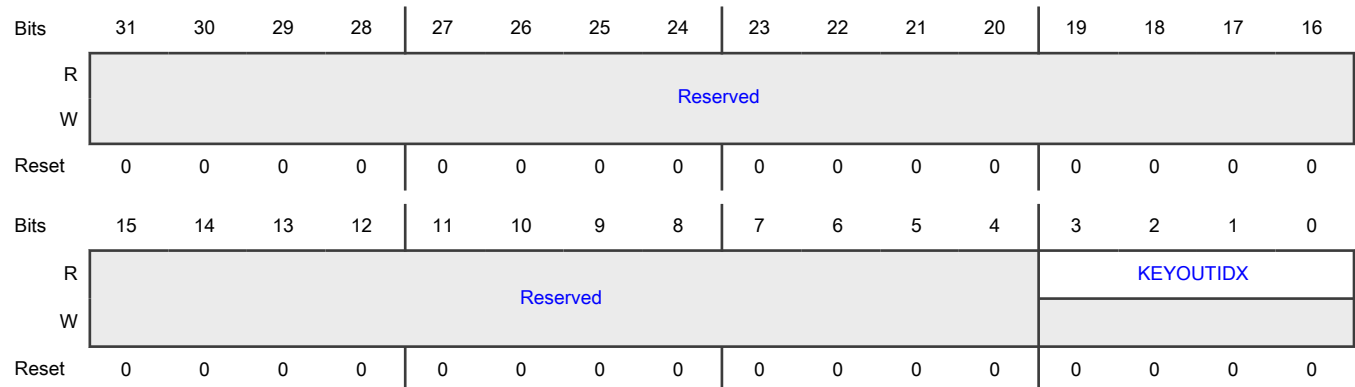
Field	Function
31-0 CODEOUT	AC/KC Output Data CODEOUT must only be written when STAT[CODEOUTAVAIL] = 1.

25.6.1.10 PUF Key Output Index (KEYOUTINDEX)**Offset**

Register	Offset
KEYOUTINDEX	60h

Function

The key index of the reconstructed key can be read using the PUF Key Output Index register.

Diagram**Fields**

Field	Function
31-4 —	Reserved
3-0 KEYOUTIDX	Key Output Index Key index for the key that is currently output via the Key Output register

25.6.1.11 PUF Key Output (KEYOUTPUT)

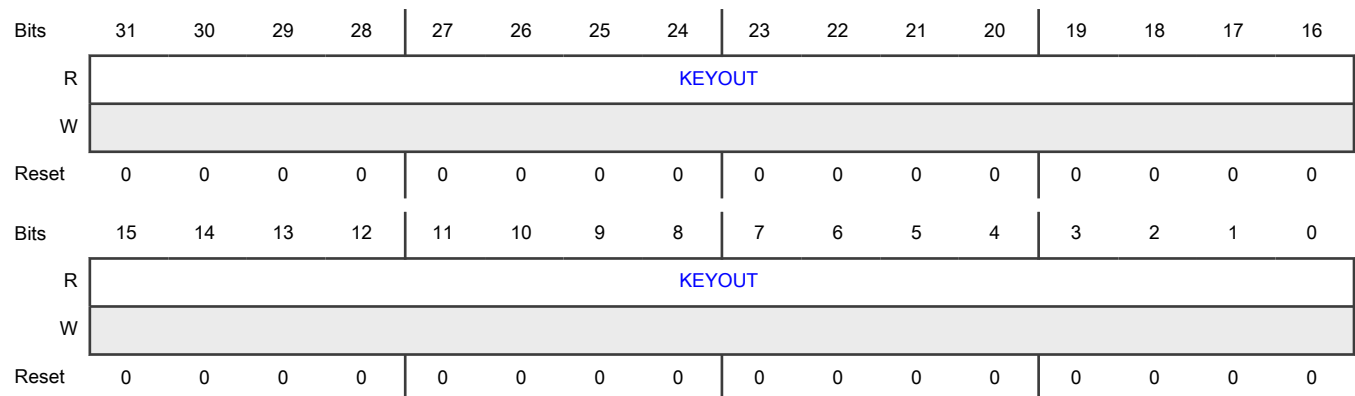
Offset

Register	Offset
KEYOUTPUT	64h

Function

The reconstructed key can be read using the PUF Key Output register.

Diagram



Fields

Field	Function
31-0	Key Output Data
KEYOUT	KEYOUT must only be read when STAT[KEYOUTAVAIL] = 1.

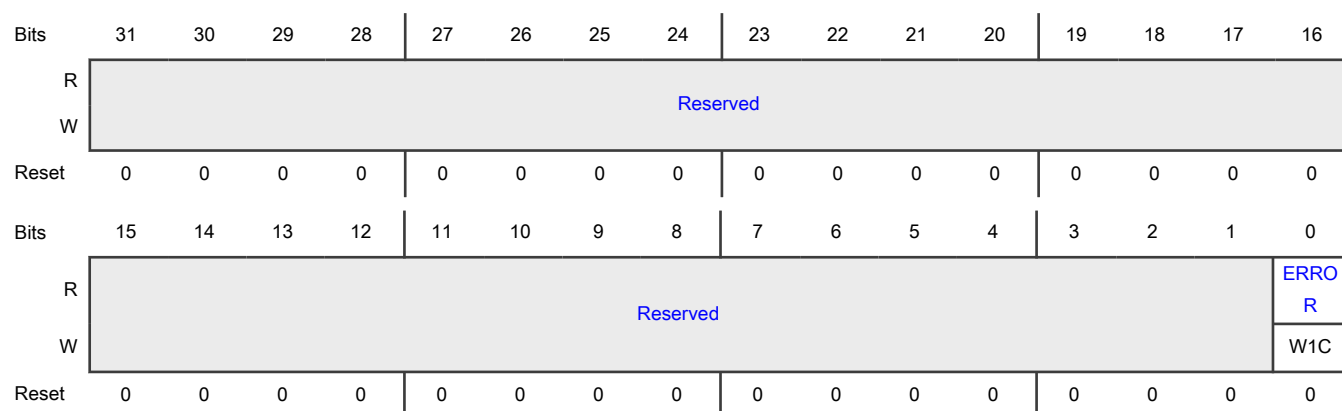
25.6.1.12 PUF Interface Status and Clear (IFSTAT)

Offset

Register	Offset
IFSTAT	DCh

Function

The status of the APB interface can be monitored with the PUF Interface Status register. IFSTAT has the same address as IFSTATCLR.

Diagram**Fields**

Field	Function
31-1 —	Reserved
0 ERROR	Error Indicates that any of the following errors have occurred: • Write to a non-existing register • Read from a non-existing register • Write to a read-only register • Read from a write-only register • KEYINPUT register is written when no key is requested (KEYINREQ). • CODEINPUT register is written when no AC/KC is requested (CODEINREQ = 0). • CODEOUTPUT register is read when no AC/KC is available (CODEOUTAVAI = 0). • KEYOUTPUT register is read when no key is available (KEYOUTAVAIL = 0). • KEYOUTPUTINDEX register is read when no key is available (KEYOUTAVAIL = 0). • Multiple commands are written at the same time to the PUF control register. • A command is written that is not allowed. Writing 1 clears the ERROR flag.

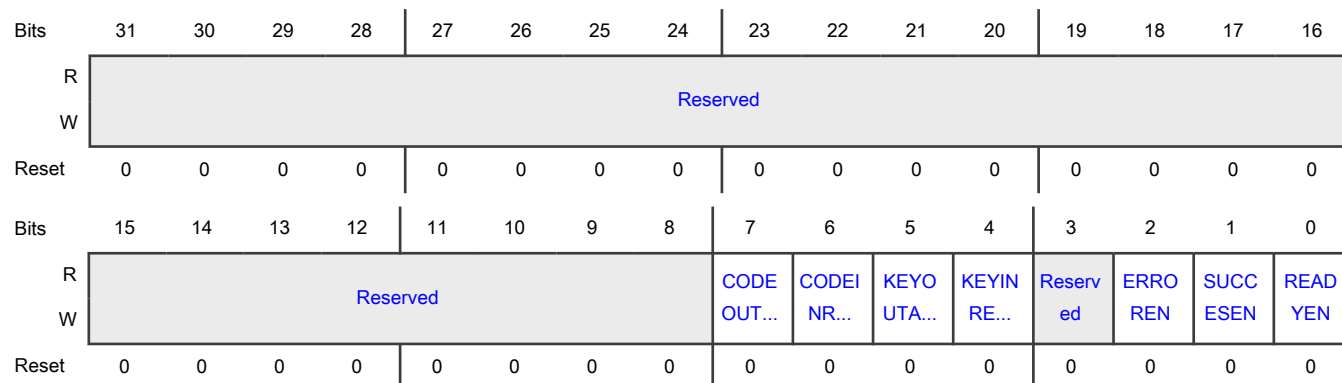
25.6.1.13 PUF Interrupt Enable (INTEN)**Offset**

Register	Offset
INTEN	100h

Function

The PUF Interrupt Enable register is used to enable various PUF controller interrupt sources. Enable bits in INTEN are mapped to locations that correspond to the flags in the STAT register.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 CODEOUTAVAIL ILEN	Enable corresponding interrupt in STAT, which is next part of AC/KC is available.
6 CODEINREQE N	Enable corresponding interrupt in STAT, which is request for next part of AC/KC.
5 KEYOUTAVAIL EN	Enable corresponding interrupt in STAT, which is next part of key is available.
4 KEYINREQEN	Enable corresponding interrupt in STAT, which is request for next part of key.
3 —	Reserved
2 ERRORREN	Enable corresponding interrupt in STAT, which indicates that PUF is in the error state and no operations can be performed.
1 SUCCESEN	Enable corresponding interrupt in STAT, which indicates last operation was successful.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 READYEN	Enable corresponding interrupt in STAT, which indicates that the initialization or a operation is completed.

25.6.1.14 PUF Interrupt Status (INTSTAT)

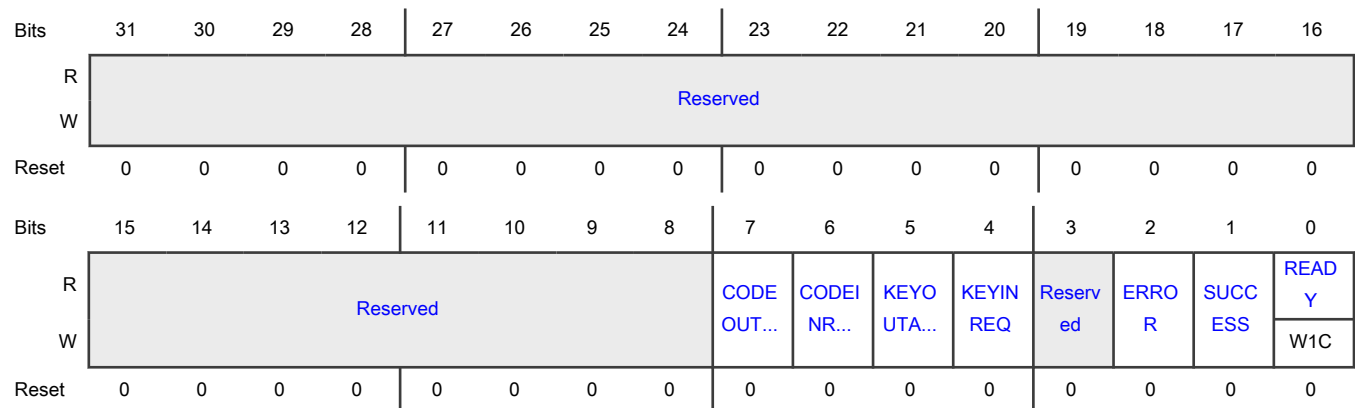
Offset

Register	Offset
INTSTAT	104h

Function

The PUF interrupt status register provides a view of interrupt flags that are currently enabled.

Diagram



Fields

Field	Function
31-8 —	Reserved
7 CODEOUTAVAIL	Code Out Available Next part of AC/KC is available. Cleared when interrupt source clears.
6 CODEINREQ	Code In Request Request for next part of AC/KC. Cleared when interrupt source clears.
5	Key Out Available

Table continues on the next page...

Table continued from the previous page...

Field	Function
KEYOUTAVAIL	Next part of key is available. Cleared when interrupt source clears.
4 KEYINREQ	Key In Request Request for next part of key. Cleared when interrupt source clears.
3 —	Reserved
2 ERROR	Error PUF is in the error state (for example, an incorrect key code, or Zeroization) and no operations can be performed.
1 SUCCESS	Success Last operation was successful. Cleared when interrupt source clears.
0 READY	Ready Indicates that the initialization or a operation is completed. Write 1 to clear.

25.6.1.15 PUF Power Control (PWRCTRL)

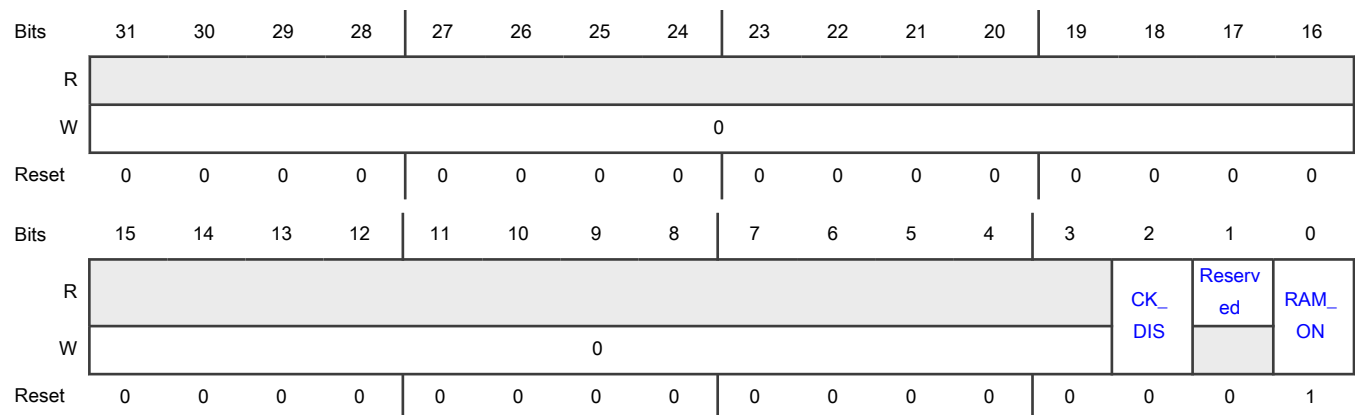
Offset

Register	Offset
PWRCTRL	108h

Function

The PUF power register controls the power of the dedicated SRAM used by PUF controller.

Diagram



Fields

Field	Function
31-3 —	Reserved. Read value is undefined, only 0 should be written.
2 CK_DIS	PUF RAM Clock Disable CK_DIS controls the PUF RAM Clock enable/disable. 0b - PUF RAM clock is disabled. 1b - PUF RAM clock is enabled.
1 —	Reserved. Read value is 0, only 0 should be written.
0 RAM_ON	RAM Power On RAM_ON is cleared by hardware when entering a low power mode. 0b - Power Off 1b - Power On

25.6.1.16 PUF Configuration (CFG)

Offset

Register	Offset
CFG	10Ch

Function

The PUF Configuration register allows blocking of enrollment and additional key code generations.

Diagram

[illegible]

Fields

Field	Function
31-2 —	Reserved Read value is undefined, only 0 should be written.
1 BLOCKKEYOUT TPUT	Block Key Output Data If BLOCKKEYOUTPUT = 0, then key out will not be blocked, even if key output index = 15. If BLOCKKEYOUTPUT = 1, it blocks key output (KEYOUTPUT[KEYOUT] = 0) when key output index = 15. 0b - Disabled. BLOCKKEYOUTPUT is cleared on reset. 1b - Enabled
0 BLOCKENROL L_SETKEY	Block Enroll and Set Key Operation Write 1 to set, cleared on reset. 0b - Disabled 1b - Enabled

25.6.1.17 Key Lock (KEYLOCK)

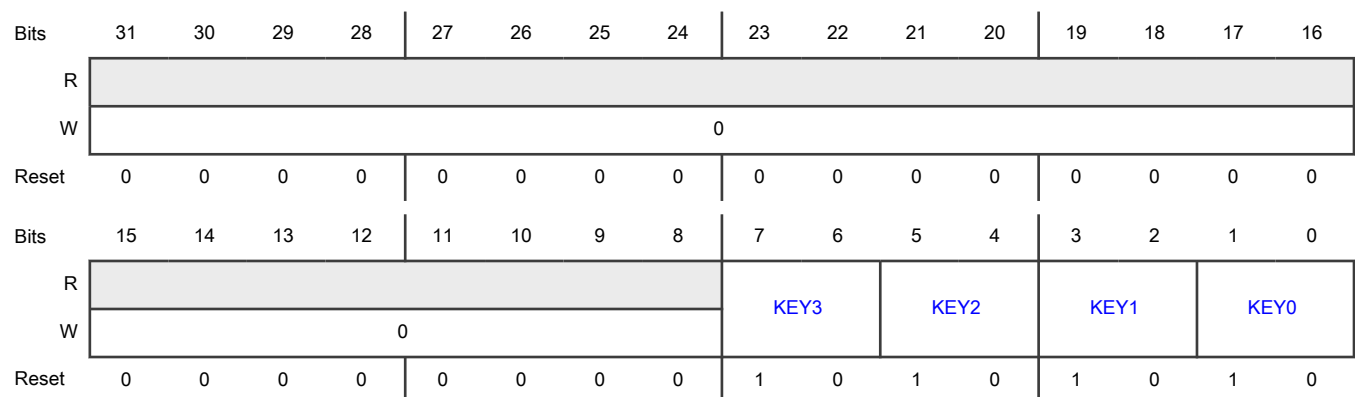
Offset

Register	Offset
KEYLOCK	200h

Function

The PUF Key Lock register allows locking write access to a set of registers associated with a given key in the Key management module. Using this feature, users have the option of locking the key settings once key loading is completed.

Diagram



Fields

Field	Function
31-8 —	Reserved Read value is undefined, only 0 should be written.
7-6 KEY3	Key 3 NOTE Once this field is written with a value different from '10', its value can no longer be modified until Power-On-Reset occurs. 00b,01b,11b - Write access to KEY3MASK, KEYENABLE[KEY3] and KEYRESET[KEY3] is NOT allowed. 10b - Write access to KEY3MASK, KEYENABLE[KEY3] and KEYRESET[KEY3] is allowed.
5-4 KEY2	Key 2 NOTE Once this field is written with a value different from '10', its value can no longer be modified until Power-On-Reset occurs. 00b,01b,11b - Write access to KEY2MASK, KEYENABLE[KEY2] and KEYRESET[KEY2] is NOT allowed. 10b - Write access to KEY2MASK, KEYENABLE[KEY2] and KEYRESET[KEY2] is allowed.
3-2 KEY1	Key 1 NOTE Once this field is written with a value different from '10', its value can no longer be modified until Power-On-Reset occurs. 00b,01b,11b - Write access to KEY1MASK, KEYENABLE[KEY1] and KEYRESET[KEY1] is NOT allowed. 10b - Write access to KEY1MASK, KEYENABLE[KEY1] and KEYRESET[KEY1] is allowed.
1-0 KEY0	Key 0 NOTE Once this field is written with a value different from '10', its value can no longer be modified until Power-On-Reset occurs. 00b,01b,11b - Write access to KEY0MASK, KEYENABLE[KEY0] and KEYRESET[KEY0] is NOT allowed. 10b - Write access to KEY0MASK, KEYENABLE[KEY0] and KEYRESET[KEY0] is allowed.

25.6.1.18 Key Enable (KEYENABLE)

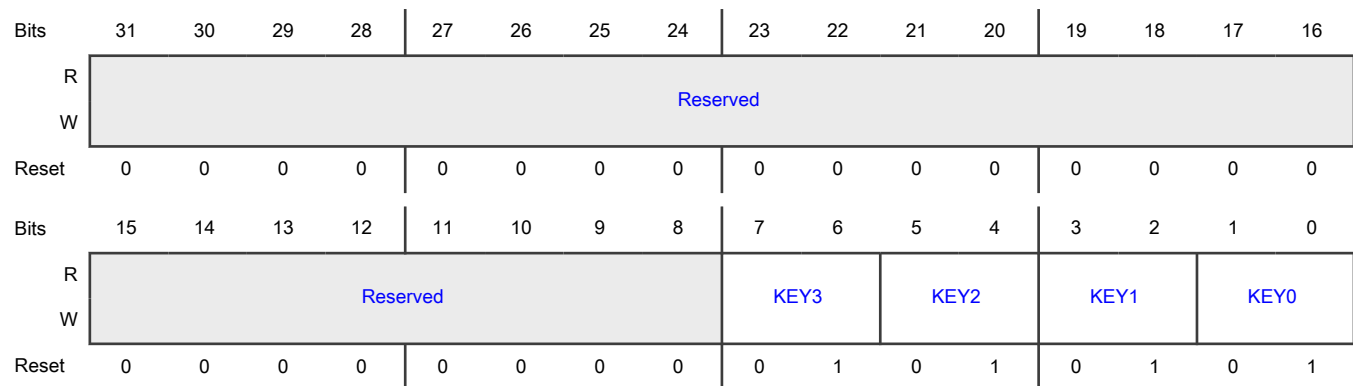
Offset

Register	Offset
KEYENABLE	204h

Function

The PUF Key Enable register allows the user to load PUF output as a secret key to a particular engine.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-6 KEY3	Key 3 00b,01b,11b - Data coming from the PUF Index 0 interface are NOT shifted in the KEY3 register. 10b - Data coming from the PUF Index 0 interface are shifted in the KEY3 register.
5-4 KEY2	Key 2 00b,01b,11b - Data coming from the PUF Index 0 interface are NOT shifted in the KEY2 register. 10b - Data coming from the PUF Index 0 interface are shifted in the KEY2 register.
3-2 KEY1	Key 1 00b,01b,11b - Data coming from the PUF Index 0 interface are NOT shifted in the KEY1 register. 10b - Data coming from the PUF Index 0 interface are shifted in the KEY1 register.
1-0 KEY0	Key 0 00b,01b,11b - Disabled. Disabled. Data coming from the PUF Index 0 interface are NOT shifted in the KEY0 register. 10b - Enabled. Enabled. Data coming from the PUF Index 0 interface are shifted in the KEY0 register.

25.6.1.19 Key Reset (KEYRESET)

Offset

Register	Offset
KEYRESET	208h

Function

The PUF Key Reset register allows the user to reset the Hold register that holds an individual key as well as the associated field in the SHIFT_STATUS register.

Diagram

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

Fields

Field	Function
31-8 —	Reserved
7-6 KEY3	Key 3 Self clearing. Must be done before loading any new key. 10b - Reset KEY3 Hold register and SHIFT_STATUS[KEY3].
5-4 KEY2	Key 2 Self clearing. Must be done before loading any new key. 10b - Reset KEY2 Hold register and SHIFT_STATUS[KEY2].
3-2 KEY1	Key 1 Self clearing. Must be done before loading any new key. 10b - Reset KEY1 Hold register and SHIFT_STATUS[KEY1].
1-0 KEY0	Key 0 Self clearing. Must be done before loading any new key. 10b - Reset KEY0 Hold register and SHIFT_STATUS[KEY0].

25.6.1.20 Index Block Low (IDXBLK_L)

Offset

Register	Offset
IDXBLK_L	20Ch

Function

The PUF index blocking register allows user to block a given index from PUF index 1 - 7. With the IDXn bit set, Key output from that index is not available on the APB register interface. Index blocking would be activated if relevant key fields in IDXBLK_L and IDXBLK_H_DP do not match. For example, IDX2 would only be accessible if IDX2 = 0x2 in both the IDXBLK_L and IDXBLK_H_DP registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	LOCK_IDX															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	IDX7		IDX6		IDX5		IDX4		IDX3		IDX2		IDX1		Reserved	
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Fields

Field	Function
31-30 LOCK_IDX	Lock Index Lock 0 to 7 PUF key indexes.
29-16 —	Reserved
15-14: IDX7 13-12: IDX6 11-10: IDX5 9-8: IDX4 7-6: IDX3 5-4: IDX2 3-2: IDX1	Index n Blocks PUF index n 00b, 11b - Reserved 01b - PUF index is blocked. 10b - PUF index is accessible.
1-0 —	Reserved

25.6.1.21 Index Block High Duplicate (IDXBLK_H_DP)

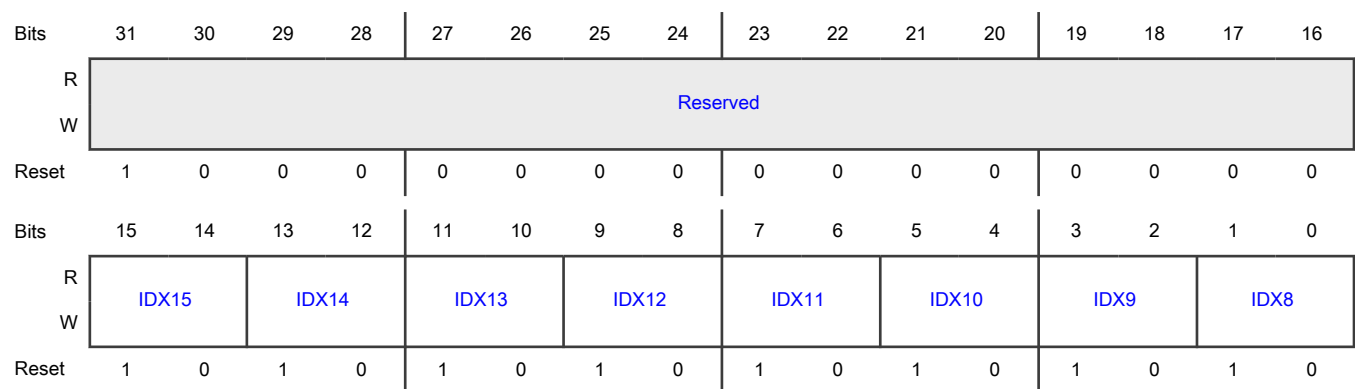
Offset

Register	Offset
IDXBLK_H_DP	210h

Function

IDXBLK_H_DP is a duplicate of IDXBLK_L register and provides protection against malicious attacks. Index blocking would be activated if relevant key fields in IDXBLK_L and IDXBLK_H_DP do not match. For example, IDX12 would only be accessible if $IDX12 = 0x2$ in both IDXBLK_L and IDXBLK_H_DP registers.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-14: IDX15	Index n
13-12: IDX14	Blocks PUF index n
11-10: IDX13	00b, 11b - Reserved
9-8: IDX12	01b - PUF index is blocked.
7-6: IDX11	10b - PUF index is accessible.
5-4: IDX10	
3-2: IDX9	
1-0: IDX8	

25.6.1.22 Key Mask a (KEY0MASK - KEY3MASK)

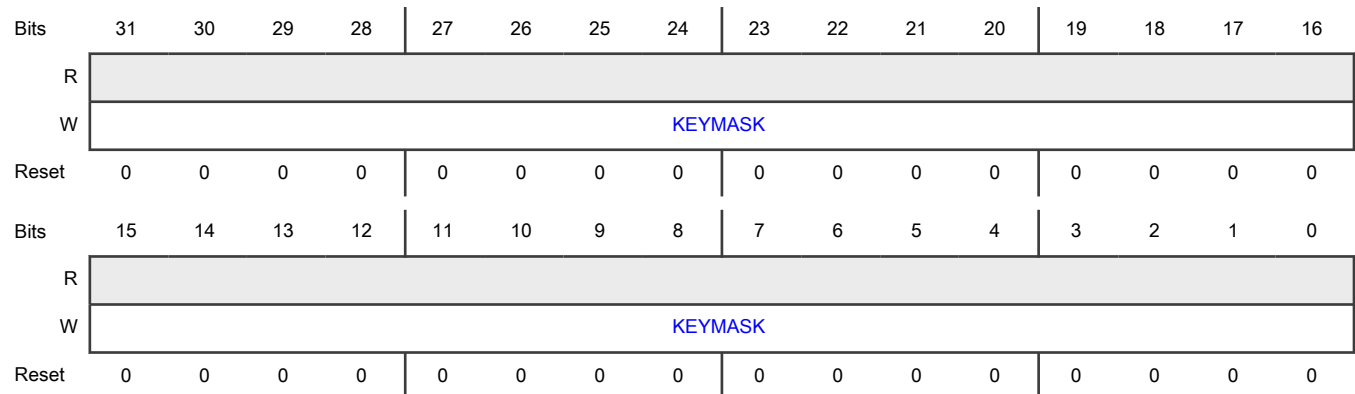
Offset

Register	Offset
KEY0MASK	214h
KEY1MASK	218h
KEY2MASK	21Ch
KEY3MASK	220h

Function

This registers has additional protection against Side Channel analysis. It obscures the secret key value stored in the Key Hold registers. A random value can be loaded into this register.

Diagram



Fields

Field	Function
31-0	Key a Mask
KEYMASK	Reserved

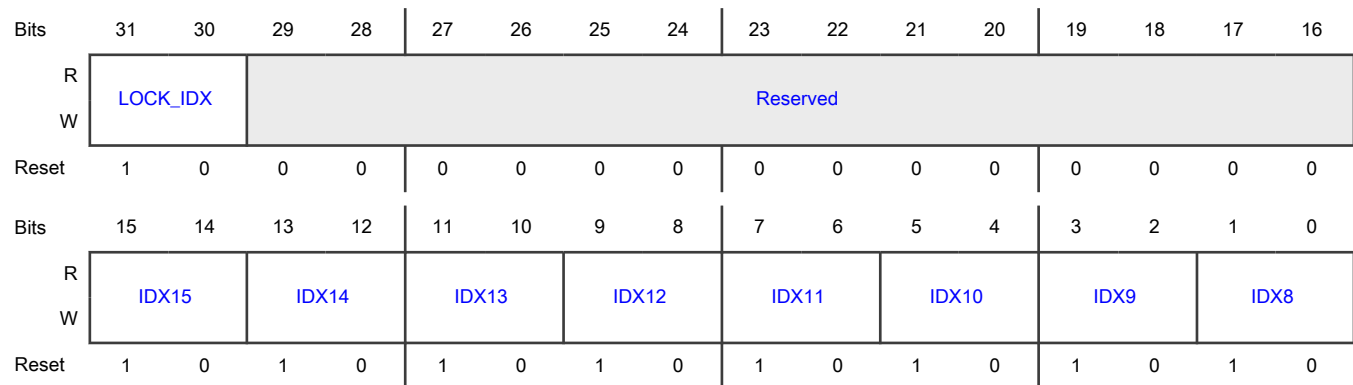
25.6.1.23 Index Block High (IDXBLK_H)

Offset

Register	Offset
IDXBLK_H	254h

Function

The IDXBLK_H register allows the user to block a given index from PUF index 8-15. With IDXn bit set, Key output from that index is not available on APB register interface.

Diagram**Fields**

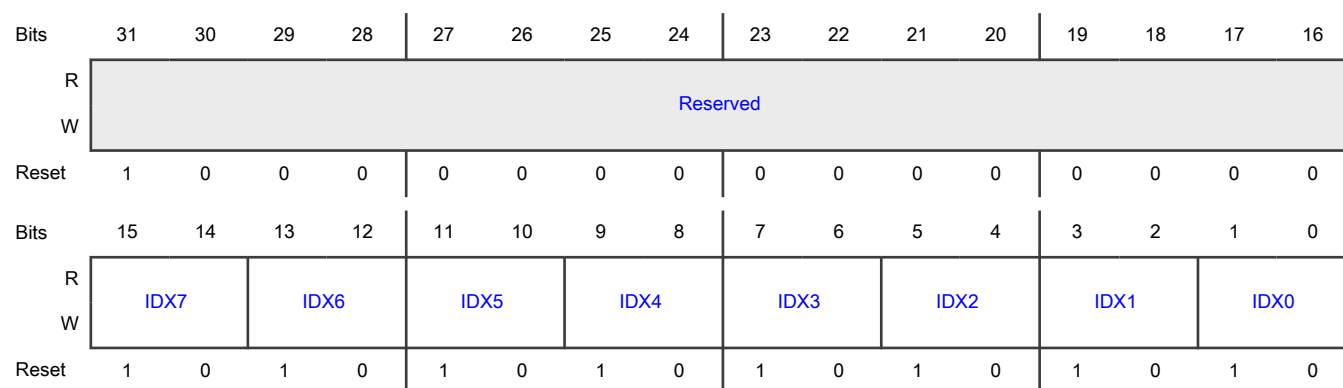
Field	Function
31-30 LOCK_IDX	Lock Index Lock 8 - 15 PUF key indexes.
29-16 —	Reserved
15-14: IDX15	Index n
13-12: IDX14	Status block index n
11-10: IDX13	00b, 11b - Reserved
9-8: IDX12	01b - PUF index is blocked.
7-6: IDX11	10b - PUF index is accessible.
5-4: IDX10	
3-2: IDX9	
1-0: IDX8	

25.6.1.24 Index Block Low Duplicate (IDXBLK_L_DP)**Offset**

Register	Offset
IDXBLK_L_DP	258h

Function

IDXBLK_L_DP is a duplicate of the IDXBLK_L register and provides protection against malicious attacks. Index blocking would be activated if a relevant key fields in IDXBLK_L and IDXBLK_L_DP do not match. For example, IDX4 would only be accessible if IDX4 = 0x2 in both IDXBLK_L and IDXBLK_L_DP registers.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-14: IDX7	Index n
13-12: IDX6	Blocks PUF index n
11-10: IDX5	00b, 11b - Reserved
9-8: IDX4	01b - PUF index is blocked.
7-6: IDX3	10b - PUF index is accessible.
5-4: IDX2	
3-2: IDX1	
1-0: IDX0	

Chapter 26

HASH-AES

26.1 Chip-specific HASH-AES information

Table 387. Reference links to related information

Topic	Related module	Reference
Full description	HASH-AES	HASH-AES
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

26.1.1 Chip-Specific initialization

Initial configuration of the SHA engine can be accomplished as follows:

- Enable the clock to the SHA engine in the PSCCTL0 register in CLKCTL0 register descriptions. This enables the register interface and the peripheral function clock.
- Clear the SHA engine peripheral reset in the PRSTCTL0 register by writing to the PRSTCTL0_CLR register (see RSTCTL0 register descriptions).
- The SHA engine provides an interrupt to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable this in the STARTEN1 register in SYSCTL0 register descriptions.
- The SHA engine provides an input DMA request to the DMA controller.
- The SHA engine output DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers.
- The priority for the SHA engine can be set in the AHBMATRIXPRIOR[M8] field in SYSCTL0 register descriptions.

26.1.2 Module instances

This device has only one instance of the HASH-AES module, HASHCRYPT0.

26.2 AES engine

The AES engine supports 128 bit, 192 bit, or 256 bit keys for encryption and decryption operations. An on-chip SCA (Side Channel Analysis) resistant hardware accelerator for AES encryption and decryption that conforms to the FIPS 197 standard, to protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin.

26.2.1 AES Features

- Encryption and decryption of data
- Secure storage of AES key that cannot be read
- AES engine peak performance of 0.5 bytes/clock cycle
- AES engine supports 128 bit, 192 bit or 256 bit key in:
 - Electronic Code Book (ECB) mode
 - Cipher Block Chaining (CBC) mode
 - Counter (CTR) mode

- Side Channel Analysis resistance, special SCA protection techniques implemented in the engine.
 - Security against Side Channel Analysis (Power and Electro Magnetic traces) using masking techniques to protect against DPA (Differential Power Analysis), CPA (correlation Power Analysis) and template attacks.
- The AES engine supports 128-bit key in ICB (Indexed Code Book) mode, that offers protection against side-channel attacks.
- The AES engine is compliant with the FIPS (Federal Information Processing Standard) Publication 197, Advanced Encryption Standard (AES).
- AES offers programmability to select little-endian or big-endian mode of operation
- The AES engine may use the processor, DMA or AHB Master for data movement.
 - The AHB Master may only be used to load data.
 - The DMA may be used to read-out results. DMA-based result reading is a "trigger", so the application must set the size correctly.

26.2.2 Functional description

The AES, a block cipher, encrypts and decrypts the user provided 128-bit block data Electronic Code Book (ECB) model.

In ECB mode the application provides a 128-bit input block and the AES encrypts or decrypts that input into a 128 bit output block.

In CBC and CTR, the other cipher modes supported by this module the application first provides an IV of the 128 bits, and then provides the 128-bit data blocks. The peripheral then XOR the data with the appropriate component.

The AES engine has an output/digest buffer of 256 bits, and two 512-bit buffers.

- The first buffer holds the key of any of three sizes: the IV/nonce of CBC or IV + counter of CTR, and the message block itself.
- The second buffer may be the output and a cache of the message for CBC decrypt. CBC decrypt needs to XOR in the message at the end, so it is held to allow the next message buffer to be copied in.

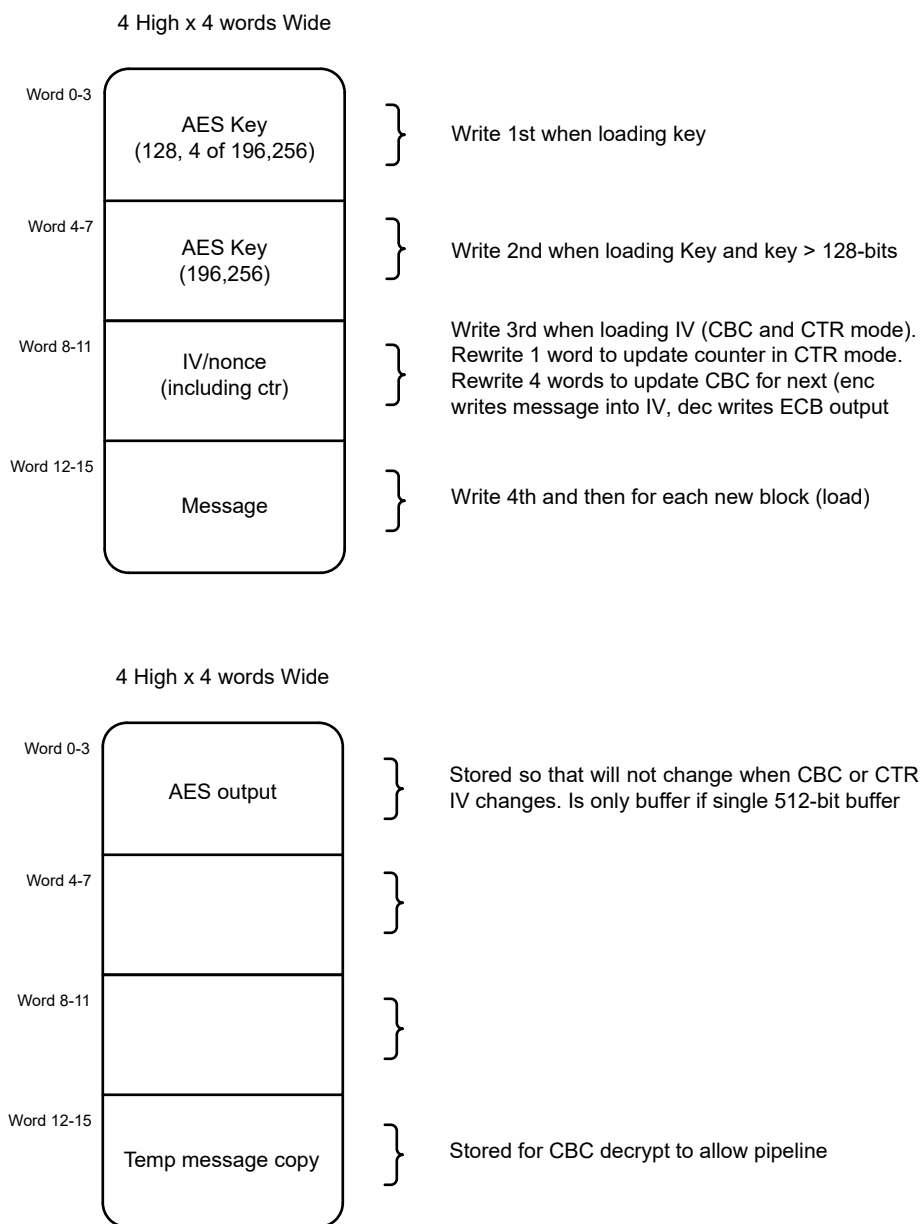


Figure 103. 2x512-bit buffers are used for AES

26.2.3 Using AES engine

1. When starting a new operation, write the CTRL[NEW] bit to initialize.
2. Read a 32-bit entropy seed from TRNG and write to PRNG_SEED register.
3. The AES engine uses a 128-bit, 192-bit, or 256-bit key depending on the value is the CRYPTCFG[AESKEYSZ] field. Hardware, PUF, or SW can supply the key.
 - If the key comes from PUF, see PUF section, "Key storage/management", which describes PUF key loading for AES.
 - If the key is selected as provided by SW provided, write the 4, 6, or 8 key word values into the INDATA register. These values will be placed in the correct place in buffer1. The field STATUS[NEEDKEY] = 1 until completed.

4. The software defined keys are not retained during Power-down or Deep power-down modes and must be reloaded on reset. PUF keys are retained during Power-down mode but not during Deep-power down mode.
5. If a Cipher mode is selected (rather than just ECB), write the IV/nonce using the four words 128 bits. These values will be placed in the correct place in buffer1. The field STATUS[NEEDIV] = 1 until completed. It may be the AHB master if enabled.
6. Read in the next 128-bit block of plain text or cipher text.
 - If the AHB master is used for read, it will read in the four words.
 - If the DMA or processor is used for read, that one will be notified to provide the four words.
7. As soon as the four words are written in, the encryption or decryption starts.
 - If a cipher mode is used, the block being processed corresponds to the rules of that mode.
8. On completion, the data is ready in the DIGESTn (or OUTDATAn) first four words. The steps allow for load next and the read out of data:
 - If the AHB master is used for read and the count is not 0, it will load in the next four words first. This allows the next block to start before the read out of the previous digest, saving time. The processor or DMA may also do this.
 - If DMA is set for out, it will trigger for reading out the four words. Else, the processor will do this through an interrupt.

26.2.4 AES performance

The AES block takes 33+2 cycles for each block to encrypt when using 128-bit keys. Using 192-bit keys adds 6 cycles and 256-bit keys adds 12 more cycles.

To decrypt, the AES block takes 43+2 cycles for each block to encrypt when using 128-bit keys. Using 192-bit keys adds 6 cycles and 256-bit keys adds 12 more cycles.

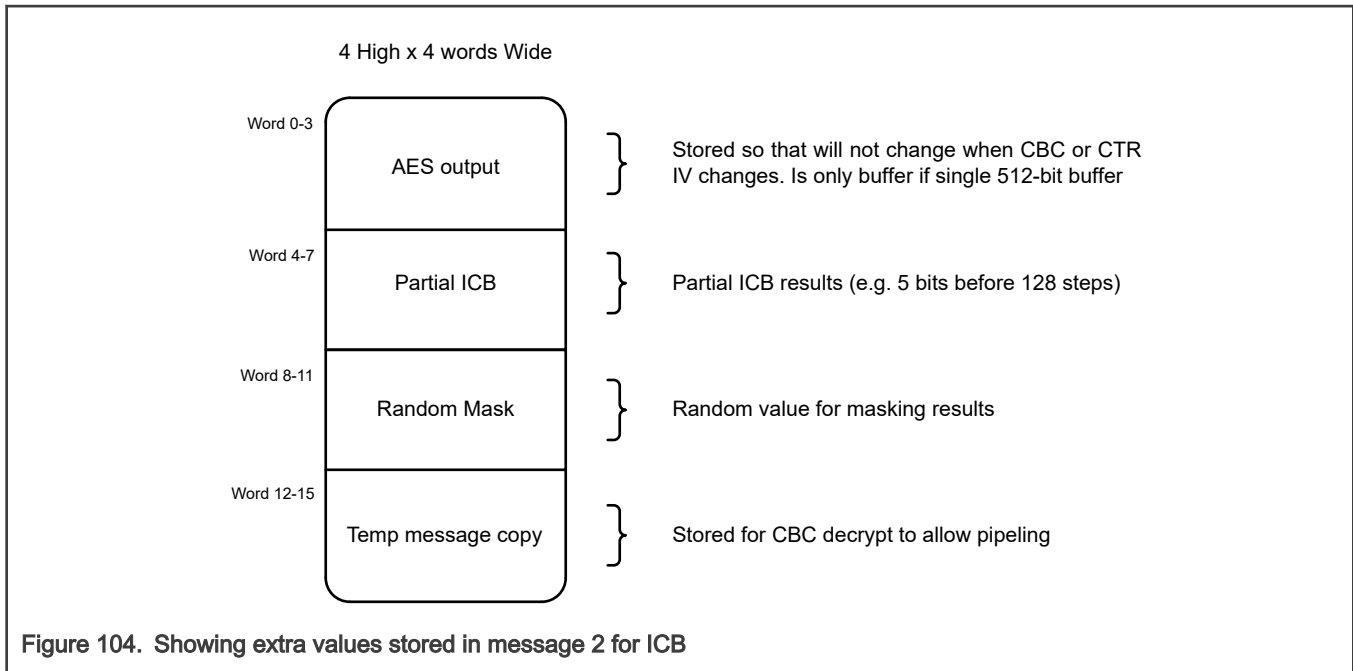
The total time required also includes the first input for example, the optional input of key and IV, as well as the input of the first four-word message data and the final output copying the final four words. All other input data and output results are pipelined and so do not add to the cycle time, unless the application or DMA is very slow.

26.2.5 Indexed Code Book AES (ICB-AES)

AES, a block cipher is an Electronic Code Book. A special hybrid cipher mode is available, called ICB (Indexed Code Book). ICB is a form of the CTR cipher mode, but its purpose is to be Side Channel Analysis (SCA) resistant. ICB uses a method which is a counter-measure to the various Side Channel attacks such as SPA/DPA/DPX (power analysis) and emanation analysis. ICB is slower than normal AES ECB and CTR mode, as a consequence of being SCA resistant. ICB mode can be used for extra-secure on-chip storage for sensitive information.

In ICB-AES mode the first block out would be 32x, 64x, 96x, or 128x slower than the normal AES block. To make processing faster for a stream of blocks, the ICB mechanism will store a partial result into word 4 - 7 of the second message buffer, which means the second and subsequent blocks are only 3, 4, 5, or 6x slower than a normal AES block, in exchange for being able to run a stream of 8, 16, 32, or 64 blocks before starting a new one.

The MASK (word 8 - 11) should be randomly generated each time, to remove any leakage from the input of the key or output store of the data



26.2.5.1 Using Indexed Code Book AES (ICB-AES) engine

To use ICB-AES, follow these rules:

1. The application should write the four word Random Mask into the MASK register. It can be written in any order.
2. The application must use little endian for keys and the like.
3. The application may use the secret key; if writing the key, it must mask it with the inverse of the random mask before writing. The block will read out using the same inverse of the random mask as used for writing.
4. The CTR must be aligned to the CRYPTCFG[ICBSZ] field. Therefore, if IV is 64 bits, then set the CRYPTCFG[AESCTRPOS] value to 4. If IV is 128 bits, set the CRYPTCFG[AESCTRPOS] value to 0. If IV is 32 bits, then set the CRYPTCFG[AESCTRPOS] value to 6.
5. The CTR can be written with any starting value, but note that the ICBSTRM model (CRYPTCFG[ICBSTRM]) is to say how many bits are in play before it has to start over. So, if not starting with 0 in those bits (the bottom 3, 4, 5, or 6 bits), then the application should stop and reload before the implicit count of 8, 16, 32, or 64.
6. The CRYPTCFG register setting must be ECB and encrypt (CRYPTCFG[AESMODE] = 00b and CRYPTCFG[AESDECRYPT] = 0b).
7. The data is written masked using the same random mask as written in the MASK registers, which means plaintext is never exposed just as keys are never exposed. The formula then is:
 - The decrypted plain text should be unmasked and it should be done carefully to not leak.
 - The ciphered text output of ICB-AES should be unmasked prior to storing.

26.2.5.2 ICB-AES performance

Using ICB-AES for SCA (Side Channel Analysis) countermeasure protection slows performance. The speed for the first block using a new IV+ctr is 32x, 64x, 96x, or 128x slower than a AES ECB block operation that does not have the SCA countermeasure protection.

26.2.6 AES Basic configuration

- AES functionality is combined with the SHA block, referred to as HASH-AES block. For clock and reset connection programmability information please refer to HASH-AES Chip-specific initialization

- For AES registers please refer to [HASHCRYPT register descriptions](#).
- The AES block shares same base address as HASH-AES block.

26.3 HASH

26.3.1 HASH Features

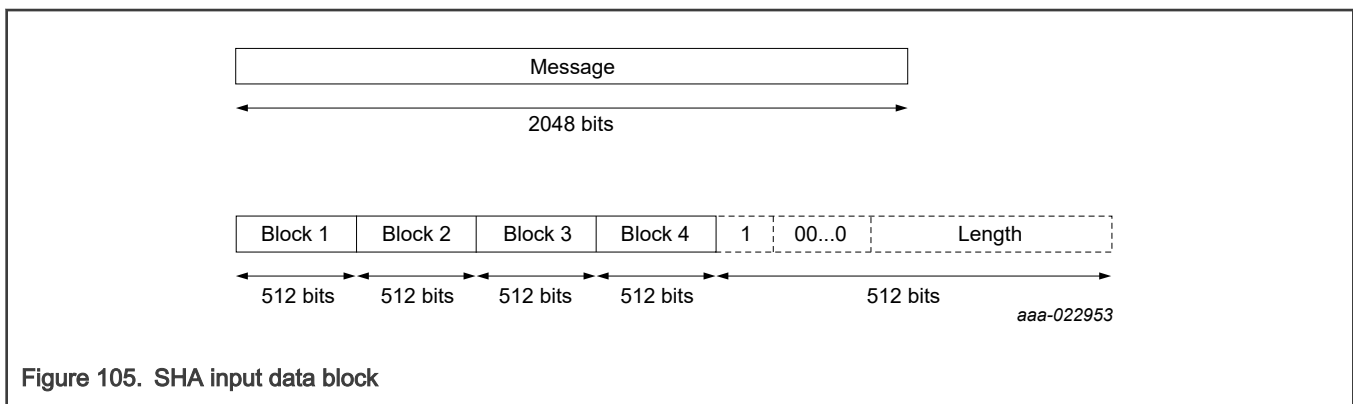
- Performs SHA-1 and SHA-2(256) based hashing
- Used with HMAC to support a challenge/response or to validate a message.

26.3.2 General description

The SHA engine processes blocks of 512 bits (16 words) at a time and performs the SHA-1 hashing in 80 clock cycles per block or SHA-256 hashing in 64 clock cycles per block. As many blocks as needed may be processed. The last block must be formatted per the SHA model:

1. The last data must be 447 bits or less. If more, then an extra block must be created.
2. After the last bit of data, a '1' bit is appended. Then, as many 0 bits are appended to take it to 448 bits long (so, 0 or more).
3. Finally, the last 64 bits contain the length of the whole message, in bits, formatted as a word.

For example, if a message is an exact multiple of 512 bits, create an extra block. The first bit of the last block will be a 1 followed by 447 zeroes. The remaining 64 bits will contain the length of the whole message including the last block.



The Arm processor uses little-endian format and therefore, the SHA engine reverses the bytes in the words written to the data register to big-endian format. It is because a hash is on bytes, so a string such as "abcd", when read as a word by the processor (or DMA) is reversed into "dcba". When the input data is provided in little-endian format, the hash block swaps them to process correctly.

DIGEST/OUTDATA Reload (RELOAD0 - RELOAD7) registers are writable to allow interleaved Hash operations, such as multiple hash context or to interleave hashing with the AES operation. During a context switch, the software can read the running hash and store it in the system memory where the necessary AES or other hashing is performed. After completion, a context restore writes the running hash to the RELOADn registers where the hashing continues from the point where it left off. The NEW_HASH bit in the [HASHCRYPT register descriptions](#) can then be set to START.

26.3.3 Security lock and register access

If a security level has locked the block using the LOCK register, no register is readable or writable from a lower level, except the LOCK register can be read any time as well as the ID at offset 0xFFC.

26.3.4 Functional description

To perform hashing, select one of the three possible ways to get the input data into the SHA engine:

- Using the core with interrupts:
 - The WAITING and ERROR interrupts are configured in the INTENSET register.
 - When STATUS[WAITING] = 1, the block is loaded by copying the 16 words using the INDATA and ALIAS registers.
 - If more blocks need to be loaded, then the WAITING interrupt bit is retained. After the last block is loaded, the DIGEST interrupt is enabled (INTENSET[DIGEST] = 1).
- Using the DMA:
 - The DMA is configured for up to 1k words (64 bits, 512 bit blocks) to be read from memory. The SHA peripheral will control the DMA to feed its data as fast as it can.
 - The SHA engine is double buffered and therefore, allows loading another 16 words while processing the previous words. This pipeline method allows continuous processing of input data.
 - An interrupt is used to notify the processor when the DMA transfer is complete. The interrupt service routine (ISR) can enable the DIGEST interrupt and ERROR interrupt to process the results. Or, it can configure the DMA for more data if needed.
 - If the last block is to be constructed separately, then either the DMA can move those 16 words or the processor can do so through interrupts.
- Using AHB Master (when available):
 - The SHA peripheral is enabled as the AHB bus master. The memory location to read from SRAM or flash and the number of blocks to read is configured using the MEMCTRL register.
 - The DIGEST and ERROR interrupts are enabled in INTENSET register. The interrupts will not occur until the last block is completed and the digest is computed.
 - If the last block is to be constructed separately, then the interrupt service routine may load the constructed last block (or use the DMA) and it will be interrupted when the DIGEST is ready.

26.3.4.1 Performance of SHA engine

The SHA engine contains two message buffers which can be loaded by the CPU, the DMA, or the AHB bus master. The performance of the block depends on the memory from where the input data is fetched (Code RAM, system RAM or flash) and activity on the system bus.

Input data loaded by CPU

The core writes 16 words to start Hashing. The block uses INDATA and ALIAS registers to support write operation to contiguous locations. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The processor can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

Input data loaded by DMA

The DMA loads the 16 words based on requests. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The DMA can request and load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

Input data loaded by AHB bus master

The AHB bus master loads 16 words from memory. The Hash operation takes 64 or 80 clock cycles based on SHA-1 or SHA-256 Hash algorithm. The AHB master can load the next 16 words during the time when the Hash operation is being performed on the previous loaded data.

26.3.4.2 Interrupt Service Routine (ISR)

ISR when using CPU:

When using CPU to load data into the SHA engine, the algorithm for ISR is

- If STATUS[ERROR] = 1, there is an issue with the application code since ERROR means overrun.

- If STATUS[WAITING] = 1, write 16 words into the SHA peripheral. The fastest method is by using structure copy as shown below. If there are no more blocks after this, clear the WAITING interrupt using INTENCLR[WAITING] and then set DIGEST using INTENSET[DIGEST].
- The fastest copy is usually

```
struct HASH_W {unsigned v[8];} *src, *dst;
src = (struct HASH_W *)memory_to_read_from;
dst = (struct HASH_W *)HASH0_INDATA; // indata and aliases
dst[0] = src[0]; // 1st 8
dst[1] = src[1]; // 2nd 8
```

- If STATUS[DIGEST] = 1, the DIGEST is ready and so process the DIGESTn registers (for example, copy) and clear the interrupt using INTENCLR[DIGEST] = 1.

ISR when using DMA

When the DMA is used for loading the data into the SHA peripheral, the ISR algorithm is:

- If all the input data are loaded into the SHA engine, enable the DIGEST interrupt in the INTENSET register to generate an interrupt when the DIGEST is ready. (INTENSET[DIGEST] = 1)
- If the last block needs to be manually loaded, write the 16 words now, or use the CPU ISR described before to do the one block.
- If STATUS[DIGEST] = 1, the DIGEST is ready and so process the DIGESTn registers (for example, copy) and clear the interrupt using INTENCLR[DIGEST] = 1.

The ERROR bit is always 0 in this case because an error is not possible.

ISR for AHB master

The ISR for AHB master is only for DIGEST or ERROR. An ERROR would be a bus error, so the algorithm is:

- If STATUS[ERROR] = 1, there is AHB master bus fault. The MEMCTRL[COUNT] field indicates which block was processing and the MEMADDR register indicates which memory location was used when the error occurred.
- If STATUS[DIGEST] = 1, the DIGEST is ready and so process the DIGESTn registers (for example, copy) and clear the interrupt using INTENCLR[DIGEST] = 1.

26.3.5 HASH-AES Initialization

Initial configuration of the chip to use the HASH-AES can be accomplished as follows:

- Enable the clock to the HASH-AES (SHA) engine.
- Clear the SHA engine peripheral reset.
- Configure the interrupts to wake-up the device from Deep-sleep mode.
- Configure input and output preferences.
- If using AHB, set the priority for the SHA engine.

Set the HASH-AES registers:

- Select SHA-1 or SHA-256 mode using the CTRL register.
- To start a new Hash write 1 to the CTRL[NEW] field. CTRL[NEW] automatically self-clears.
- To input data into the SHA engine, when using:
 - CPU: Write to the INTENSET register to enable the WAITING and ERROR interrupts.
 - DMA: Configure the DMA
 - Enable the DMA interrupt so the application knows when DMA transfer is done.
 - Set the CTRL[DMA] bit.
- AHB master

- Enable the INTENSET[DIGEST] and INTENSET[ERROR] interrupts.
- Write to the MEMADDR register with the offset in SRAM or flash.
- Write to the MEMCTRL[MASTER] field to enable the SHA engine as AHB bus master and write the number of 512-bit blocks to process in the MEMCTRL[COUNT] field.

26.4 Memory map and register definition

This section includes the HASH-AES or HASHCRYPT module memory map and detailed descriptions of all registers.

26.4.1 HASHCRYPT register descriptions

26.4.1.1 HASH memory map

HASHCRYPT0 base address: 4020_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CTRL)	32	RW	0000_0000h
4h	Status (STATUS)	32	RW	0000_0000h
8h	Interrupt Enable (INTENSET)	32	RW	0000_0000h
Ch	Interrupt Clear (INTENCLR)	32	RW	0000_0000h
10h	Memory Control (MEMCTRL)	32	RW	0000_0000h
14h	Memory Address (MEMADDR)	32	RW	0000_0000h
20h	Input Data (INDATA)	32	W	0000_0000h
24h - 3Ch	Alias (ALIAS0 - ALIAS6)	32	W	0000_0000h
40h - 5Ch	Digest0 n/Output Data0 n (DIGEST00 - DIGEST07)	32	R	0000_0000h
80h	Cryptographic Configuration (CRYPTCFG)	32	RW	0000_0000h
84h	Configuration (CONFIG)	32	R	0000_09CBh
8Ch	Lock (LOCK)	32	RW	0000_0000h
90h - 9Ch	Mask (MASK0 - MASK3)	32	W	See section
A0h - BCh	DIGEST/OUTDATA Reload (RELOAD0 - RELOAD7)	32	RW	0000_0000h
D0h	PRNG Seed (PRNG_SEED)	32	W	0000_0000h
D8h	PRNG Output (PRNG_OUT)	32	W	0000_0000h

26.4.1.2 Control (CTRL)

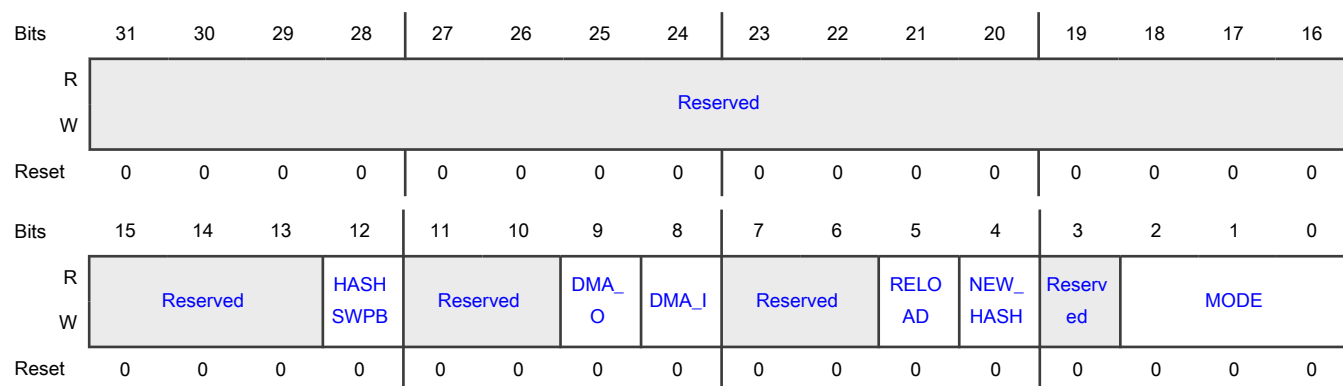
Offset

Register	Offset
CTRL	0h

Function

The Control register configures the Hash-AES engine. The Hash-AES engine is enabled when the MODE bit is selected to SHA-1 or SHA-256. The NEW bit field is written to 1, before the data can be loaded into INDATA (or its aliases, or both) register.

Diagram



Fields

Field	Function
31-13 —	Reserved
12 HASHSWPB	Hash Swap Bytes If 1, will swap bytes in the word for SHA hashing. The default is byte order (so LSB is 1st byte) but this allows swapping to MSB is 1st such as is shown in SHS spec. For cryptographic swapping, see the CRYPTCFG register.
11-10 —	Reserved
9 DMA_O	DMA to Drain the Digest/Output If both DMA_I and DMA_O are set, the DMA has to know to switch direction and the locations. This can be used for crypto uses. 0b - DMA is not used. Processor reads the digest/output in response to DIGEST interrupt. 1b - DMA will drain the data.
8 DMA_I	DMA to Fill INDATA. If Hash, will request from DMA for 16 words and then will process the Hash. If Cryptographic, it will load as many words as needed, including key if not already loaded. It will then request again. Normal model is that the DMA interrupts the processor when its length expires. Note that if the processor will write the key and optionally IV, it should not enable this until it has done so. Otherwise, the DMA will be expected to load those for the 1st block (when needed). 0b - DMA is not used. Processor writes the necessary words when WAITING is set (interrupts), unless AHB Master is used.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - DMA will push in the data.
7-6 —	Reserved
5 RELOAD	Reload RELOAD is used to save a partial Hash Digest (e.g. when need to run AES) and then reload it later for continuation. 0b - Disabled 1b - Allows the SHA RELOAD registers to be used.
4 NEW_HASH	New Hash Operation Written with 1 when starting a new Hash/Crypto. It self clears. NOTE The WAITING Status bit will clear for a cycle during the initialization from New=1. 1b - Starts a new Hash/Crypto and initializes the Digest/Result.
3 —	Reserved
2-0 MODE	Operational Mode The CONFIG register will indicate if specific modes beyond SHA1 and SHA2-256 are available. 000b - Disabled 001b - SHA1 is enabled 010b - SHA2-256 is enabled 011b - Reserved 100b - AES is enabled (see also CRYPTCFG register for more controls) 101b - ICB-AES is enabled (see also CRYPTCFG register for more controls) 110b - Reserved 111b - Reserved

26.4.1.3 Status (STATUS)

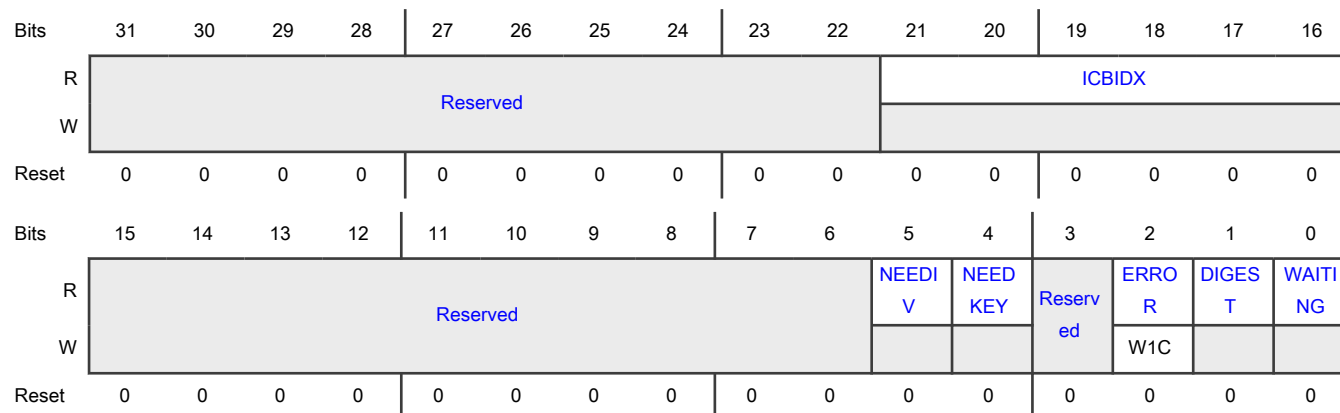
Offset

Register	Offset
STATUS	4h

Function

It indicates the status of the Hash-AES peripheral. It shows when the SHA engine is waiting for data and when the results are available. These bits correspond to both interrupts and DMA (in the case of data).

Diagram



Fields

Field	Function
31-22 —	Reserved
21-16 ICBIDX	ICB Index Count If ICB-AES is selected, then reads as the ICB index count based on ICBSTRM (from CRYPTCFG). That is, if 3 bits of ICBSTRM, then this will count from 0 to 7 and then back to 0. On 0, it has to compute the full ICB, quicker when not 0.
15-6 —	Reserved
5 NEEDIV	Need IV/Nonce Indicates the block wants an IV/Nonce to be written in (set along with WAITING) 0b - No IV/Nonce is needed, either because written already or because not needed. 1b - IV/Nonce is needed and INDATA/ALIAS will be accepted as IV/Nonce. Will also set WAITING.
4 NEEDKEY	Need Key to be Written Indicates the block wants the key to be written in (set along with WAITING) 0b - No Key is needed and writes will not be treated as Key 1b - Key is needed and INDATA/ALIAS will be accepted as Key. Will also set WAITING.
3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 ERROR	Error If 1, an error occurred. For normal uses, this is due to an attempted overrun: INDATA was written when it was not appropriate. For Master cases, this is an AHB bus error; the COUNT field will indicate which block it was on. 0b - No error. 1b - An error occurred since last cleared (written 1 to clear).
1 DIGEST	Digest/Outdata For Hash, if 1 then a DIGEST is ready and waiting and there is no active next block already started. For Cryptographic uses, this will be set for each block processed, indicating OUTDATA (and OUTDATA2 if larger output) contains the next value to read out. This is cleared when any data is written, when New is written, for Cryptographic uses when the last word is read out, or when the block is disabled. 0b - Digest is not ready 1b - Digest is ready. Application may read it or may write more data.
0 WAITING	Waiting for Data Block is waiting for more data to process. 0b - Not waiting for data – may be disabled or may be busy. For cryptographic uses, this is not set if IsLast is set nor will it set until at least 1 word is read of the output. 1b - Waiting for data to be written (16 words)

26.4.1.4 Interrupt Enable (INTENSET)

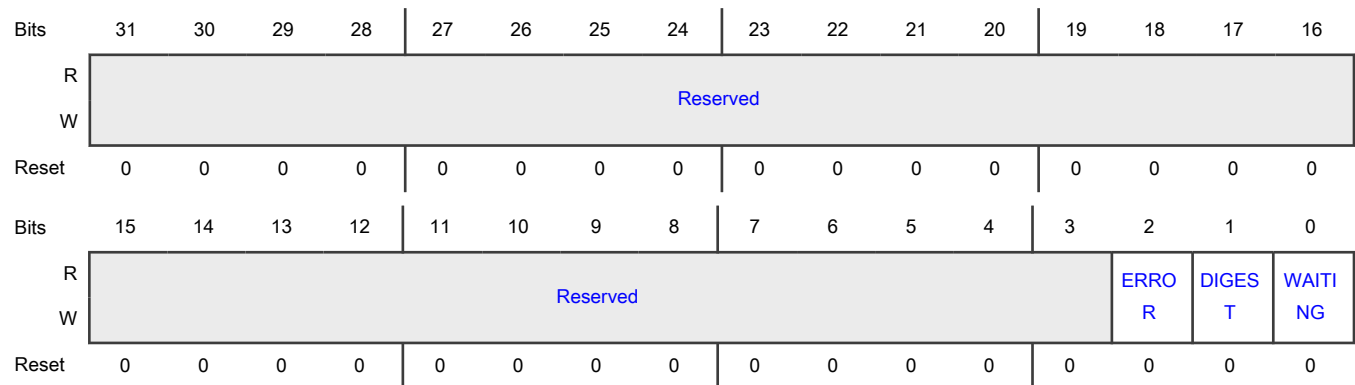
Offset

Register	Offset
INTENSET	8h

Function

The Interrupt enable register is used to enable interrupt sources that cause processor interrupts.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 ERROR	Interrupt on Error It indicates if interrupt is generated on an ERROR (as defined in STAT register). 0b - Interrupt not enabled on Error. 1b - Interrupt is enabled on Error (until cleared).
1 DIGEST	Digest/Outdata DIGEST indicates if interrupt is generated when Digest is ready (completed a Hash or completed a full sequence). The interrupt is cleared when any data is written to INDATA or ALIAS registers, when NEW bit is written, or when the SHA engine is disabled. 0b - Interrupt not enabled when Digest is ready 1b - Interrupt is enabled when Digest is ready. Interrupt cleared by writing more data, starting a new Hash, or disabling (done).
0 WAITING	Interrupt When Waiting for Data Input WAITING indicates if interrupt should be enabled when waiting for input data. The interrupt is cleared when any data is written to INDATA or ALIAS registers. 0b - Interrupt not enabled when waiting 1b - Interrupt is enabled when waiting

26.4.1.5 Interrupt Clear (INTENCLR)

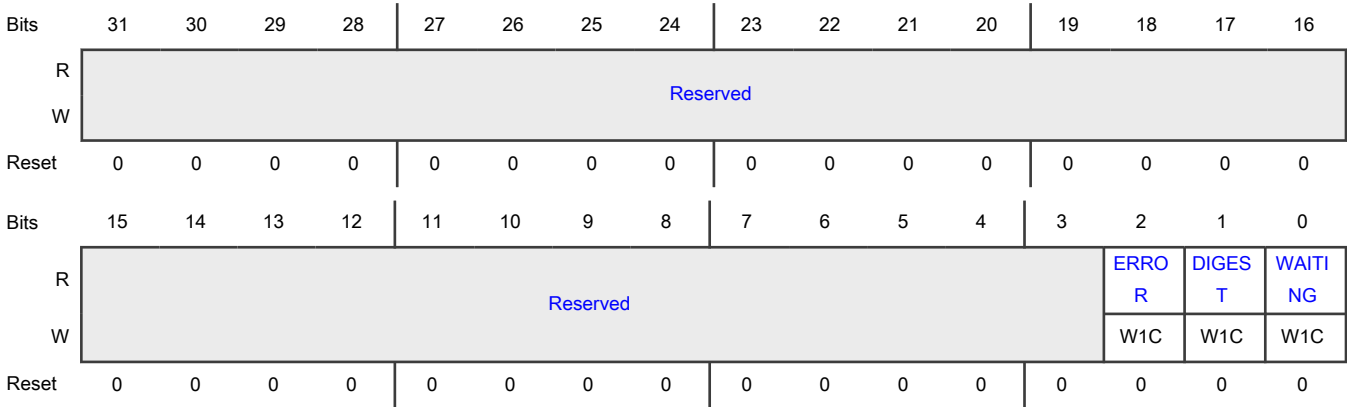
Offset

Register	Offset
INTENCLR	Ch

Function

The Interrupt clear register is used to clear the interrupt mask enabled by the INTENSET register.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 ERROR	Error Writing a 1 clears the interrupt enabled by the INTENSET register.
1 DIGEST	Digest Writing a 1 clears the interrupt enabled by the INTENSET register.
0 WAITING	Waiting Writing a 1 clears the interrupt enabled by the INTENSET register.

26.4.1.6 Memory Control (MEMCTRL)

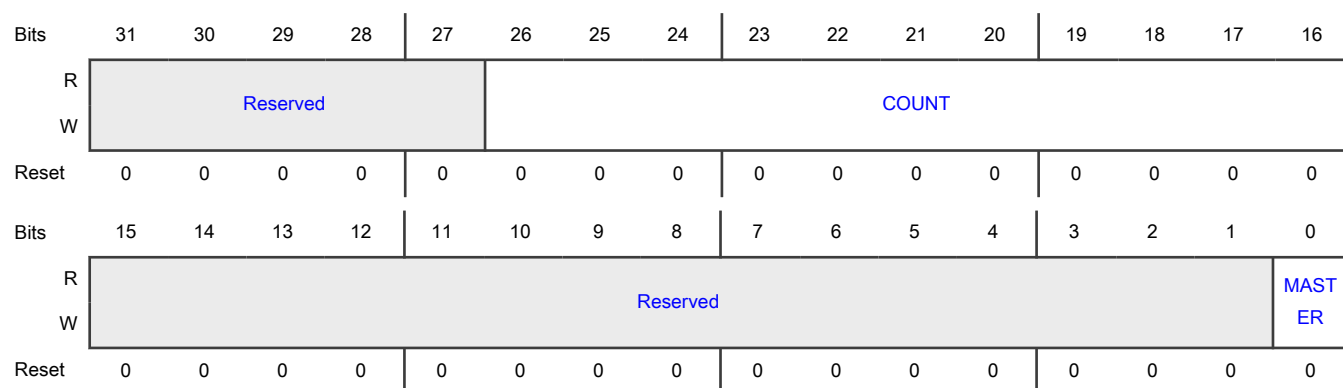
Offset

Register	Offset
MEMCTRL	10h

Function

The Memory Control register (MEMCTRL) allows setting up the SHA engine to be the AHB bus master to read memory for hashing. It can be used to read 512-bit blocks from memory. The starting location must be word aligned and the length may be up to 128 KB.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-16 COUNT	Count Number of 512-bit (128-bit if AES, except 1st block which may include key and IV) blocks to copy starting at MEMADDR. This register will decrement after each block is copied, ending in 0. For Hash, the DIGEST interrupt will occur when it reaches 0. Fro AES, the DIGEST/OUTDATA interrupt will occur on ever block. If a bus error occurs, it will stop with this field set to the block that failed. 000_0000_0000b - Done. Nothing to process 000_0000_0001b - One 512-bit block to hash 000_0000_0010b - Two 512-bit block to hash 000_0000_0011b - Three 512-bit block to hash. The maximum number of 512-bit blocks that can be processed is 2047 blocks.
15-1 —	Reserved
0 MASTER	Master This field is used to enable SHA engine as AHB bus master. 0b - Mastering is not used and the normal DMA or Interrupt based model is used with INDATA. SHA engine is not AHB bus master and the DMA or Interrupt based model is used with INDATA. 1b - Mastering is enabled and DMA and INDATA should not be used. Enables SHA engine as AHB bus master. DMA and INDATA should not be used.

26.4.1.7 Memory Address (MEMADDR)

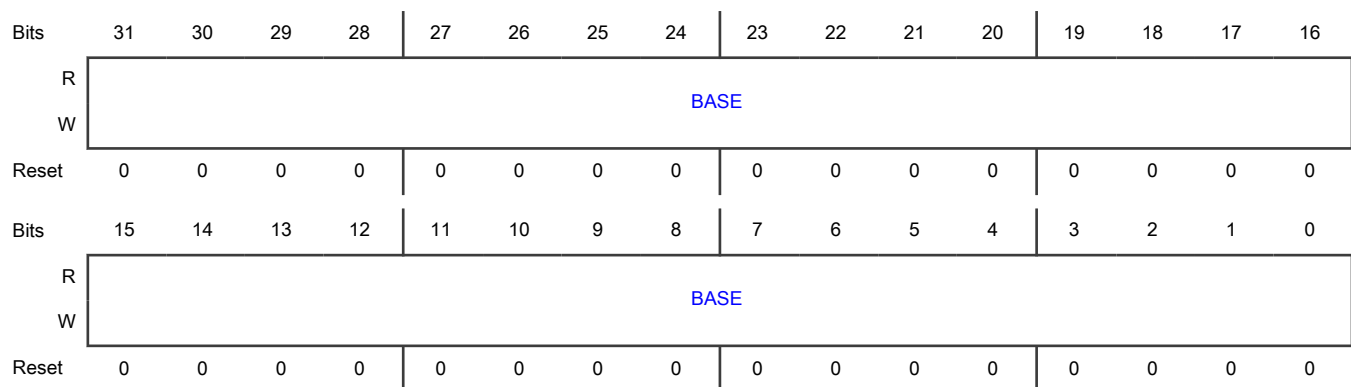
Offset

Register	Offset
MEMADDR	14h

Function

The Memory address register (MEMADDR) holds the base address for MEMCTRL. It must only point to valid locations in memory based on the RT600 device used and must be word aligned.

Diagram



Fields

Field	Function
31-0	Base
BASE	Address base to start copying from, word aligned (so bits 1:0 must be 0). This field will advance as it processes the words. If it fails with a bus error, the register will contain the failing word. N:Address in Flash or RAM space; RAM only as mapped in this part. May also be able to address SPIFI.

26.4.1.8 Input Data (INDATA)

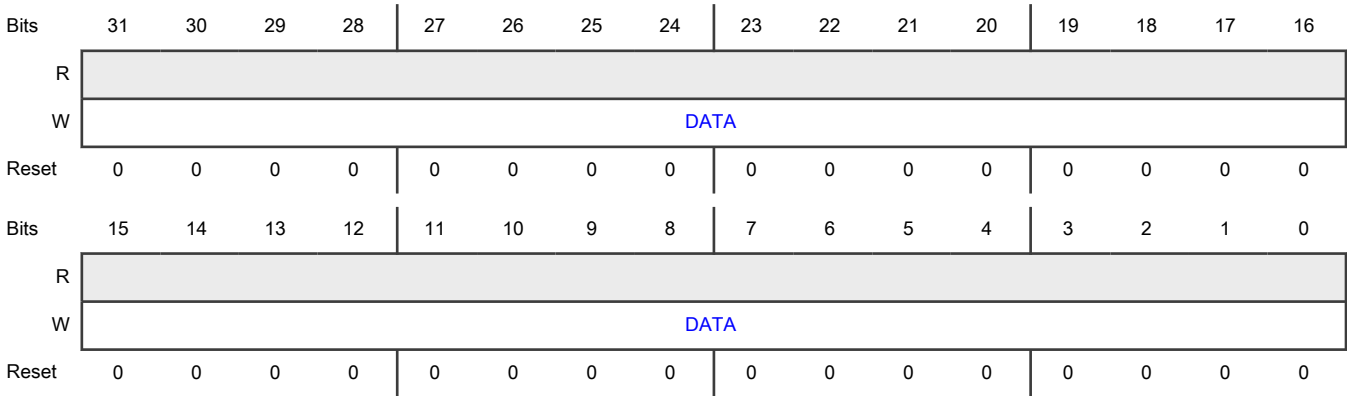
Offset

Register	Offset
INDATA	20h

Function

The INDATA registers are used for writing the 16 words per hash. The aliases exist so the processor can use Store Multiple (STM). The DMA only writes to INDATA.

Diagram



Fields

Field	Function
31-0 DATA	Data Write next word in little-endian form. The hash requires big endian word data, but this block swaps the bytes automatically. SHA engine swaps the word to the required big-endian format.

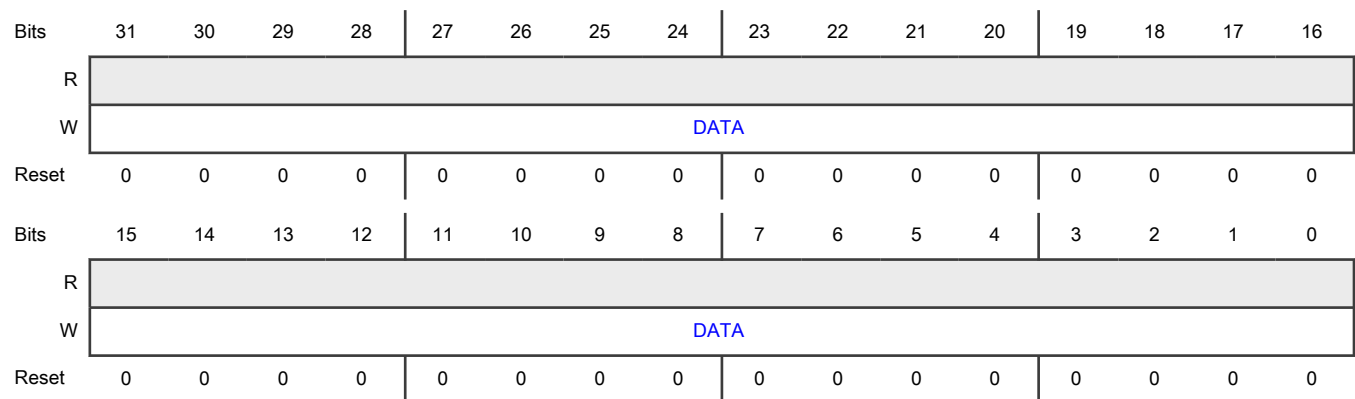
26.4.1.9 Alias (ALIAS0 - ALIAS6)

Offset

Register	Offset
ALIAS0	24h
ALIAS1	28h
ALIAS2	2Ch
ALIAS3	30h
ALIAS4	34h
ALIAS5	38h
ALIAS6	3Ch

Function

Aliases to allow writing words in a burst. The INDATA and its ALIAS registers are used for writing the 16 words per hash. The aliases exist so the processor can use Store Multiple (STM). The DMA only writes to INDATA.

Diagram**Fields**

Field	Function
31-0	Data
DATA	Write next word in little-endian form. SHA engine swaps the word to the required big-endian format.

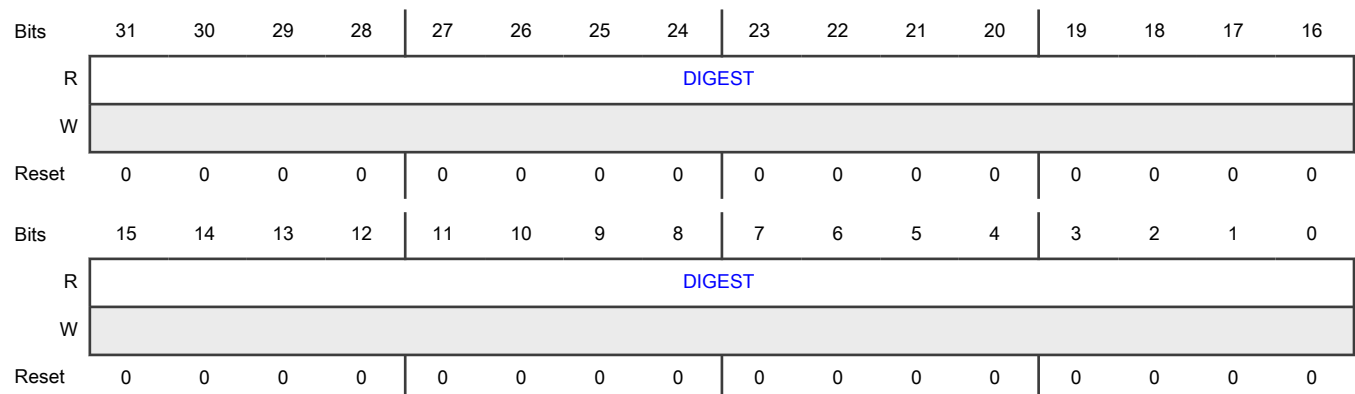
26.4.1.10 Digest0 n/Output Data0 n (DIGEST00 - DIGEST07)**Offset**

For n = 0 to 7:

Register	Offset
DIGEST0n	40h + (n × 4h)

Function

DIGEST0n contains 128 bits, 160 bits or 256 bits, depending on AES, SHA1, SHA256 or crypto or SHA512. The registers are written in word format, therefore, endianness should be considered when sending or comparing. The first five DIGEST registers are populated for SHA-1 and all eight DIGEST registers are populated for SHA-256. If SHA-1 is used, DIGEST [0:4] are populated and if SHA-256 is used, DIGEST [0:7] are populated.

Diagram

Fields

Field	Function
31-0 DIGEST	Digest One word of the Digest or output.

26.4.1.11 Cryptographic Configuration (CRYPTCFG)

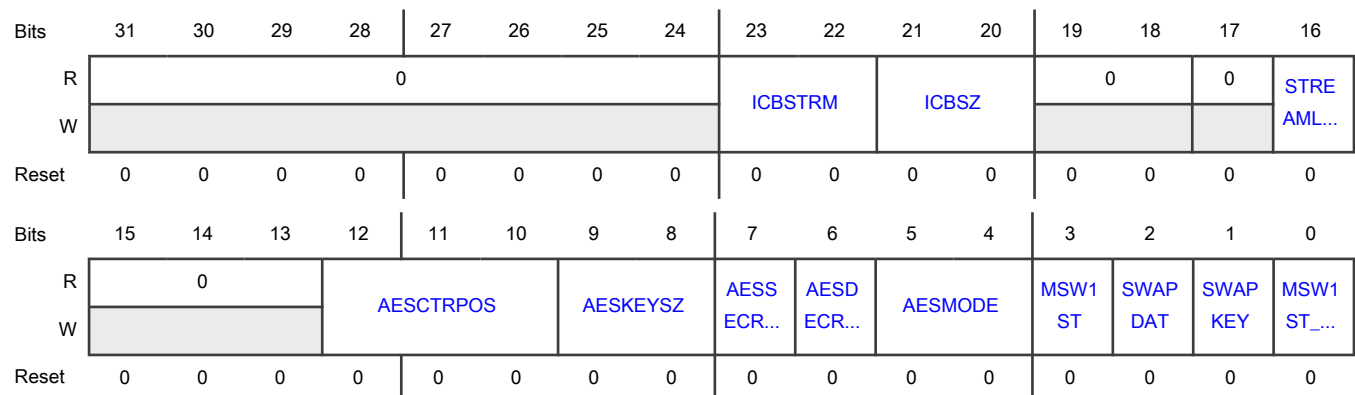
Offset

Register	Offset
CRYPTCFG	80h

Function

Crypto settings for AES. It is ignored if SHA hashing is selected. Only the fields for the selected encryption scheme will be used and any field relating to a feature not supported will be ignored (writes will have no effect and the bits will read back as 0).

Diagram



Fields

Field	Function
31-24 —	Reserved
23-22 ICBSTRM	ICB Stream Size The size of the ICB-AES stream that can be pushed before needing to compute a new IV/ctr (counter start). ICBSTRM optimizes the performance of the stream of blocks after the 1st. 00b - 8 blocks

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - 16 blocks 10b - 32 blocks 11b - 64 blocks
21-20 ICBSZ	ICB Size ICBSZ sets the ICB size between 32 and 128 bits, using the following rules. The counter is assumed to occupy the low order bits of the IV. 00b - 32 bits of the IV/ctr are used (from 127:96) 01b - 64 bits of the IV/ctr are used (from 127:64) 10b - 96 bits of the IV/ctr are used (from 127:32) 11b - All 128 bits of the IV/ctr are used
19-18 —	Reserved
17 —	Reserved
16 STREAMLAST	Stream Last STREAMLAST is 1 if last stream block. If not 1, then the engine will compute the next "hash".
15-13 —	Reserved
12-10 AESCTRPOS	AES CTR Position Only supports 16b counter, so application must control any additional bytes if using more. The 16-bit counter is read from the IV and incremented by 1 each time. Any other use CTR should use ECB directly and do its own XOR and so on.
9-8 AESKEYSZ	AES Key Size Sets the AES key size 00b - 128 bit key 01b - 192 bit key 10b - 256 bit key 11b - Reserved
7 AESSECRET	AES Secret Selects the Hidden Secret key vs. User key, if provided. If security levels are used, only the highest level is permitted to select this.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - User key provided in normal way 1b - Secret key provided in hidden way by HW
6 AESDECRYPT	AES Decrypt AES ECB direction. Only encryption is used with CTR mode or manual modes such as CFB. 0b - Encrypt 1b - Decrypt
5-4 AESMODE	AES Cipher Mode AES Cipher Mode to use if using plain AES. 00b - ECB – used as is 01b - CBC mode (see details on IV/nonce) 10b - CTR mode (see details on IV/nonce). See also AESCTRPOS. 11b - Reserved
3 MSW1ST	Most Significant Word 1st Load If 1, load of key, IV, and data is MSW 1st for AES. Else, the words are little endian, if allowed by configuration.
2 SWAPDAT	Swap Data/IV Inputs If 1, will SWAP the data and IV inputs (bytes in each word).
1 SWAPKEY	Swap Key If 1, will Swap the key input (bytes in each word).
0 MSW1ST_OUT	Most Significant Word 1st Out It reads the most significant word 1st for AES when set to 1. Else it will be read in normal little endian – Least significant word 1st, if allowed by configuration.

26.4.1.12 Configuration (CONFIG)

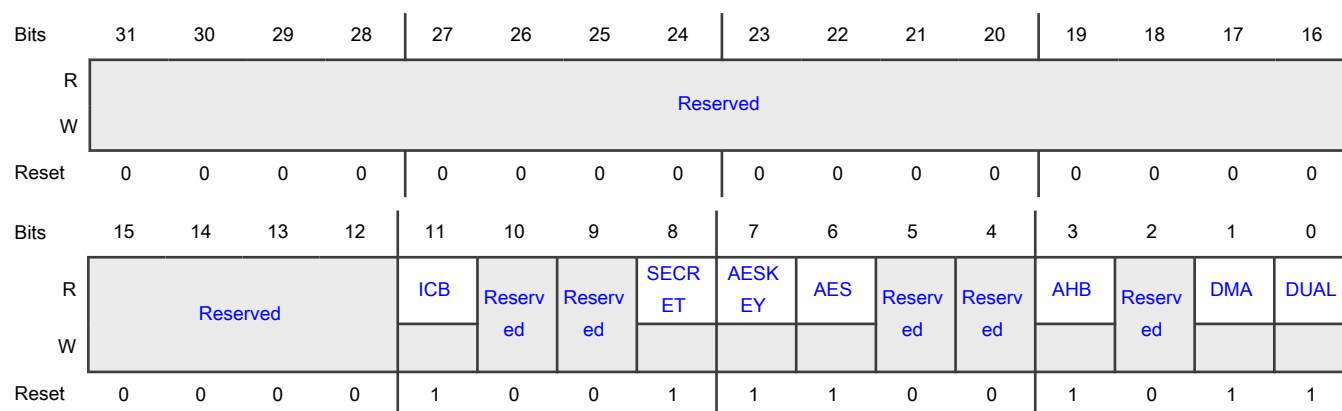
Offset

Register	Offset
CONFIG	84h

Function

Returns the configuration of the module. CONFIG indicates what services are available.

Diagram



Fields

Field	Function
31-12 —	Reserved
11 ICB	ICB Reads 1 if ICB over AES is included
10 —	Reserved
9 —	Reserved
8 SECRET	Reads 1 if AES Secret key is available
7 AESKEY	Reads 1 if AES 192 and 256 also included
6 AES	Reads 1 if AES 128 is included
5 —	Reserved
4 —	Reserved
3 AHB	Reads 1 if AHB Master is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	Reserved
1 DMA	Reads 1 if DMA is connected
0 DUAL	Reads 1 if 2 x 512 bit buffers, 0 if only 1 x 512 bit

26.4.1.13 Lock (LOCK)

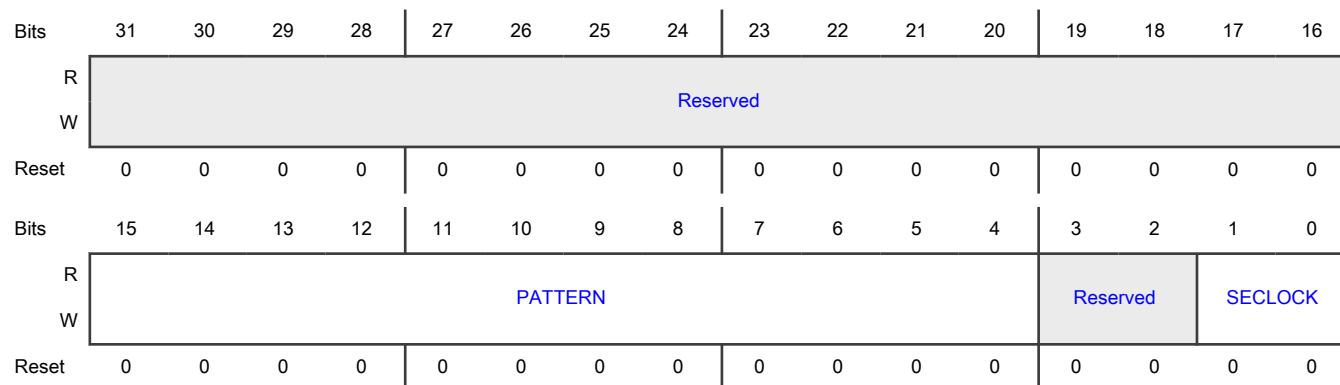
Offset

Register	Offset
LOCK	8Ch

Function

The Lock register is used to secure-lock the block from use by lower security levels. When the lock is written, it records the current security level. Only that level and higher may use the block (read or write) until it is unlocked. It may only be unlocked by a security level which is the lock-level or higher. The lock state is readable by any security level along with the ID, but all other registers are masked off to lower levels when locked. Changing the SECLOCK field erases the current crypto key (if any). It also resets the secret key selector (AESSECRET), if set.

Diagram



Fields

Field	Function
31-16	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
15-4 PATTERN	Must write 0xA75 to change lock state. A75:Pattern needed to change bits 1:0
3-2 —	Reserved
1-0 SECLOCK	Secure Lock Write 1 to secure-lock this block (if running in a security state). Write 0 to unlock. If locked already, may only write if at same or higher security level as lock. Reads as: 0 if unlocked, else 1, 2, 3 to indicate security level it is locked at. NOTE Lock and ID are the only readable registers if locked and current state is lower than lock level. 00b - Unlocks, so block is open to all. But, AHB Master will only issue non-secure requests. 01b - Locks to the current security level. AHB Master will issue requests at this level.

26.4.1.14 Mask (MASK0 - MASK3)

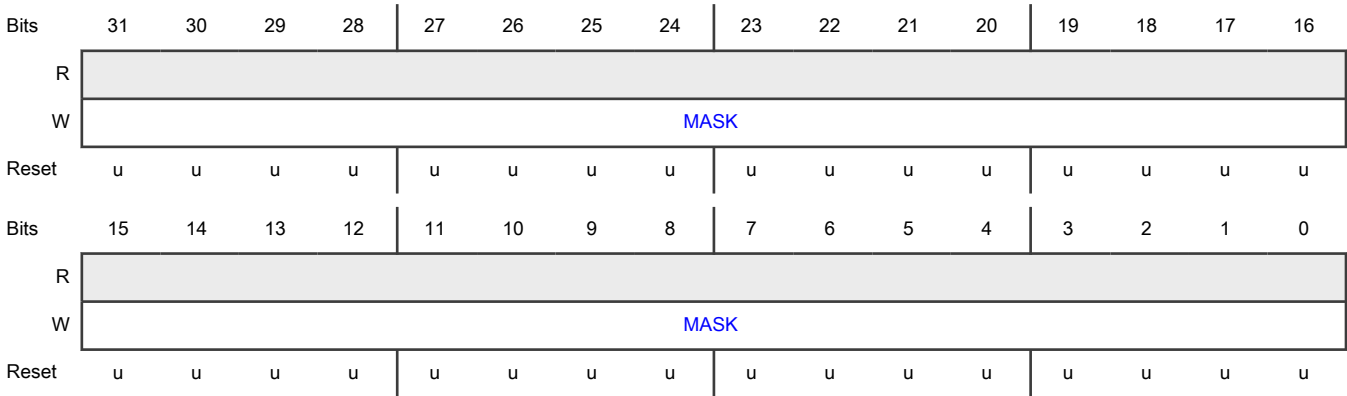
Offset

Register	Offset
MASK0	90h
MASK1	94h
MASK2	98h
MASK3	9Ch

Function

The write-only mask registers are written with a random mask to form 128 bits of randomness for masking of the ICB output results. This means that the plaintext is not stored ever, but is always masked.

Diagram



Fields

Field	Function
31-0	A random word.
MASK	

26.4.1.15 DIGEST/OUTDATA Reload (RELOAD0 - RELOAD7)

Offset

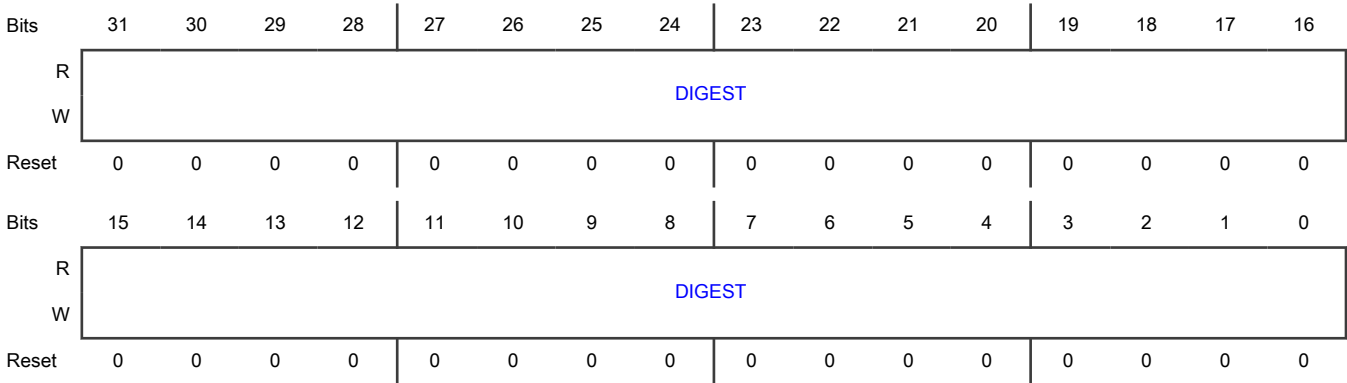
For a = 0 to 7:

Register	Offset
RELOADa	A0h + (a × 4h)

Function

Provides the SHA Digest word to reload.

Diagram



Fields

Field	Function
31-0 DIGEST	SHA Digest word to reload.

26.4.1.16 PRNG Seed (PRNG_SEED)

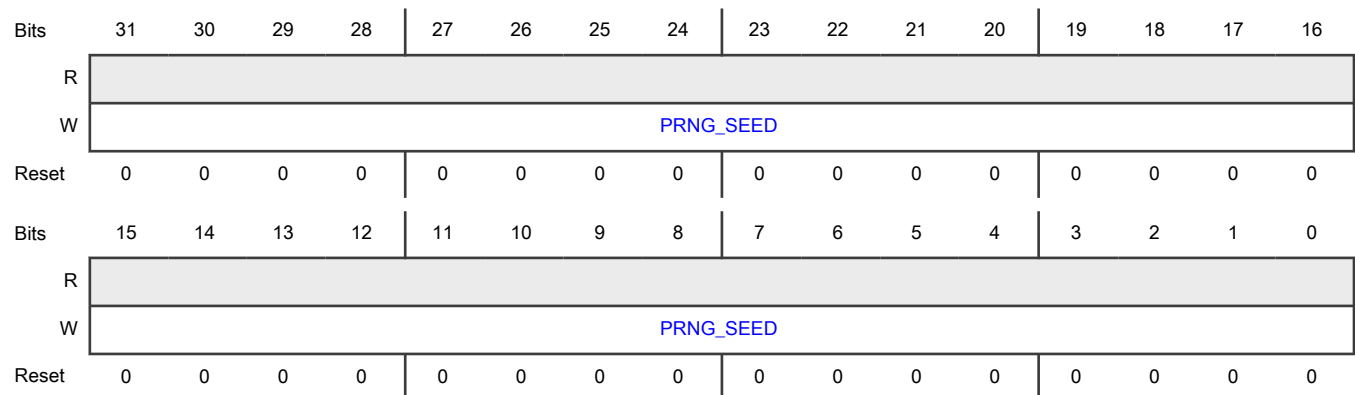
Offset

Register	Offset
PRNG_SEED	D0h

Function

Provides the seed and access to a randomly generated number.

Diagram



Fields

Field	Function
31-0 PRNG_SEED	SHA Digest word to reload. This input offers the random seed for a PRNG block that is used for masking in AES operations. This register is written at the start of an AES operation and should be updated periodically. Updating it at the start of every new AES operation is recommended to provide enough entropy. TRNG can be used as source for this value.

26.4.1.17 PRNG Output (PRNG_OUT)

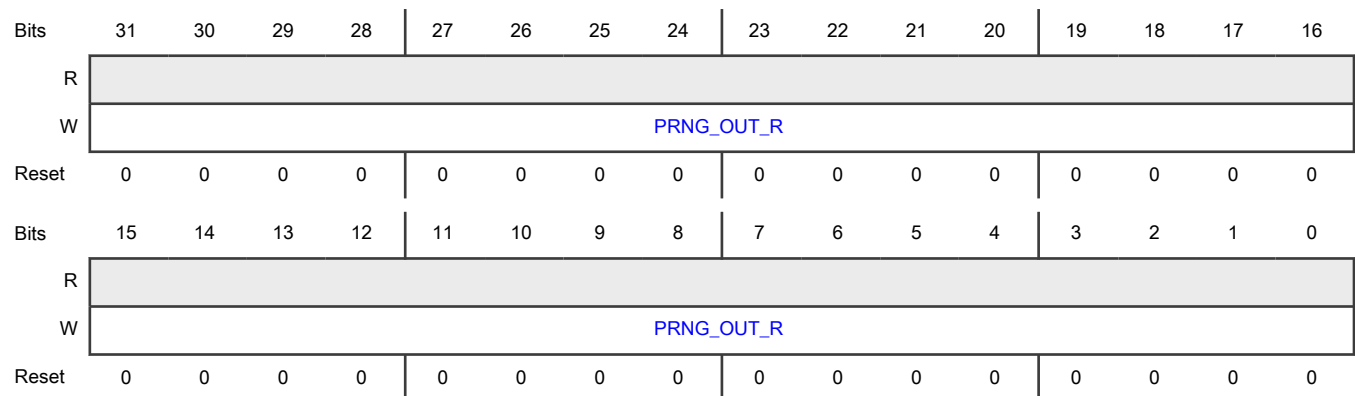
Offset

Register	Offset
PRNG_OUT	D8h

Function

Provides the output for the random number seed.

Diagram



Fields

Field	Function
31-0	SHA Digest word to reload.
PRNG_OUT_R	This register provides a 32bit pseudo random number. This register is not accessible when the AES engine is in use.

Chapter 27

Standalone TRNG

27.1 Overview

The Standalone True Random Number Generator (SA-TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy-consuming module or by other post-processing functions. A typical entropy consumer is a pseudo random number generator (PRNG) that can be implemented to achieve both true randomness and cryptographic-strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module.

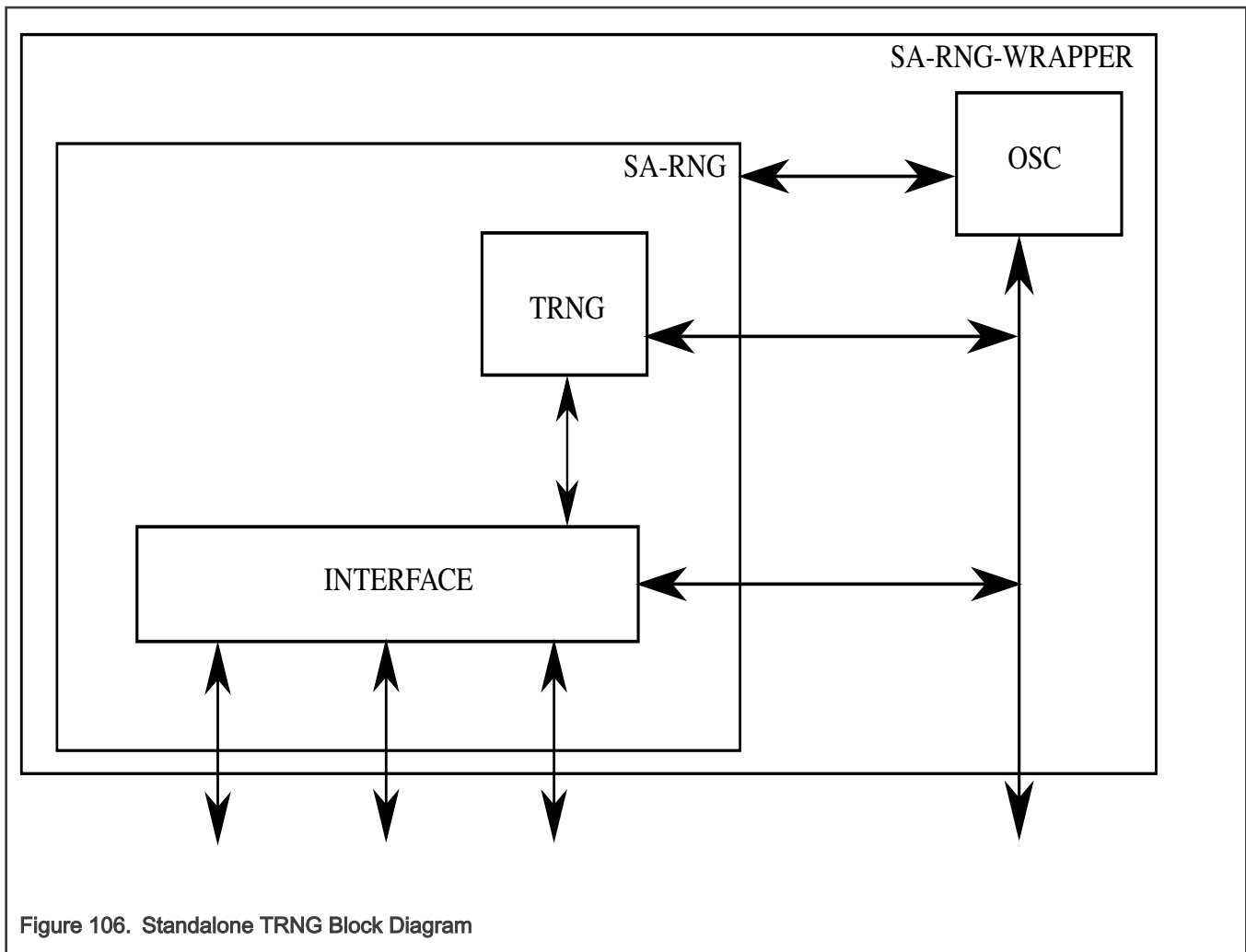
The entropy generated by a TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number be difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is particularly difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness and quality of the generated random data, PRNGs are often used to post-process the output of a TRNG.

This document describes only the TRNG design functionality and usage.

Note that before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

The TRNG contains the following submodules: IP Slave bus (SkyBlue bus) interface, the TRNG core, and the free-running oscillator (OSC).

27.1.1 Block Diagram



27.2 Functional Description

The TRNG consists of several functional submodules. Its overall functionality can be described from the top-level in terms of generating entropy for seed generation. The functionality of each sub-module is briefly described in the following subsections.

TRNG generates random bits by collecting bits from a random noise source. This random noise source is a ring oscillator that is sensitive to random noise (temperature variations, voltage variations, cross-talk and other random noise) within the device where the TRNG is used. This noise causes various small changes in the period of the oscillator. Therefore, if the count of the ring oscillator clock cycles is sampled after a known period of time, this count will vary each time the sample is taken. By using the variance in this count over a large number of samples, random bits can be derived. In addition to generating entropy, the TRNG also performs several statistical tests on its output.

27.3 Register Interface Usage

The TRNG register interface is used to read and write registers within TRNG for the following purposes:

- to configure internal registers and TRNG register interface during chip initialization
- to initialize TRNG self tests
- to read status registers for hardware and software debug

Note that accesses to registers must be 32-bit reads or writes.

27.4 Operation

The TRNG is designed to operate as a slave module and can be controlled via the slave bus. In order to write to most TRNG registers, the MCTL register must be initialized in programming mode. After a power-on reset (POR) or an asynchronous system hard reset, the TRNG resets to programming mode. TRNG will not generate entropy until it has transitioned out of programming mode (into run mode) and access to the entropy registers have been enabled.

There are four steps that the user (programmer/integrator) will want to do with a TRNG.

1. Initialize.

Set up the parameters to proper values and then start generation of the first block of entropy. This is done once every time after a reset or any time the user wants to update operating parameters and/or built-in self test parameter bounds.

2. Read entropy and generate next block of entropy.

This is the normal flow of operation and is looped until enough entropy has been generated.

3. Run a self test to assure proper continued operation.

This involves taking TRNG off line, setting some self test parameters, running TRNG, and then reading the statistical test registers to see if they are within proper operation values. This may not be needed as TRNG has built-in self tests.

4. Determine and check parameter values off line.

This is done in development in order to determine the proper initialization and self test parameters. The TRNG is taken off line. Test parameter values are written and entropy generation is started. If the statistical tests indicate poor operation (i.e., failing statistical tests), the entropy delay value should be increased and entropy generation should be restarted. Every case is a variation of setting TRNG parameter values, starting or restarting entropy generation, and reading out the entropy. This process requires pausing and restarting the TRNG.

27.4.1 Operation Programming Flow

Below is an example of a normal operation programming flow.

1. After reset, the TRNG will be in programming mode with the ok-to-stop bit asserted ($MCTL[TSTOP\_OK] = 1$). The TRNG must be put into run mode for entropy generation to begin ($MCTL[PRGM] = 0$). Additionally, in order to have access to the entropy registers and other critical TRNG registers, the TRNG access bit must be set ($MCTL[TRNG\_ACC] = 1$). With the default self test limits after bootup, the entropy valid bit can be polled until it asserts ($MCTL[ENT\_VAL] = 1$). Alternatively if the interrupts are enabled in the INT_MASK register, the ipi_rng_int_b output will assert when the entropy valid bit asserts ($MCTL[ENT\_VAL] = 1$).
2. After the polling completes, the 512-bit entropy generated by the TRNG can be read. The values can be read in any order from entropy register 0 to register 15, (ENT0 to ENT15). After reading ENT15, the old entropy value is reset and a new entropy value is generated. **Note that reading ENT15 will reset the entropy, so it should always be read last.**
3. The new entropy value can be polled again or the Interrupt Status Register can be used to handle reading the entropy values when the entropy valid interrupt is triggers.
4. The interrupt can be masked or cleared as needed. See the Interrupt Status Register description.
5. To change the self-test limits, the seed counters, how fast the entropy is generated, and how entropy is sampled, see the register description section. In particular, see the the TRNG Frequency Count Minimum Limit Register (FRQMIN), the seed control register(SCML), the statistical run length registers, and other parameter registers.
6. Once in Run Mode, the entropy is re-generated automatically after ENT15 is read. To stop the TRNG or access to TRNG registers at any point while in running mode, you can always set $MCTL[TRNG\_ACC]=0$. Setting the TRNG back to programming mode ($MCTL[PRGM]=1$) also achieves the purpose of stopping entropy generation.

27.4.2 Interrupt Programming Flow

The TRNG has three interrupts that are managed using three TRNG registers: INT_CTRL, INT_STATUS and INT_MASK registers. The interrupts are FRQ_CT_FAIL, ENT_VAL and HW_ERR. By default, these interrupts are disabled. They have to be individually enabled, as preferred, to work.

For example, to enable the ENT_VAL interrupt, the corresponding bit in the INT_MASK register must be set (INT_MASK[ENT_VAL] = 1). When a new interrupt is triggered, the corresponding bit in the INT_STATUS register will assert (INT_STATUS[ENT_VAL]). To clear an asserted interrupt, the corresponding bit in the INT_CTRL register is set (INT_CTRL[ENT_VAL] = 0).

After a power cycle or reset, the TRNG interrupts need to be enabled again. Note that these interrupt bits correspond to the MCTL[FCT_FAIL], MCTL[ENT_VAL] and MCTL[ERR]. These MCTL bits will always assert/deassert regardless of whether interrupts are masked or temporarily enabled/disabled. The interrupt registers mirror these MCTL bits but are independent.

Below is an example of an interrupt setup and programming flow for ENT_VAL. HW_ERR and FRQ_CT_FAIL interrupts can be enabled in the same way.

1. Initialize the TRNG normally to generate entropy, then poll for entropy valid bit (MCTL[ENT_VAL] = 1).
2. When entropy is ready, the active-low interrupt port should be expected to be high (ipi_rng_int_b = 1), because the entropy valid interrupt has not been enabled yet.
3. Make sure that the INT_MASK register does not have any interrupts enabled (INT_MASK[2:0] = 0x7). Wait/sleep for about 100 clocks.
4. Disable or ensure that INT_CTRL register has temporarily disabled the ENT_VAL interrupt (INT_CTRL[ENT_VAL] = 0). Wait/sleep for a few clocks, maybe 100 clocks.
5. The active-low interrupt port should still be high (ipi_rng_int_b = 1).
6. Read out entropy values from ENT0 to ENT15 registers and make sure ENT15 is read last. Values from all registers should appear random. If SDCTL[SAMP_SIZE] is less than 512, only the bits up to SAMP_SIZE will be random. For example, if SAMP_SIZE is 64, only bits in ENT14 and ENT15 will be random and valid.
7. Read out entropy values again from ENT0 to ENT15 registers. Verify that zeros are in all registers now after ENT15 is read.
8. Switch TRNG into programing mode by writing MCTL[PRGM] = 1.
9. Turn on the ENT_VAL interrupt functionality by setting INT_CTRL[ENT_VAL] = 1 and INT_MASK[ENT_VAL] = 1.
10. Switch TRNG into running mode by setting MCTL[PRGM] = 0 and MCTL[TRNG_ACC] = 1.
11. Wait or poll for entropy valid (MCTL[ENT_VAL] = 1).
12. When entropy is ready, the active-low interrupt is expected to trigger (ipi_rng_int_b = 0). This should also be reflected in the interrupt status register (INT_STATUS[ENT_VAL] = 1).
13. Read out all entropy registers or at least just ENT15 to clear entropy and start generation again.
14. The active-low interrupt is expected to be cleared (ipi_rng_int_b = 1). This should also be reflected in the interrupt status register (INT_STATUS[ENT_VAL] = 0).
15. Wait or poll for entropy valid (MCTL[ENT_VAL] = 1).
16. When entropy is ready, the active-low interrupt is expected to trigger (ipi_rng_int_b = 0). This should also be reflected in the interrupt status register (INT_STATUS[ENT_VAL] = 1).
17. The interrupt can be temporarily cleared before reading from ENT15 by
 - sleeping or waiting for 5 to 50 clocks, then
 - temporarily clearing the interrupt (INT_CTRL[ENT_VAL] = 0), then
 - sleeping or waiting for 5 to 50 clocks.
18. The active-low interrupt is expected to be cleared (ipi_rng_int_b = 1). This should also be reflected in the interrupt status register (INT_STATUS[ENT_VAL] = 0).
19. Read out all entropy registers or at least just ENT15 to clear entropy and start generation again.
20. Poll for the active-low interrupt port (ipi_rng_int_b = 0).
21. Read out all entropy registers or at least just ENT15 to clear entropy and start generation again.

22. Now mask any future ENT_VAL interrupt by setting INT_CTRL[ENT_VAL] = 0 and INT_MASK[ENT_VAL] = 0.
23. Even though the interrupts are now masked, the ENT_VAL event can still be polled by polling for MCTL[ENT_VAL] = 1.
24. When MCTL[ENT_VAL] = 1 is polled, the active-low interrupt port should NOT assert (ipi_rng_int_b = 1). The corresponding interrupt should also not assert in the interrupt status register (INT_STATUS[ENT_VAL] = 0).
25. The ENT_VAL interrupt can be enabled again by setting INT_MASK[ENT_VAL] = 1).
26. Read out all entropy registers or at least just ENT15 to clear entropy and start generation again.
27. Poll for the active-low interrupt port (ipi_rng_int_b = 0). The interrupt functionality should be enabled again.

27.4.3 Sample Size Adjustment Programming Flow

The sample size (SDCTL[SAMP_SIZE]) parameter can be adjusted to speed up or to limit the number of bits that an application wants to use. The higher the sample size, the better the entropy quality (the internal self tests are more sensitive in monitoring entropy quality). However, this slows down entropy generation. The tradeoff between the speed versus the quality of entropy needs to be balanced per application.

Every time the sample size parameter is updated, the bounds of the built-in self tests (BIST) must also be updated to match the acceptable error ranges for the new sample size. The permutations can be almost unlimited.

Below are the emperical options for some sample size parameters and their corresponding self bounds adjustment. For these options, the "else" case will be the fastest but with the lowest entropy quality of only 128 bits. In other words, there will only be valid random bits in registers ENT12 to ENT15. Bits in other registers will be invalid.

```
// example Seed Control Register (SDCTL) parameters
samp_size    = 128; // decimal
ent_dly_curr = 1000; // decimal

// target rejection ratio = 1.00E-07

if ( samp_size == 256 ) {
    reg32_write( TRNG_SCMISC, 0x0001001F ); // retry once, long run max 31

    // min range max
    reg32_write( TRNG_SCML, 0x005600AB ); // 85 86 171 [ monobit limit]
    reg32_write( TRNG_SCR1L, 0x0038003f ); // 7 56 63 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x00260026 ); // 0 38 38 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x001A0019 ); // -1 26 25 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x00120011 ); // -1 18 17 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x000E000D ); // -1 14 13 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x000D000C ); // -1 13 12 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x000001FF );
    reg32_write( TRNG_PKRRNG, 0x000000FC ); // 260 252 511 [ poker test]
} else if ( samp_size == 384 ) {
    reg32_write( TRNG_SCMISC, 0x0001001F ); // retry once, long run max 31

    // min range max
    reg32_write( TRNG_SCML, 0x008B00F4 ); // 139 105 244 [ monobit limit]
    reg32_write( TRNG_SCR1L, 0x00450056 ); // 17 69 86 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x002F0032 ); // 3 47 50 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x00210020 ); // -1 33 32 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x00170016 ); // -1 23 22 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x0010000F ); // -1 16 15 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x0010000F ); // -1 16 15 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x000003BF );
    reg32_write( TRNG_PKRRNG, 0x0000017B ); // 581 379 959 [ poker test]
} else if ( samp_size == 512 ) {
```

```

reg32_write( TRNG_SCMISC, 0x00010020 ); // retry once, long run max 32

// min range max
reg32_write( TRNG_SCML, 0x007A013D ); // 195 122 317 [ monobit limit]
reg32_write( TRNG_SCR1L, 0x0050006B ); // 27 80 107 [run length 1 limit]
reg32_write( TRNG_SCR2L, 0x0037003E ); // 7 55 62 [run length 2 limit]
reg32_write( TRNG_SCR3L, 0x00270027 ); // 0 39 39 [run length 3 limit]

reg32_write( TRNG_SCR4L, 0x001B001A ); // -1 27 26 [run length 4 limit]
reg32_write( TRNG_SCR5L, 0x00130012 ); // -1 19 18 [run length 5 limit]
reg32_write( TRNG_SCR6PL, 0x00120011 ); // -1 18 17 [run length 6+ limit]
reg32_write( TRNG_PKRMAX, 0x00000640 );
reg32_write( TRNG_PKRRNG, 0x0000023A ); // 1031 570 1600 [ poker test]
} else if ( samp_size == 1000 ) {
    reg32_write( TRNG_SCMISC, 0x00010021 ); // retry once, long run max 33

    // min range max
    reg32_write( TRNG_SCML, 0x00AA0249 ); // 415 170 585 [ monobit limit]
    reg32_write( TRNG_SCR1L, 0x007000B8 ); // 72 112 184 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x004D0068 ); // 27 77 104 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x0037003E ); // 7 55 62 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x00280027 ); // -1 40 39 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x001B001A ); // -1 27 26 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x001A0019 ); // -1 26 25 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x00001329 );
    reg32_write( TRNG_PKRRNG, 0x000003DB ); // 3919 987 4905 [ poker test]
} else if ( samp_size == 1024 ) {
    reg32_write( TRNG_SCMISC, 0x00010021 ); // retry once, long run max 33

    // min range max
    reg32_write( TRNG_SCML, 0x00AC0256 ); // 426 172 598 [ monobit limit]
    reg32_write( TRNG_SCR1L, 0x007200BC ); // 74 114 188 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x004D0069 ); // 28 77 105 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x0038003F ); // 7 56 63 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x00280028 ); // 0 40 40 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x001B001A ); // -1 27 26 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x001B001A ); // -1 27 26 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x00001438 );
    reg32_write( TRNG_PKRRNG, 0x0000042C ); // 4109 1068 5176 [ poker test]
} else if ( samp_size == 2000 ) {
    reg32_write( TRNG_SCMISC, 0x00010022 ); // retry once, long run max 34

    // min range max
    reg32_write( TRNG_SCML, 0x00F00460 ); // 880 240 1120 [ monobit limit]
    reg32_write( TRNG_SCR1L, 0x009F014D ); // 174 159 333 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x006C00B5 ); // 73 108 181 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x004E0069 ); // 27 78 105 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x0039003F ); // 6 57 63 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x00290028 ); // -1 41 40 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x00280028 ); // 0 40 40 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x000044D8 );
    reg32_write( TRNG_PKRRNG, 0x000007B7 ); // 15650 1975 17624 [ poker test]
} else if ( samp_size == 2500 ) {
    reg32_write( TRNG_SCMISC, 0x00010022 ); //retry once, long run max 34

    // min range max

```



```

reg32_write( TRNG_SCML, 0x010C0568 ); // 1116 268 1384 [ monobit limit]
reg32_write( TRNG_SCR1L, 0x00B10194 ); // 227 177 404 [run length 1 limit]
reg32_write( TRNG_SCR2L, 0x007900DC ); // 99 121 220 [run length 2 limit]
reg32_write( TRNG_SCR3L, 0x0057007D ); // 38 87 125 [run length 3 limit]

reg32_write( TRNG_SCR4L, 0x003F004A ); // 11 63 74 [run length 4 limit]
reg32_write( TRNG_SCR5L, 0x002D002E ); // 1 45 46 [run length 5 limit]
reg32_write( TRNG_SCR6PL, 0x002D002E ); // 1 45 46 [run length 6+ limit]
reg32_write( TRNG_PKRMAX, 0x00006920 );
reg32_write( TRNG_PKRRNG, 0x000009A3 ); // 24446 2467 26912 [ poker test]
} else {
    reg32_write( TRNG_SCMISC, 0x0001001D ); // retry once, long run max 29

    // min range max
    reg32_write( TRNG_SCML, 0x003D005E ); // 33 61 94 monobit limit]
    reg32_write( TRNG_SCR1L, 0x00270027 ); // 0 39 39 [run length 1 limit]
    reg32_write( TRNG_SCR2L, 0x00190018 ); // -1 25 24 [run length 2 limit]
    reg32_write( TRNG_SCR3L, 0x00120011 ); // -1 18 17 [run length 3 limit]

    reg32_write( TRNG_SCR4L, 0x000D000C ); // -1 13 12 [run length 4 limit]
    reg32_write( TRNG_SCR5L, 0x000A0009 ); // -1 10 9 [run length 5 limit]
    reg32_write( TRNG_SCR6PL, 0x000A0009 ); // -1 10 9 [run length 6+ limit]
    reg32_write( TRNG_PKRMAX, 0x000000DA );
    reg32_write( TRNG_PKRRNG, 0x00000099 ); // 66 153 218 [ poker test]
}

// adjust delay
reg32_write( TRNG_SDCTL, ( (uint32_t)ent_dly_curr << 16 |
                          (uint32_t)samp_size
                          )
);

```

Once the sample size parameter has been set, entropy can be requested in a normally without generating any errors.

27.5 Other Applications

The TRNG can also be used by a post-processing pseudo-random number generator function. For example, TRNG can be used to seed a hardware- or software-based implementation of a deterministic random bit generator (DRBG) as defined by SP800-90.

27.6 TRNG register descriptions

All accesses of undefined addresses always return zero and assert IPS transfer error. Writes to undefined and read-only addresses are ignored. Undefined addresses are those undocumented, protected or reserved addresses within and outside the range of the addresses defined in the memory map below. The register addresses shown in the Memory Map below represents how these registers are accessed over the register bus as 32-bit words. Accesses to registers must use full-word (32-bit) reads or writes.

The format and fields in each TRNG register are defined below. Some of the register format figures apply to several different registers. In such cases a different register name will be associated with each of the register offset addresses that appear at the top of the register format figure. Although these registers share the same format, they are independent registers. In addition, many registers can be accessed at multiple addresses. In these cases there will be a single register name and the list of addresses at which that register is accessible will be indicated as aliases. Unless noted in the individual register descriptions, registers are reset only at Power-On Reset (POR) or with an asynchronous system hard reset.

27.6.1 TRNG memory map

TRNG.TRNG base address: 4013_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Miscellaneous Control Register (MCTL)	32	RW	0001_2001h
4h	Statistical Check Miscellaneous Register (SCMISC)	32	RW	0001_0022h
8h	Poker Range Register (PKRRNG)	32	RW	0000_09A3h
Ch	Poker Maximum Limit Register (PKRMAX)	32	RW	0000_6920h
Ch	Poker Square Calculation Result Register (PKRSQ)	32	R	0000_0000h
10h	Seed Control Register (SDCTL)	32	RW	0C80_09C4h
14h	Sparse Bit Limit Register (SBLIM)	32	RW	0000_003Fh
14h	Total Samples Register (TOTSAM)	32	R	0000_0000h
18h	Frequency Count Minimum Limit Register (FRQMIN)	32	RW	0000_0640h
1Ch	Frequency Count Register (FRQCNT)	32	R	0000_0000h
1Ch	Frequency Count Maximum Limit Register (FRQMAX)	32	RW	0000_6400h
20h	Statistical Check Monobit Count Register (SCMC)	32	R	0000_0000h
20h	Statistical Check Monobit Limit Register (SCML)	32	RW	010C_0568h
24h	Statistical Check Run Length 1 Count Register (SCR1C)	32	R	0000_0000h
24h	Statistical Check Run Length 1 Limit Register (SCR1L)	32	RW	00B2_0195h
28h	Statistical Check Run Length 2 Count Register (SCR2C)	32	R	0000_0000h
28h	Statistical Check Run Length 2 Limit Register (SCR2L)	32	RW	007A_00DCh
2Ch	Statistical Check Run Length 3 Count Register (SCR3C)	32	R	0000_0000h
2Ch	Statistical Check Run Length 3 Limit Register (SCR3L)	32	RW	0058_007Dh
30h	Statistical Check Run Length 4 Count Register (SCR4C)	32	R	0000_0000h
30h	Statistical Check Run Length 4 Limit Register (SCR4L)	32	RW	0040_004Bh
34h	Statistical Check Run Length 5 Count Register (SCR5C)	32	R	0000_0000h
34h	Statistical Check Run Length 5 Limit Register (SCR5L)	32	RW	002E_002Fh
38h	Statistical Check Run Length 6+ Count Register (SCR6PC)	32	R	0000_0000h
38h	Statistical Check Run Length 6+ Limit Register (SCR6PL)	32	RW	002E_002Fh
3Ch	Status Register (STATUS)	32	R	0000_0000h
40h - 7Ch	Entropy Read Register (ENT0 - ENT15)	32	R	0000_0000h
80h	Statistical Check Poker Count 1 and 0 Register (PKRCNT10)	32	R	0000_0000h
84h	Statistical Check Poker Count 3 and 2 Register (PKRCNT32)	32	R	0000_0000h
88h	Statistical Check Poker Count 5 and 4 Register (PKRCNT54)	32	R	0000_0000h
8Ch	Statistical Check Poker Count 7 and 6 Register (PKRCNT76)	32	R	0000_0000h
90h	Statistical Check Poker Count 9 and 8 Register (PKRCNT98)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
94h	Statistical Check Poker Count B and A Register (PKRCNTBA)	32	R	0000_0000h
98h	Statistical Check Poker Count D and C Register (PKRCNTDC)	32	R	0000_0000h
9Ch	Statistical Check Poker Count F and E Register (PKRCNTFE)	32	R	0000_0000h
A0h	Security Configuration Register (SEC_CFG)	32	RW	0000_0000h
A4h	Interrupt Control Register (INT_CTRL)	32	RW	FFFF_FFFFh
A8h	Mask Register (INT_MASK)	32	RW	0000_0000h
ACh	Interrupt Status Register (INT_STATUS)	32	R	0000_0000h
F0h	Version ID Register (MS) (VID1)	32	R	0030_0104h
F4h	Version ID Register (LS) (VID2)	32	R	0000_0000h

27.6.2 Miscellaneous Control Register (MCTL)

Offset

Register	Offset
MCTL	0h

Function

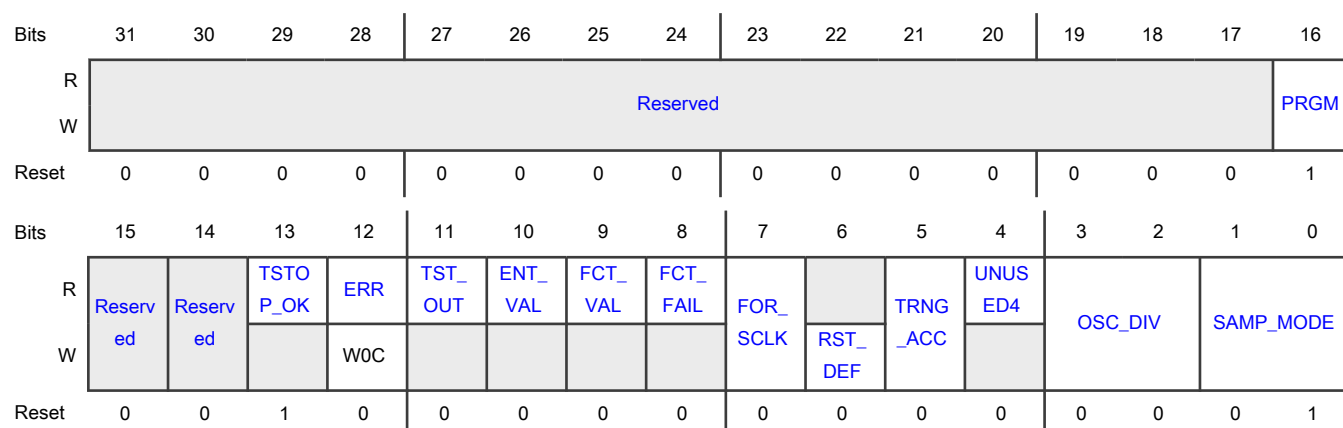
This register is intended to be used for programming, configuring, and testing the RNG. It is the main register to read/write in order to enable entropy generation, to stop entropy generation, and to block access to entropy registers. This is done via the TRNG_ACC and PRGM bit below.

The Miscellaneous Control Register is a read/write register used to control the RNG's True Random Number Generator (TRNG) access, operation and test.

NOTE

Note that in many cases two RNG registers share the same address, and a particular register at the shared address is selected based upon the value in the PRGM field of the MCTL register.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 PRGM	Programming Mode Select. When this bit is 1, the TRNG is in Program Mode, otherwise it is in Run Mode. No Entropy value will be generated while the TRNG is in Program Mode. Note that different RNG registers are accessible at the same address depending on whether PRGM is set to 1 or 0. This is noted in the RNG register descriptions. NOTE If PRGM is set to 1, then TRNG_ACC should also be set to 1.
15 —	Reserved
14 —	Reserved
13 TSTOP_OK	TRNG_OK_TO_STOP. Software can check that this bit is a 1 before transitioning TRNG to low power mode (TRNG clock stopped). TRNG turns on the TRNG free-running ring oscillator whenever new entropy is being generated and turns off the ring oscillator when entropy generation is complete. If the TRNG clock is stopped while the TRNG ring oscillator is running, the oscillator will continue running even though the TRNG clock is stopped. To make sure the ring oscillator is stopped, assert ipg_stop input and verify ipg_rng_stop_ack is asserted after at least two (2) clock cycles. TSTOP_OK is asserted when the TRNG ring oscillator is not running.
12 ERR	Read: Error status. 1 = error detected. 0 = no error. Write: Write 1 to clear errors. Writing 0 has no effect.
11 TST_OUT	Read only: Test point inside ring oscillator.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 ENT_VAL	Read only: Entropy Valid. Will assert after an entropy value is generated and only if TRNG_ACC bit is set. Will be cleared at most one (1) bus clock cycle after reading the ENT15 register. (ENT0 through ENT14 should be read before reading ENT15).
9 FCT_VAL	Read only: Frequency Count Valid. Indicates that a valid frequency count may be read from FRQCNT.
8 FCT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. This error may be cleared by writing a 1 to the ERR bit.
7 FOR_SCLK	Force System Clock. If set, the system clock is used to operate the TRNG, instead of the ring oscillator. This is for test use only, and indeterminate results may occur. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1.
6 RST_DEF	Reset Defaults. Writing a 1 to this bit clears various TRNG registers, and bits within registers, to their default state. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. Reading this bit always produces a 0.
5 TRNG_ACC	TRNG Access Mode. If this bit is set to 1, the TRNG will generate an Entropy value that can be read via the ENT0-ENT15 registers. The Entropy value may be read once the ENT_VAL bit is asserted. Also see ENTa register descriptions (For a = 0 to 15). NOTE If PRGM is set to 1, then TRNG_ACC should also be set to 1.
4 UNUSED4	This bit is unused. Always reads zero.
3-2 OSC_DIV	Oscillator Divide. Determines the amount of dividing done to the ring oscillator before it is used by the TRNG. This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1. 00b - use ring oscillator with no divide 01b - use ring oscillator divided-by-2 10b - use ring oscillator divided-by-4 11b - use ring oscillator divided-by-8
1-0 SAMP_MODE	Sample Mode. Determines the method of sampling the ring oscillator while generating the Entropy value: This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously with writing this field. This field is cleared to the POR default value by writing the RST_DEF bit to 1. 00b - use Von Neumann data into both Entropy shifter and Statistical Checker 01b - use raw data into both Entropy shifter and Statistical Checker 10b - use Von Neumann data into Entropy shifter. Use raw data into Statistical Checker

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - undefined/reserved.

27.6.3 Statistical Check Miscellaneous Register (SCMISC)

Offset

Register	Offset
SCMISC	4h

Function

The Statistical Check Miscellaneous Register contains the Long Run Maximum Limit value and the Retry Count value. This register is accessible only when the MCTL[PRGM] bit is 1, otherwise this register will read zeroes, and cannot be written.

Diagram

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

Fields

Field	Function
31-20 —	Reserved
19-16 RTY_CT	RETRY COUNT. If a statistical check fails during the TRNG Entropy Generation, the RTY_CT value indicates the number of times a retry should occur before generating an error. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 LRUN_MAX	LONG RUN MAX LIMIT. This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the MCTL[RST_DEF] bit to 1.

27.6.4 Poker Range Register (PKRRNG)

Offset

Register	Offset
PKRRNG	8h

Function

The Poker Range Register defines the difference between the TRNG Poker Maximum Limit and the minimum limit. These limits are used during the TRNG Statistical Check Poker Test.

Diagram

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

Fields

Field	Function
31-16 —	Reserved Always 0.
15-0 PKR_RNG	Poker Range. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register, and the minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the minimum allowable Poker result is PKR_MAX - PKR_RNG + 1.

27.6.5 Poker Maximum Limit Register (PKRMAX)

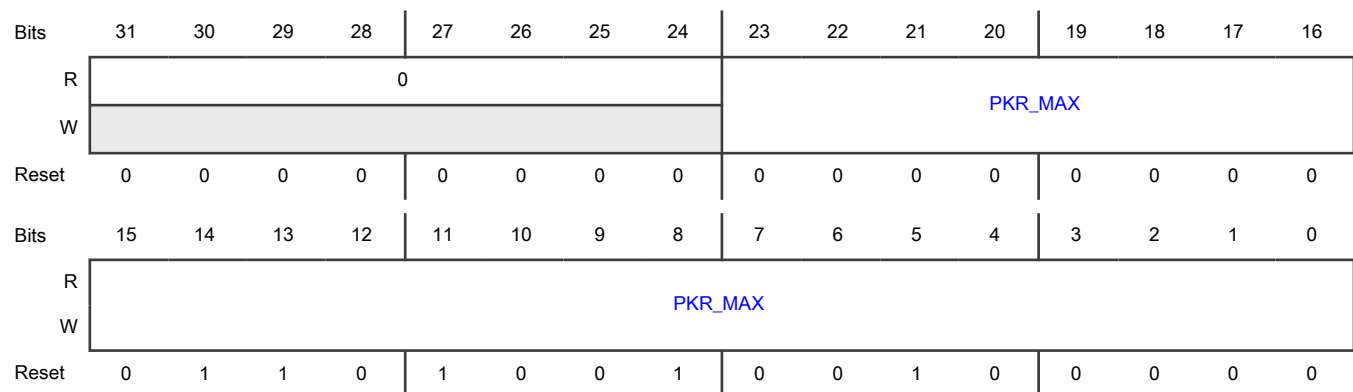
Offset

Register	Offset	Description
PKRMAX	Ch	Accessible at this address when MCTL[PRGM] = 1]

Function

The Poker Maximum Limit Register defines Maximum Limit allowable during the TRNG Statistical Check Poker Test. Note that this offset (0x0C) is used as PKRMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as the PKRSQ readback register.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 PKR_MAX	Poker Maximum Limit. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the PKRMAX[PKR_MAX] register. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the PKRMAX and PKRRNG registers combined are used to define the minimum allowable Poker result, which is $PKR\_MAX - PKR\_RNG + 1$. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Poker Test Square Calculation result in register PKRSQ, as defined in the following section.

27.6.6 Poker Square Calculation Result Register (PKRSQ)

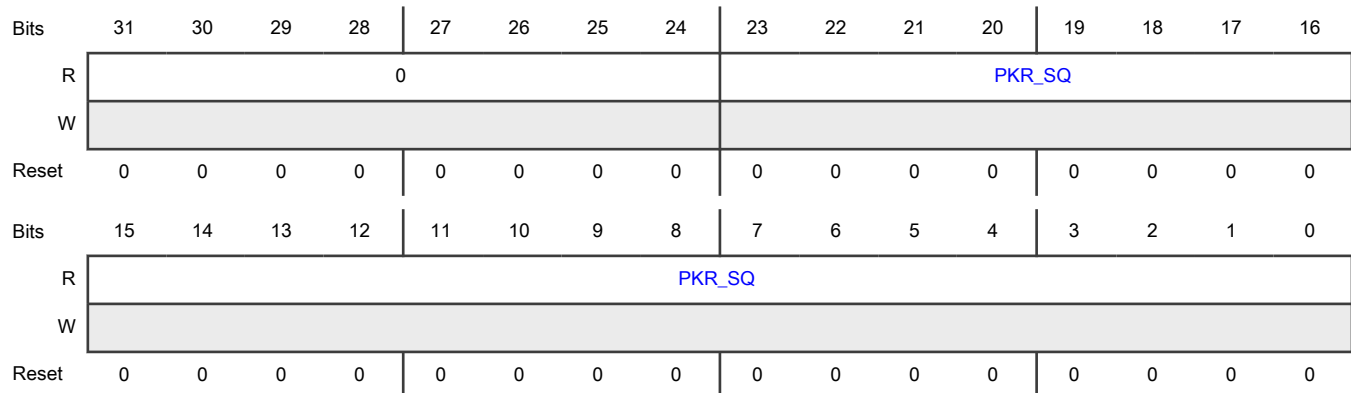
Offset

Register	Offset	Description
PKRSQ	Ch	Accessible at this address when MCTL[PRGM] = 0]

Function

The Poker Square Calculation Result Register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed PKRRNG value. Note that this offset (0x0C) is used as PKRMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as PKRSQ readback register, as described here.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 PKR_SQ	Poker Square Calculation Result. During the TRNG Statistical Checks, a "Poker Test" is run which starts with the value PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm, and must remain greater than zero, but less than the PKRRNG[PKR_RNG] limit. The resulting value may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register PKRMAX, as defined in the previous section.

27.6.7 Seed Control Register (SDCTL)

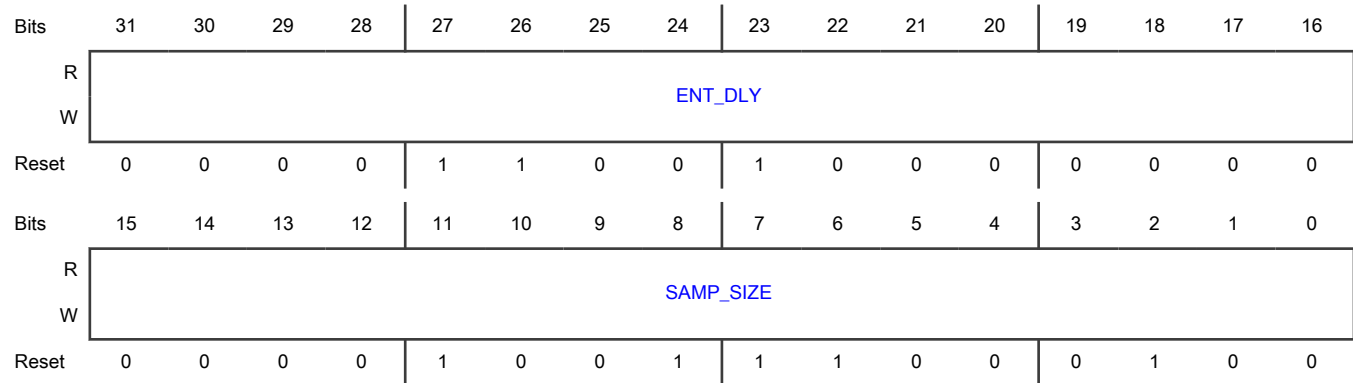
Offset

Register	Offset
SDCTL	10h

Function

The Seed Control Register contains two fields. One field defines the length (in system clocks) of each Entropy sample (ENT_DLY), and the other field indicates the number of samples that will taken during each TRNG Entropy generation (SAMP_SIZE).

Diagram



Fields

Field	Function
31-16 ENT_DLY	Entropy Delay. Defines the length (in system clocks) of each Entropy sample taken. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.
15-0 SAMP_SIZE	Sample Size. Defines the total number of Entropy samples that will be taken during Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

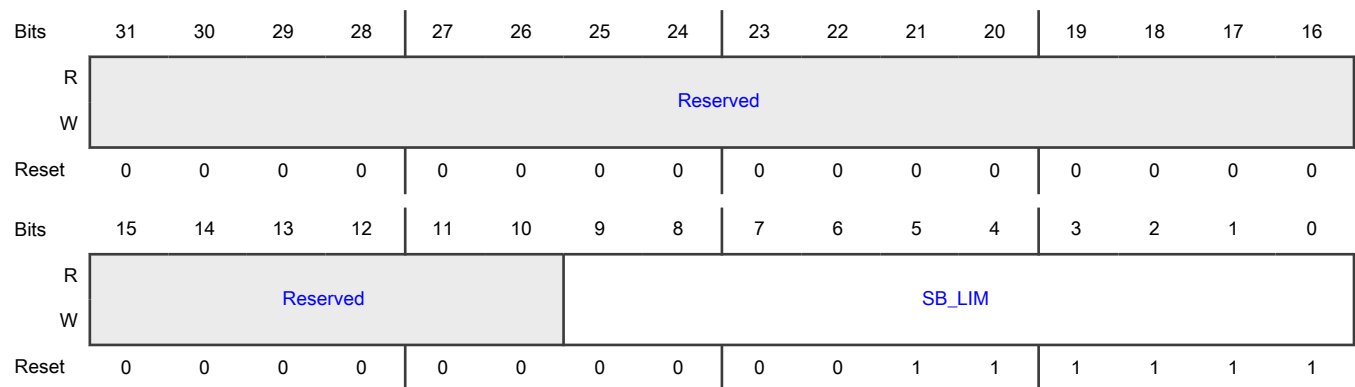
27.6.8 Sparse Bit Limit Register (SBLIM)

Offset

Register	Offset	Description
SBLIM	14h	Accessible at this address when MCTL[PRGM] = 1]

Function

The Sparse Bit Limit Register is used when Von Neumann sampling is selected during Entropy Generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated. Note that this address (0x14) is used as SBLIM only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as TOTSAM readback register.

Diagram**Fields**

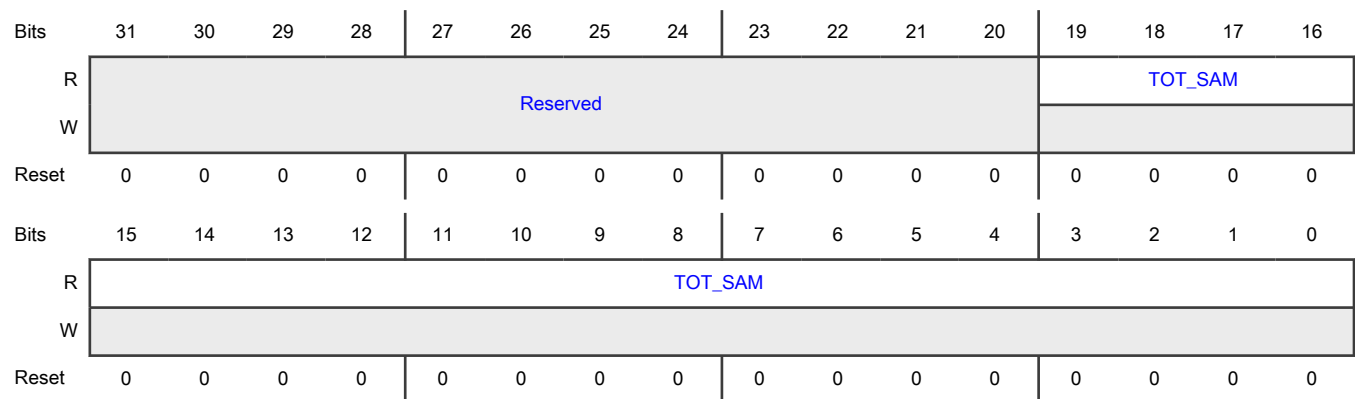
Field	Function
31-10 —	Reserved Always 0.
9-0 SB_LIM	Sparse Bit Limit. During Von Neumann sampling (if enabled by MCTL[SAMP_MODE], samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient Entropy. The Sparse Bit Limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Total Samples count in register TOTSAM, as defined in the following section.

27.6.9 Total Samples Register (TOTSAM)**Offset**

Register	Offset	Description
TOTSAM	14h	Accessible at this address when MCTL[PRGM] = 0]

Function

The Total Samples Register is a read-only register used to read the total number of samples taken during Entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling. Note that this offset (0x14) is used as SBLIM if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as TOTSAM readback register, as described here.

Diagram**Fields**

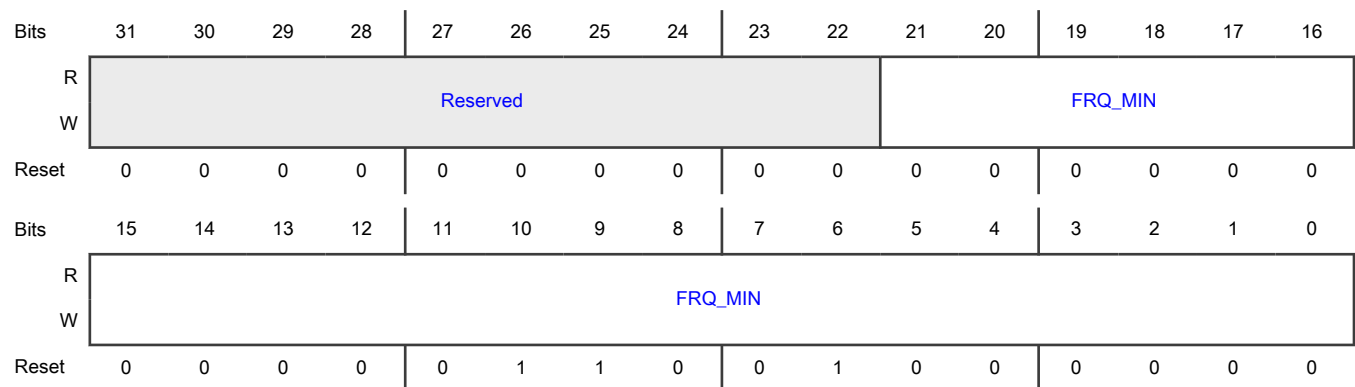
Field	Function
31-20 —	Reserved Always 0.
19-0 TOT_SAM	Total Samples. During Entropy generation, the total number of raw samples is counted. This count is useful in determining how often a sample is used during Von Neumann sampling. The count may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Sparse Bit Limit in register SBLIM, as defined in the previous section.

27.6.10 Frequency Count Minimum Limit Register (FRQMIN)**Offset**

Register	Offset
FRQMIN	18h

Function

The Frequency Count Minimum Limit Register defines the minimum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is less than this programmed minimum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated.

Diagram**Fields**

Field	Function
31-22 —	Reserved Always 0.
21-0 FRQ_MIN	Frequency Count Minimum Limit. Defines the minimum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.

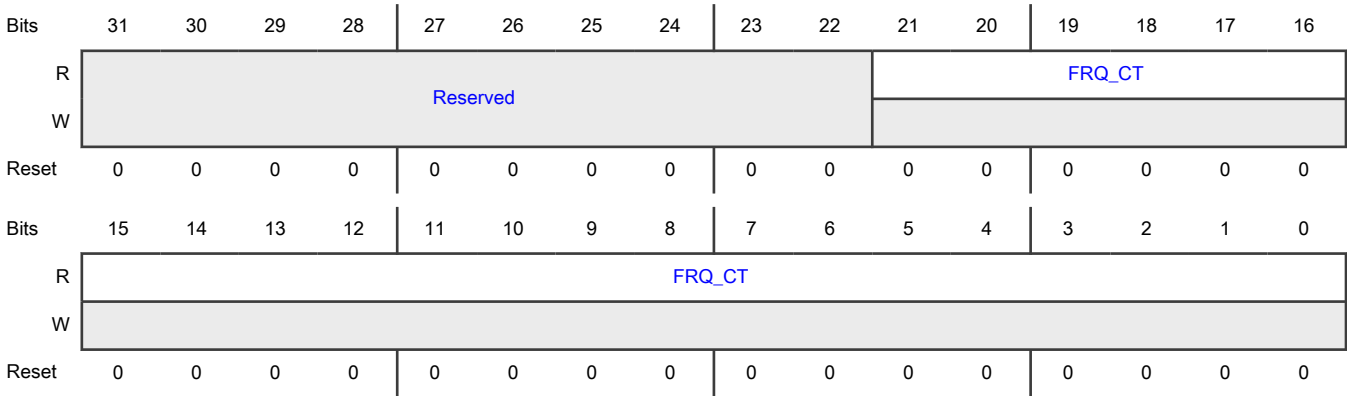
27.6.11 Frequency Count Register (FRQCNT)**Offset**

Register	Offset	Description
FRQCNT	1Ch	Accessible at this address when MCTL[PRGM] = 0]

Function

The Frequency Count Register is a read-only register used to read the frequency counter within the TRNG entropy generator. It will read all zeroes unless PRGM[MCTL] = 0 and MCTL[TRNG_ACC] = 1. Note that this offset (0x1C) is used as FRQMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as live FRQCNT readback register, as described here. This register must be read once, as a full 32-bit value. A subsequent read is only valid when MCTL[FCT_VAL] bit is HIGH. A read clears this register. The next read when MCTL[FCT_VAL] bit is HIGH must be treated as a different value from the previous one, even though the counts might be the same.

Diagram



Fields

Field	Function
31-22 —	Reserved Always 0.
21-0 FRQ_CT	Frequency Count. If MCTL[TRNG_ACC] = 1, Reads a sample frequency count taken during entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Frequency Maximum Limit in register FRQMAX, as defined in the FRQ_MAX field.

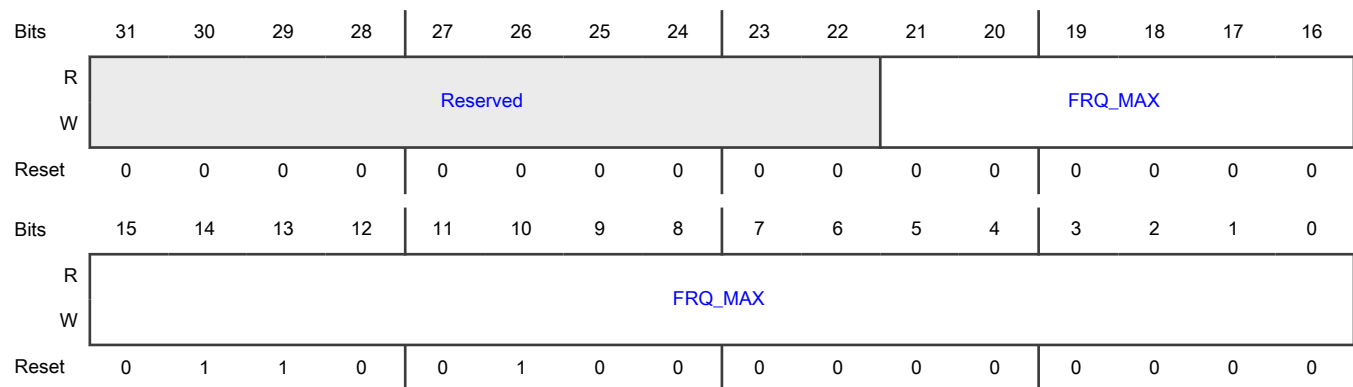
27.6.12 Frequency Count Maximum Limit Register (FRQMAX)

Offset

Register	Offset	Description
FRQMAX	1Ch	Accessible at this address when MCTL[PRGM] = 1]

Function

The Frequency Count Maximum Limit Register defines the maximum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is greater than this programmed maximum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated. Note that this address (001C) is used as FRQMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as FRQCNT readback register.

Diagram**Fields**

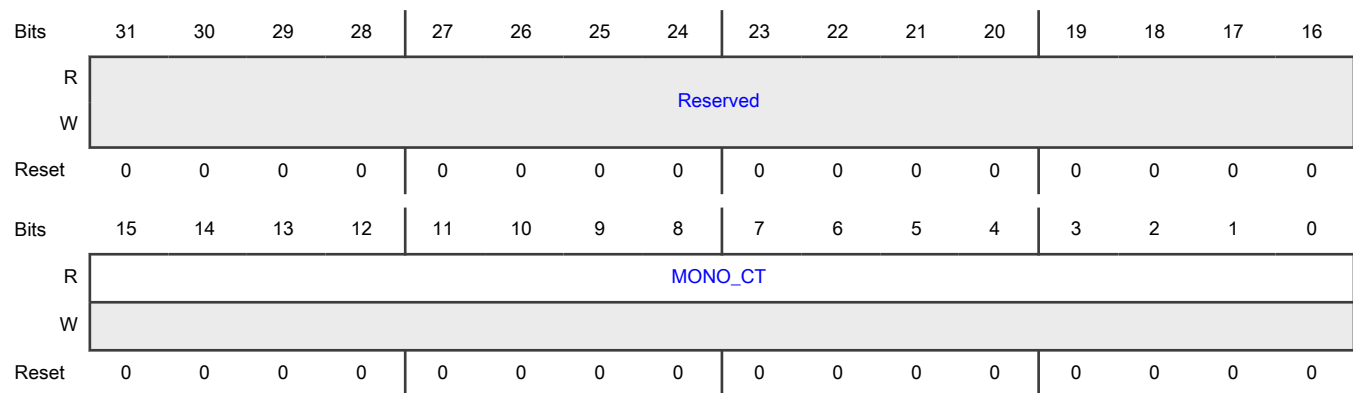
Field	Function
31-22 —	Reserved Always 0.
21-0 FRQ_MAX	Frequency Counter Maximum Limit. Defines the maximum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Frequency Count result in register FRQCNT, as defined in the FRQ_CT field.

27.6.13 Statistical Check Monobit Count Register (SCMC)**Offset**

Register	Offset	Description
SCMC	20h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Monobit Count Register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in SCML[MONO_MAX], and is decremented each time a one is sampled. Note that this offset (0x20) is used as SCML if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register, as described here.

Diagram**Fields**

Field	Function
31-16 —	Reserved Always 0.
15-0 MONO_CT	Monobit Count. Reads the final Monobit count after entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Statistical Check Monobit Limit in register SCML, as defined in the previous section.

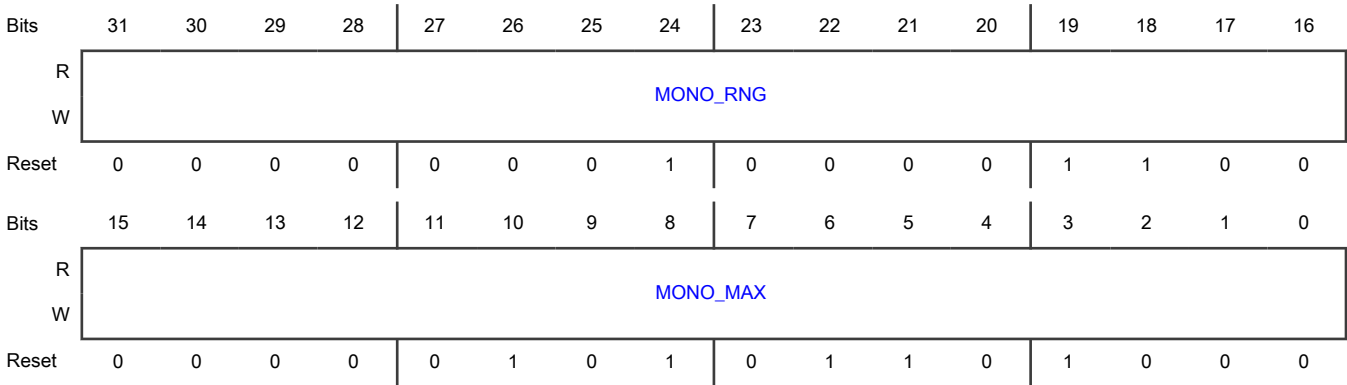
27.6.14 Statistical Check Monobit Limit Register (SCML)**Offset**

Register	Offset	Description
SCML	20h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Monobit Limit Register defines the allowable maximum and minimum number of ones/zero detected during entropy generation. To pass the test, the number of ones/zeros generated must be less than the programmed maximum value, and the number of ones/zeros generated must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x20) is used as SCML only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register.

Diagram



Fields

Field	Function
31-16 MONO_RNG	Monobit Range. The number of ones/zeroes detected during entropy generation must be greater than MONO_MAX - MONO_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-0 MONO_MAX	Monobit Maximum Limit. Defines the maximum allowable count taken during entropy generation. The number of ones/zeroes detected during entropy generation must be less than MONO_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.15 Statistical Check Run Length 1 Count Register (SCR1C)

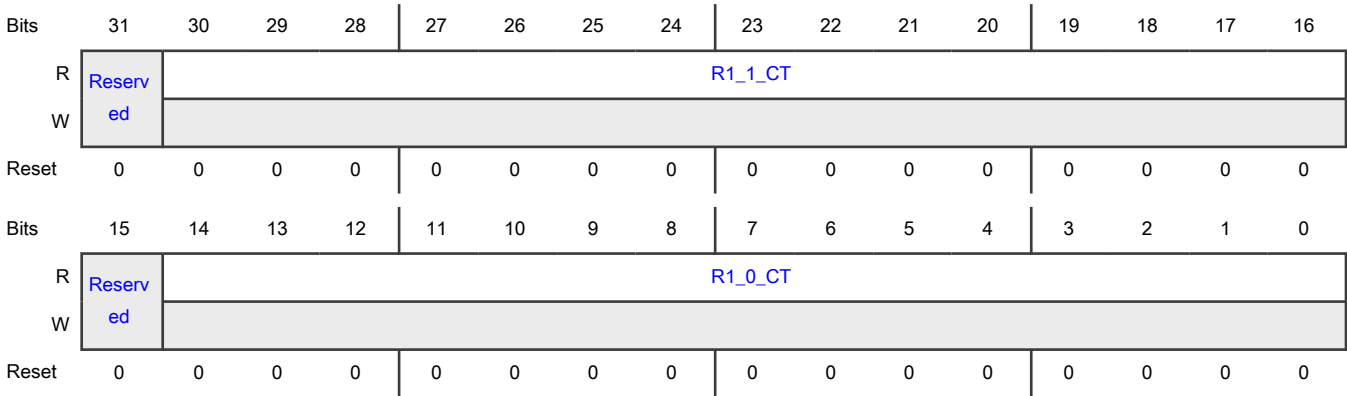
Offset

Register	Offset	Description
SCR1C	24h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 1 Counters Register is a read-only register used to read the final Run Length 1 counts after entropy generation. These counters start with the value in SCR1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one). Note that this offset (0x24) is used as SCR1L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR1C readback register, as described here.

Diagram



Fields

Field	Function
31 —	Reserved Always 0.
30-16 R1_1_CT	Runs of One, Length 1 Count. Reads the final Runs of Ones, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.
15 —	Reserved Always 0.
14-0 R1_0_CT	Runs of Zero, Length 1 Count. Reads the final Runs of Zeroes, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.

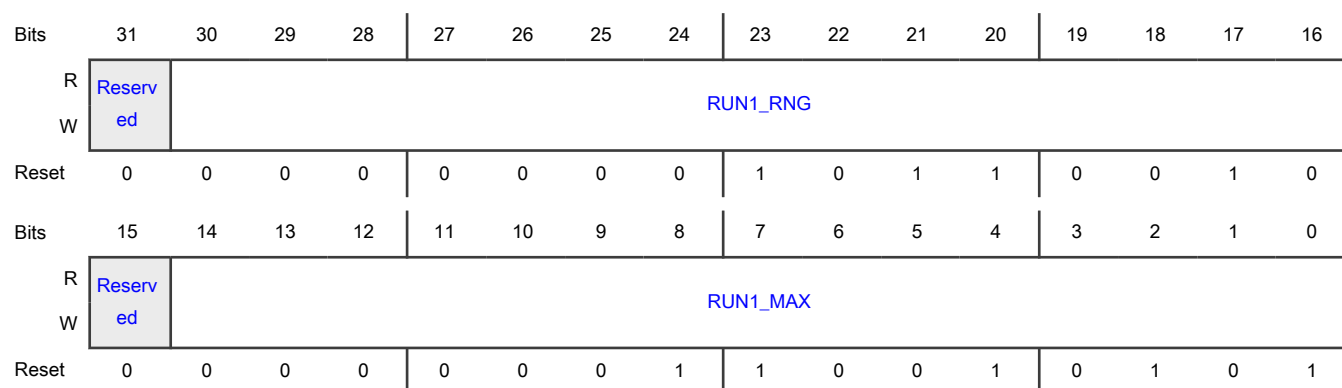
27.6.16 Statistical Check Run Length 1 Limit Register (SCR1L)

Offset

Register	Offset	Description
SCR1L	24h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 1 Limit Register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 1 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x24) is used as SCR1L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR1C readback register.

Diagram**Fields**

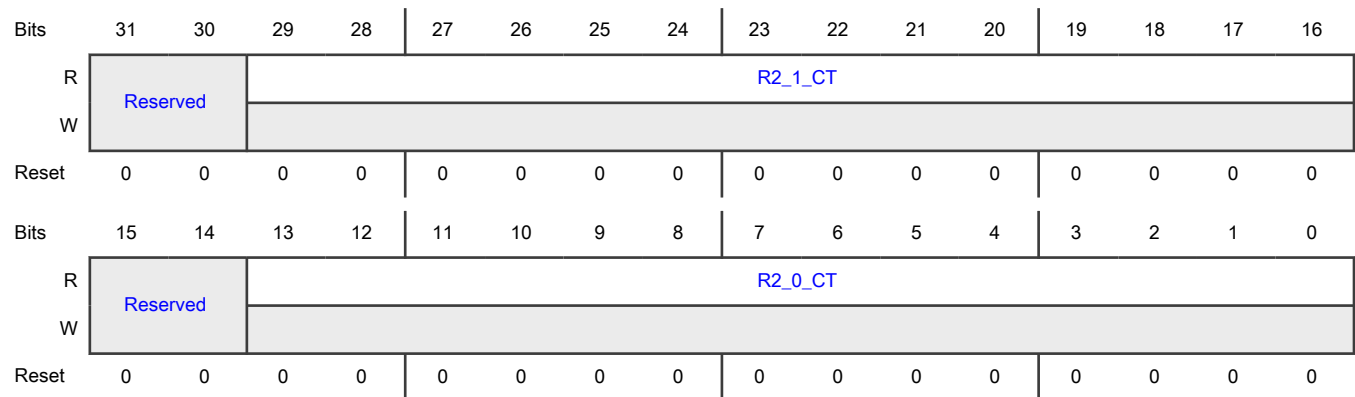
Field	Function
31 —	Reserved Always 0.
30-16 RUN1_RNG	Run Length 1 Range. The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX - RUN1_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15 —	Reserved Always 0.
14-0 RUN1_MAX	Run Length 1 Maximum Limit. Defines the maximum allowable runs of length 1 (for both 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.17 Statistical Check Run Length 2 Count Register (SCR2C)**Offset**

Register	Offset	Description
SCR2C	28h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 2 Counters Register is a read-only register used to read the final Run Length 2 counts after entropy generation. These counters start with the value in SCR2L[RUN2_MAX]. The R2_1_CT decrements each time two consecutive ones are sampled (preceded by a zero and followed by a zero). The R2_0_CT decrements each time two consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x28) is used as SCR2L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR2C readback register, as described here.

Diagram**Fields**

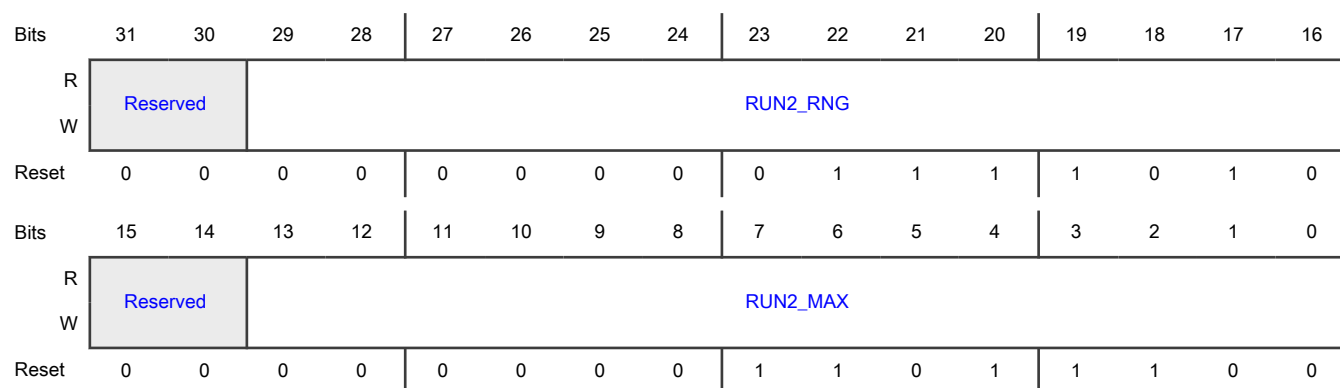
Field	Function
31-30 —	Reserved Always 0.
29-16 R2_1_CT	Runs of One, Length 2 Count. Reads the final Runs of Ones, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.
15-14 —	Reserved Always 0.
13-0 R2_0_CT	Runs of Zero, Length 2 Count. Reads the final Runs of Zeroes, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.

27.6.18 Statistical Check Run Length 2 Limit Register (SCR2L)**Offset**

Register	Offset	Description
SCR2L	28h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 2 Limit Register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 2 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x28) is used as SCR2L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR2C readback register.

Diagram**Fields**

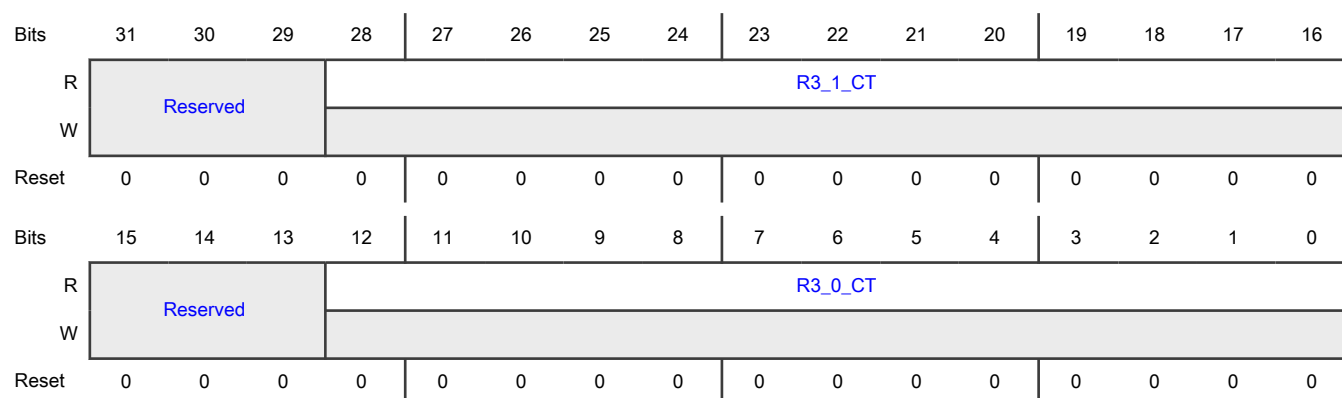
Field	Function
31-30 —	Reserved Always 0.
29-16 RUN2_RNG	Run Length 2 Range. The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX - RUN2_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-14 —	Reserved Always 0.
13-0 RUN2_MAX	Run Length 2 Maximum Limit. Defines the maximum allowable runs of length 2 (for both 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.19 Statistical Check Run Length 3 Count Register (SCR3C)**Offset**

Register	Offset	Description
SCR3C	2Ch	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 3 Counters Register is a read-only register used to read the final Run Length 3 counts after entropy generation. These counters start with the value in SCR3L[RUN3_MAX]. The R3_1_CT decrements each time three consecutive ones are sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time three consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x2C) is used as SCR3L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR3C readback register, as described here.

Diagram**Fields**

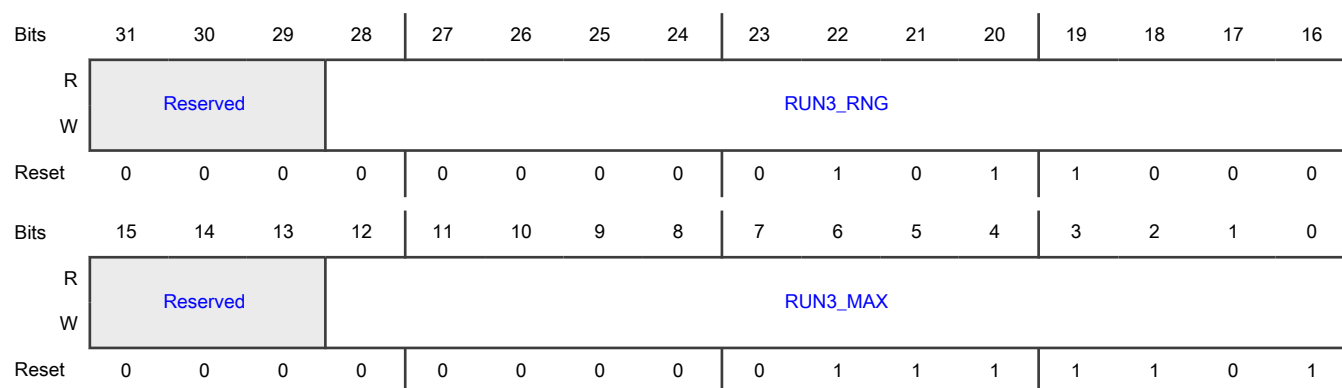
Field	Function
31-29 —	Reserved Always 0.
28-16 R3_1_CT	Runs of Ones, Length 3 Count. Reads the final Runs of Ones, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.
15-13 —	Reserved Always 0.
12-0 R3_0_CT	Runs of Zeroes, Length 3 Count. Reads the final Runs of Zeroes, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.

27.6.20 Statistical Check Run Length 3 Limit Register (SCR3L)**Offset**

Register	Offset	Description
SCR3L	2Ch	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 3 Limit Register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 3 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x2C) is used as SCR3L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR3C readback register.

Diagram**Fields**

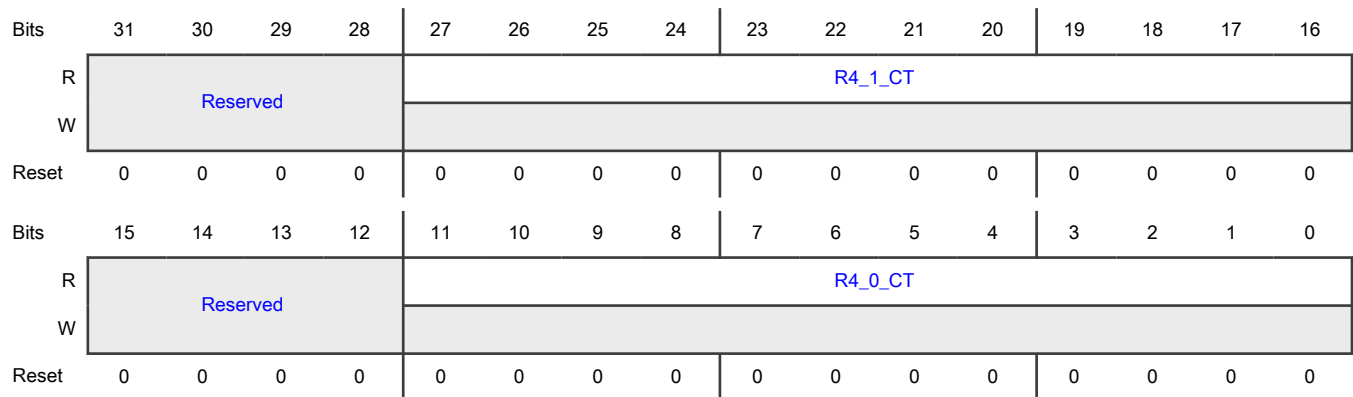
Field	Function
31-29 —	Reserved Always 0.
28-16 RUN3_RNG	Run Length 3 Range. The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX - RUN3_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-13 —	Reserved Always 0.
12-0 RUN3_MAX	Run Length 3 Maximum Limit. Defines the maximum allowable runs of length 3 (for both 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.21 Statistical Check Run Length 4 Count Register (SCR4C)**Offset**

Register	Offset	Description
SCR4C	30h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 4 Counters Register is a read-only register used to read the final Run Length 4 counts after entropy generation. These counters start with the value in SCR4L[RUN4_MAX]. The R4_1_CT decrements each time four consecutive ones are sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time four consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x30) is used as SCR4L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR4C readback register, as described here.

Diagram**Fields**

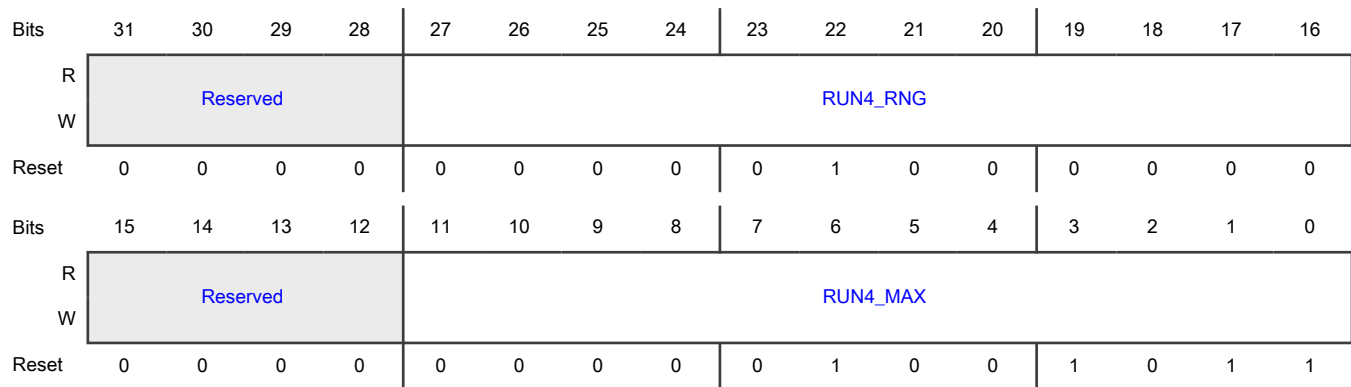
Field	Function
31-28 —	Reserved Always 0.
27-16 R4_1_CT	Runs of One, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.
15-12 —	Reserved Always 0.
11-0 R4_0_CT	Runs of Zero, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.

27.6.22 Statistical Check Run Length 4 Limit Register (SCR4L)**Offset**

Register	Offset	Description
SCR4L	30h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 4 Limit Register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 4 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x30) is used as SCR4L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR4C readback register.

Diagram**Fields**

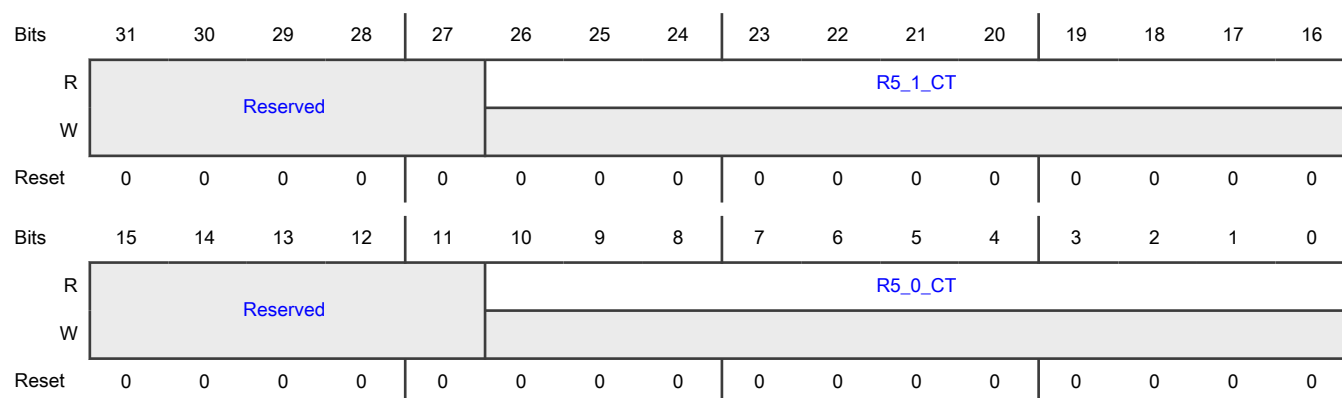
Field	Function
31-28 —	Reserved Always 0.
27-16 RUN4_RNG	Run Length 4 Range. The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX - RUN4_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-12 —	Reserved Always 0.
11-0 RUN4_MAX	Run Length 4 Maximum Limit. Defines the maximum allowable runs of length 4 (for both 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.23 Statistical Check Run Length 5 Count Register (SCR5C)**Offset**

Register	Offset	Description
SCR5C	34h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 5 Counters Register is a read-only register used to read the final Run Length 5 counts after entropy generation. These counters start with the value in SCR5L[RUN5_MAX]. The R5_1_CT decrements each time five consecutive ones are sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time five consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x34) is used as SCR5L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR5C readback register, as described here.

Diagram**Fields**

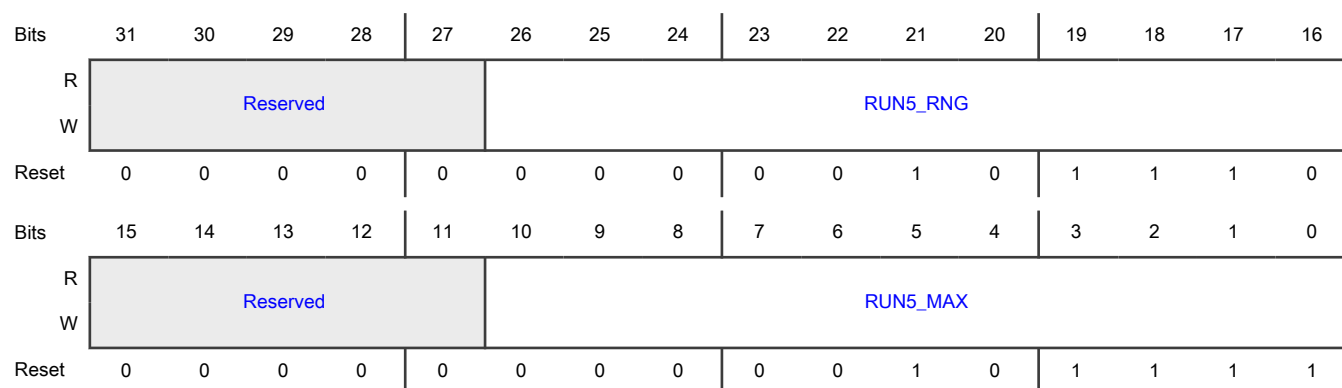
Field	Function
31-27 —	Reserved Always 0.
26-16 R5_1_CT	Runs of One, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.
15-11 —	Reserved Always 0.
10-0 R5_0_CT	Runs of Zero, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.

27.6.24 Statistical Check Run Length 5 Limit Register (SCR5L)**Offset**

Register	Offset	Description
SCR5L	34h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 5 Limit Register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 5 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x34) is used as SCR5L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR5C readback register.

Diagram**Fields**

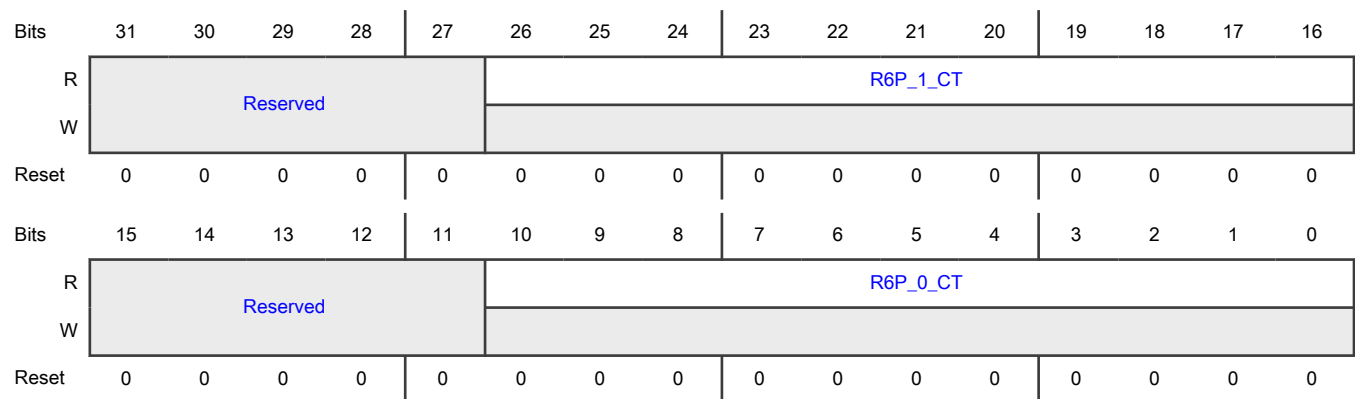
Field	Function
31-27 —	Reserved Always 0.
26-16 RUN5_RNG	Run Length 5 Range. The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX - RUN5_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-11 —	Reserved Always 0.
10-0 RUN5_MAX	Run Length 5 Maximum Limit. Defines the maximum allowable runs of length 5 (for both 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.25 Statistical Check Run Length 6+ Count Register (SCR6PC)**Offset**

Register	Offset	Description
SCR6PC	38h	Accessible at this address when MCTL[PRGM] = 0

Function

The Statistical Check Run Length 6+ Counters Register is a read-only register used to read the final Run Length 6+ counts after entropy generation. These counters start with the value in SCR6PL[RUN6P_MAX]. The R6P_1_CT decrements each time six or more consecutive ones are sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time six or more consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x38) is used as SCR6PL if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register, as described here.

Diagram**Fields**

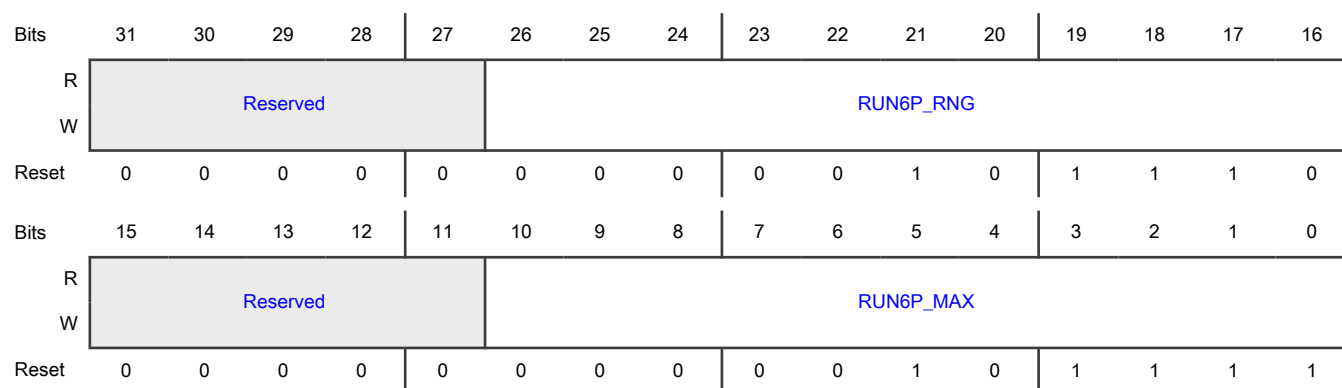
Field	Function
31-27 —	Reserved Always 0.
26-16 R6P_1_CT	Runs of One, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.
15-11 —	Reserved Always 0.
10-0 R6P_0_CT	Runs of Zero, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.

27.6.26 Statistical Check Run Length 6+ Limit Register (SCR6PL)**Offset**

Register	Offset	Description
SCR6PL	38h	Accessible at this address when MCTL[PRGM] = 1

Function

The Statistical Check Run Length 6+ Limit Register defines the allowable maximum and minimum number of runs of length 6 or more detected during entropy generation. To pass the test, the number of runs of length 6 or more (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 6 or more must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x38) is used as SCR6PL only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register.

Diagram**Fields**

Field	Function
31-27 —	Reserved Always 0.
26-16 RUN6P_RNG	Run Length 6+ Range. The number of runs of length 6 or more (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX - RUN6P_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-11 —	Reserved Always 0.
10-0 RUN6P_MAX	Run Length 6+ Maximum Limit. Defines the maximum allowable runs of length 6 or more (for both 0 and 1) detected during entropy generation. The number of runs of length 6 or more detected during entropy generation must be less than RUN6P_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

27.6.27 Status Register (STATUS)**Offset**

Register	Offset
STATUS	3Ch

Function

Various statistical tests are run as a normal part of the TRNG's entropy generation process. The least-significant 16 bits of the STATUS register reflect the result of each of these tests. The status of these bits will be valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the Miscellaneous Control Register.

Note that there is a very small probability that a statistical test will fail even though the TRNG is operating properly. If this happens the TRNG will automatically retry the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is

nonzero, a retry is initiated. But if a statistical check fails when the retry count is zero, an error is generated by the RNG. By default RETRY_CT is initialized to 1, but software can increase the retry count by writing to the RTY_CT field in the SCMISC register.

All 0s will be returned if this register address is read while the RNG is in Program Mode (see PRGM field in MCTL register). If this register is read while the RNG is in Run Mode the value returned will be formatted as follows.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RETRY_CT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFMB	TFP	TFLR	TFSB	TF6PB R1	TF6PB R0	TF5BR 1	TF5BR 0	TF4BR 1	TF4BR 0	TF3BR 1	TF3BR 0	TF2BR 1	TF2BR 0	TF1BR 1	TF1BR 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved Always 0.
19-16 RETRY_CT	RETRY COUNT. This represents the current number of entropy generation retries left before a statistical text failure will cause the RNG to generate an error condition.
15 TFMB	Test Fail, Mono Bit. If TFMB=1, the Mono Bit Test has failed.
14 TFP	Test Fail, Poker. If TFP=1, the Poker Test has failed.
13 TFLR	Test Fail, Long Run. If TFLR=1, the Long Run Test has failed.
12 TFSB	Test Fail, Sparse Bit. If TFSB=1, the Sparse Bit Test has failed.
11 TF6PBR1	Test Fail, 6 Plus Bit Run, Sampling 1s. If TF6PBR1=1, the 6 Plus Bit Run, Sampling 1s Test has failed.
10 TF6PBR0	Test Fail, 6 Plus Bit Run, Sampling 0s. If TF6PBR0=1, the 6 Plus Bit Run, Sampling 0s Test has failed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 TF5BR1	Test Fail, 5-Bit Run, Sampling 1s. If TF5BR1=1, the 5-Bit Run, Sampling 1s Test has failed.
8 TF5BR0	Test Fail, 5-Bit Run, Sampling 0s. If TF5BR0=1, the 5-Bit Run, Sampling 0s Test has failed.
7 TF4BR1	Test Fail, 4-Bit Run, Sampling 1s. If TF4BR1=1, the 4-Bit Run, Sampling 1s Test has failed.
6 TF4BR0	Test Fail, 4-Bit Run, Sampling 0s. If TF4BR0=1, the 4-Bit Run, Sampling 0s Test has failed.
5 TF3BR1	Test Fail, 3-Bit Run, Sampling 1s. If TF3BR1=1, the 3-Bit Run, Sampling 1s Test has failed.
4 TF3BR0	Test Fail, 3-Bit Run, Sampling 0s. If TF3BR0=1, the 3-Bit Run, Sampling 0s Test has failed.
3 TF2BR1	Test Fail, 2-Bit Run, Sampling 1s. If TF2BR1=1, the 2-Bit Run, Sampling 1s Test has failed.
2 TF2BR0	Test Fail, 2-Bit Run, Sampling 0s. If TF2BR0=1, the 2-Bit Run, Sampling 0s Test has failed.
1 TF1BR1	Test Fail, 1-Bit Run, Sampling 1s. If TF1BR1=1, the 1-Bit Run, Sampling 1s Test has failed.
0 TF1BR0	Test Fail, 1-Bit Run, Sampling 0s. If TF1BR0=1, the 1-Bit Run, Sampling 0s Test has failed.

27.6.28 Entropy Read Register (ENT0 - ENT15)

Offset

For a = 0 to 15:

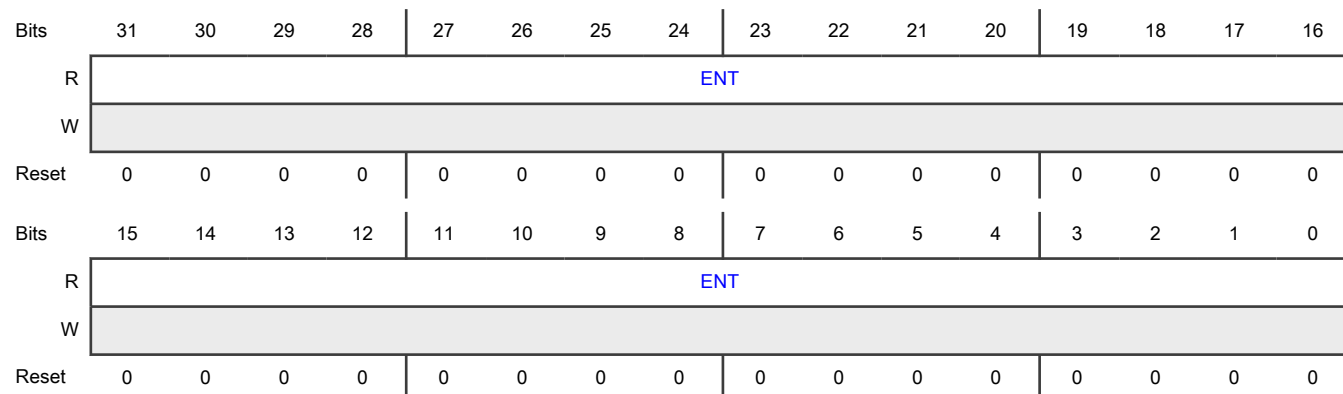
Register	Offset	Description
ENTa	40h + (a × 4h)	Word a

Function

The RNG TRNG can be programmed to generate an entropy value that is readable via the SkyBlue bus. To do this, set the MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the MCTL[ENT_VAL] bit will be set to 1. At this point, ENT0 through ENT15 may be read to retrieve the 512-bit entropy value. Note that once ENT15 is read, the entropy value will be

cleared and a new value will begin generation, so it is important that ENT15 be read last. These registers are readable only when MCTL[PRGM] = 0 (Run Mode) , MCTL[TRNG_ACC] = 1 (TRNG access mode) and MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid ENT15 register value, reading any ENT0 through ENT15 register would return zeros.

Diagram



Fields

Field	Function
31-0 ENT	Entropy Value. Will be non-zero only if MCTL[PRGM] = 0 Run Mode) and MCTL[ENT_VAL] = 1 (Entropy Valid). The most significant bits of the entropy are read from the lowest offset, and the least significant bits are read from the highest offset. Note that reading the highest offset also clears the entire entropy value, and starts a new entropy generation.

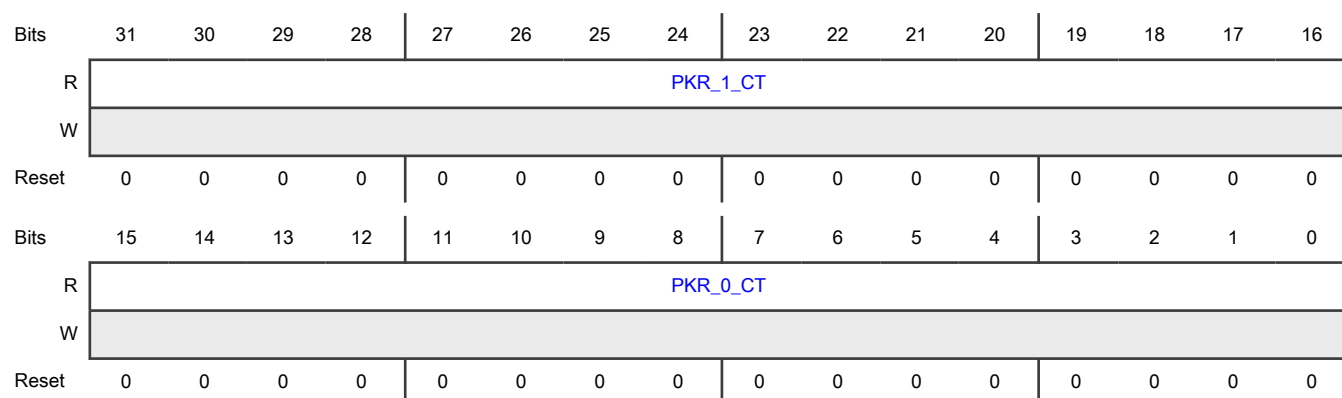
27.6.29 Statistical Check Poker Count 1 and 0 Register (PKRCNT10)

Offset

Register	Offset
PKRCNT10	80h

Function

The Statistical Check Poker Count 1 and 0 Register is a read-only register used to read the final Poker test counts of 1h and 0h patterns. The Poker 0h Count increments each time a nibble of sample data is found to be 0h. The Poker 1h Count increments each time a nibble of sample data is found to be 1h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

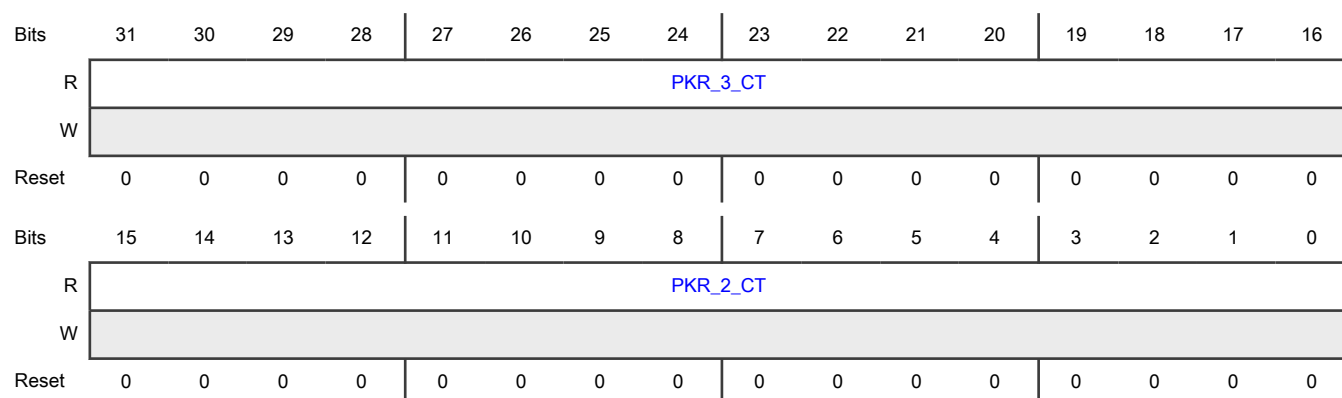
Field	Function
31-16 PKR_1_CT	Poker 1h Count. Total number of nibbles of sample data which were found to be 1h. Requires MCTL[PRGM] = 0.
15-0 PKR_0_CT	Poker 0h Count. Total number of nibbles of sample data which were found to be 0h. Requires MCTL[PRGM] = 0.

27.6.30 Statistical Check Poker Count 3 and 2 Register (PKRCNT32)**Offset**

Register	Offset
PKRCNT32	84h

Function

The Statistical Check Poker Count 3 and 2 Register is a read-only register used to read the final Poker test counts of 3h and 2h patterns. The Poker 2h Count increments each time a nibble of sample data is found to be 2h. The Poker 3h Count increments each time a nibble of sample data is found to be 3h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

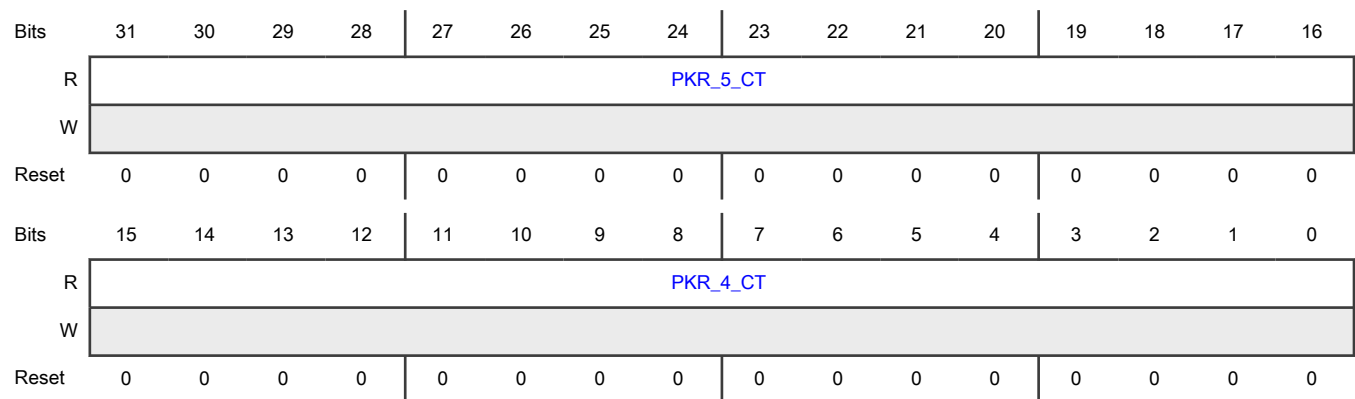
Field	Function
31-16 PKR_3_CT	Poker 3h Count. Total number of nibbles of sample data which were found to be 3h. Requires MCTL[PRGM] = 0.
15-0 PKR_2_CT	Poker 2h Count. Total number of nibbles of sample data which were found to be 2h. Requires MCTL[PRGM] = 0.

27.6.31 Statistical Check Poker Count 5 and 4 Register (PKRCNT54)**Offset**

Register	Offset
PKRCNT54	88h

Function

The Statistical Check Poker Count 5 and 4 Register is a read-only register used to read the final Poker test counts of 5h and 4h patterns. The Poker 4h Count increments each time a nibble of sample data is found to be 4h. The Poker 5h Count increments each time a nibble of sample data is found to be 5h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

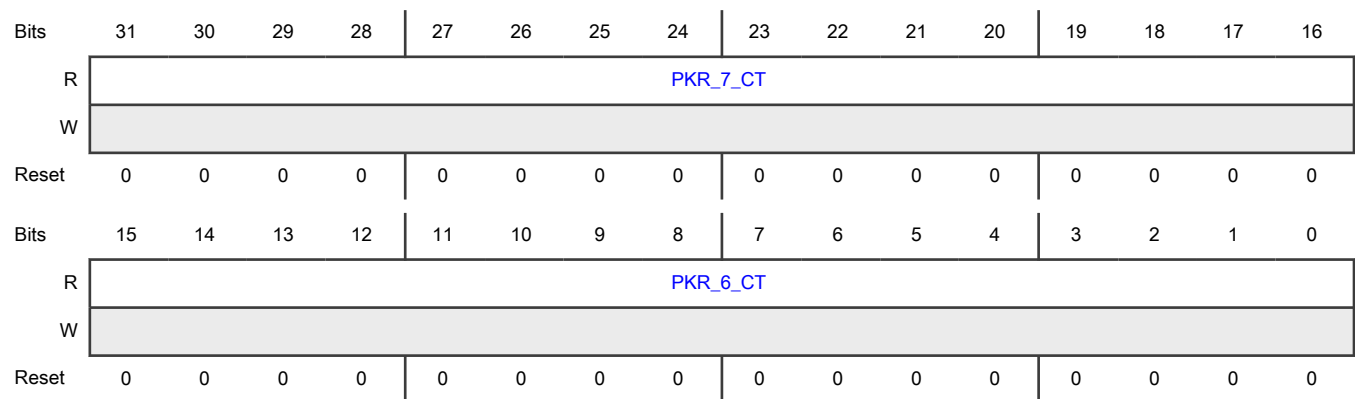
Field	Function
31-16 PKR_5_CT	Poker 5h Count. Total number of nibbles of sample data which were found to be 5h. Requires MCTL[PRGM] = 0.
15-0 PKR_4_CT	Poker 4h Count. Total number of nibbles of sample data which were found to be 4h. Requires MCTL[PRGM] = 0.

27.6.32 Statistical Check Poker Count 7 and 6 Register (PKRCNT76)**Offset**

Register	Offset
PKRCNT76	8Ch

Function

The Statistical Check Poker Count 7 and 6 Register is a read-only register used to read the final Poker test counts of 7h and 6h patterns. The Poker 6h Count increments each time a nibble of sample data is found to be 6h. The Poker 7h Count increments each time a nibble of sample data is found to be 7h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

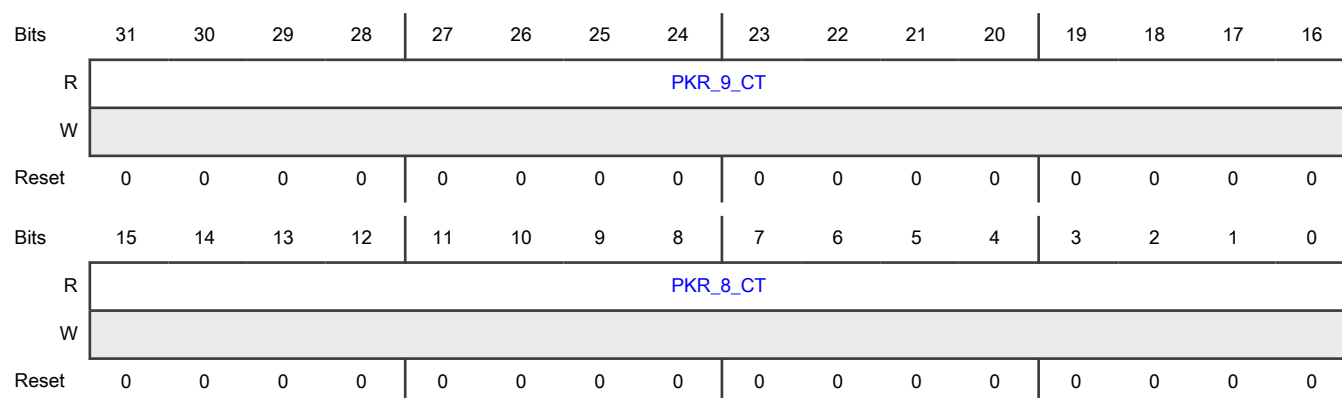
Field	Function
31-16 PKR_7_CT	Poker 7h Count. Total number of nibbles of sample data which were found to be 7h. Requires MCTL[PRGM] = 0.
15-0 PKR_6_CT	Poker 6h Count. Total number of nibbles of sample data which were found to be 6h. Requires MCTL[PRGM] = 0.

27.6.33 Statistical Check Poker Count 9 and 8 Register (PKRCNT98)**Offset**

Register	Offset
PKRCNT98	90h

Function

The Statistical Check Poker Count 9 and 8 Register is a read-only register used to read the final Poker test counts of 9h and 8h patterns. The Poker 8h Count increments each time a nibble of sample data is found to be 8h. The Poker 9h Count increments each time a nibble of sample data is found to be 9h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

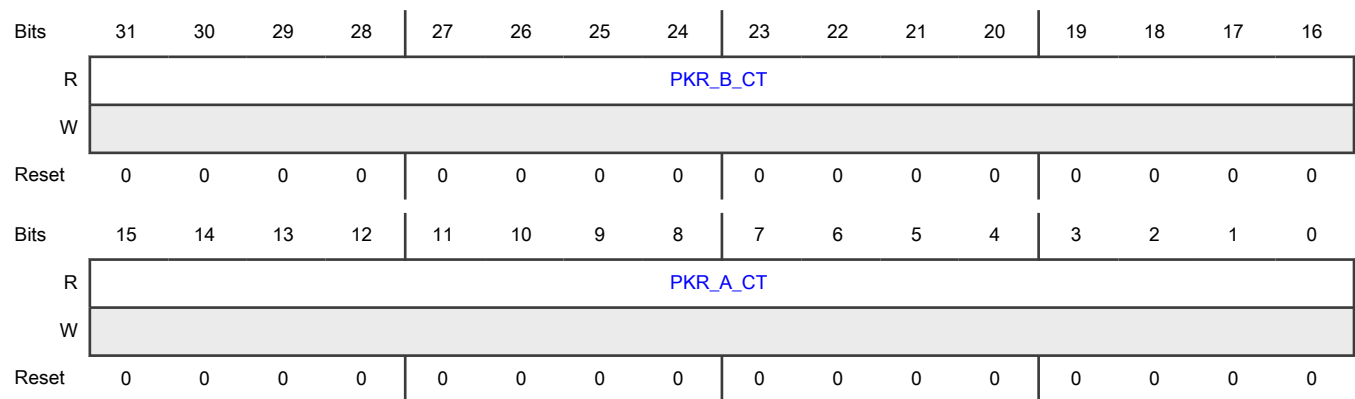
Field	Function
31-16 PKR_9_CT	Poker 9h Count. Total number of nibbles of sample data which were found to be 9h. Requires MCTL[PRGM] = 0.
15-0 PKR_8_CT	Poker 8h Count. Total number of nibbles of sample data which were found to be 8h. Requires MCTL[PRGM] = 0.

27.6.34 Statistical Check Poker Count B and A Register (PKRCNTBA)**Offset**

Register	Offset
PKRCNTBA	94h

Function

The Statistical Check Poker Count B and A Register is a read-only register used to read the final Poker test counts of Bh and Ah patterns. The Poker Ah Count increments each time a nibble of sample data is found to be Ah. The Poker Bh Count increments each time a nibble of sample data is found to be Bh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

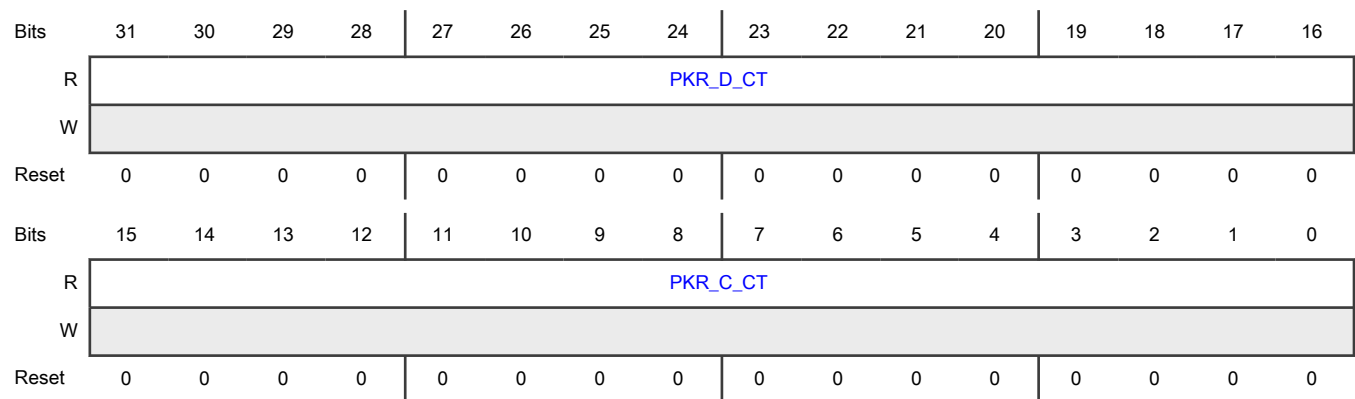
Field	Function
31-16 PKR_B_CT	Poker Bh Count. Total number of nibbles of sample data which were found to be Bh. Requires MCTL[PRGM] = 0.
15-0 PKR_A_CT	Poker Ah Count. Total number of nibbles of sample data which were found to be Ah. Requires MCTL[PRGM] = 0.

27.6.35 Statistical Check Poker Count D and C Register (PKRCNTDC)**Offset**

Register	Offset
PKRCNTDC	98h

Function

The Statistical Check Poker Count D and C Register is a read-only register used to read the final Poker test counts of Dh and Ch patterns. The Poker Ch Count increments each time a nibble of sample data is found to be Ch. The Poker Dh Count increments each time a nibble of sample data is found to be Dh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

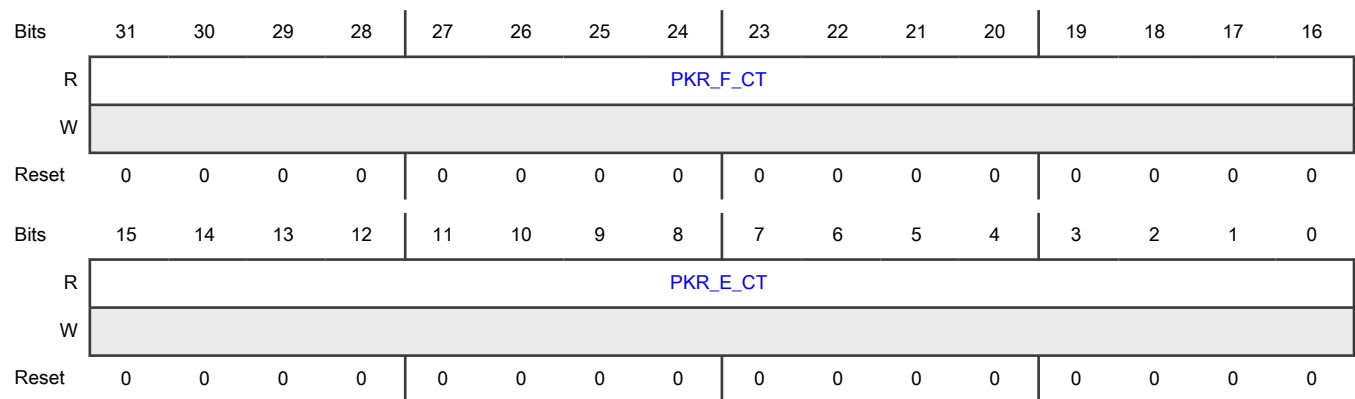
Field	Function
31-16 PKR_D_CT	Poker Dh Count. Total number of nibbles of sample data which were found to be Dh. Requires MCTL[PRGM] = 0.
15-0 PKR_C_CT	Poker Ch Count. Total number of nibbles of sample data which were found to be Ch. Requires MCTL[PRGM] = 0.

27.6.36 Statistical Check Poker Count F and E Register (PKRCNTFE)**Offset**

Register	Offset
PKRCNTFE	9Ch

Function

The Statistical Check Poker Count F and E Register is a read-only register used to read the final Poker test counts of Fh and Eh patterns. The Poker Eh Count increments each time a nibble of sample data is found to be Eh. The Poker Fh Count increments each time a nibble of sample data is found to be Fh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

Diagram**Fields**

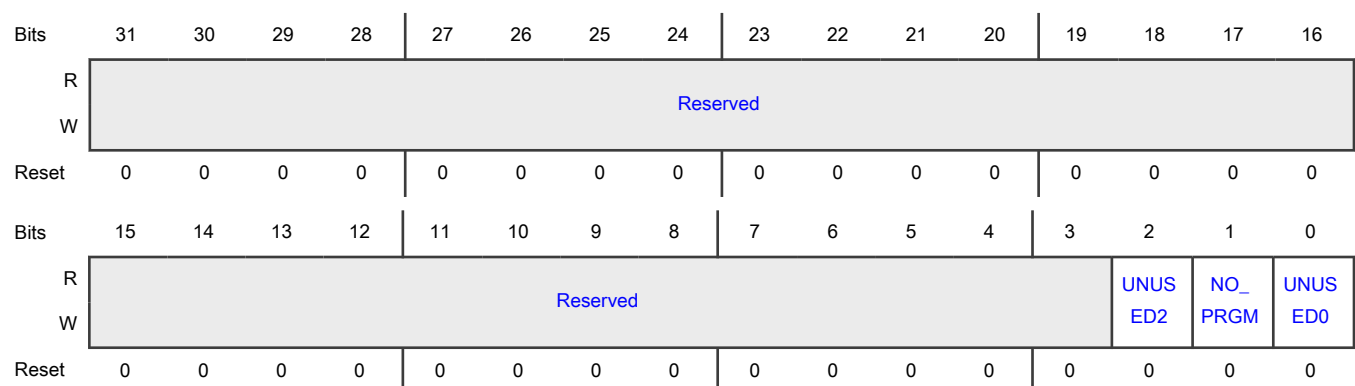
Field	Function
31-16 PKR_F_CT	Poker Fh Count. Total number of nibbles of sample data which were found to be Fh. Requires MCTL[PRGM] = 0.
15-0 PKR_E_CT	Poker Eh Count. Total number of nibbles of sample data which were found to be Eh. Requires MCTL[PRGM] = 0.

27.6.37 Security Configuration Register (SEC_CFG)**Offset**

Register	Offset
SEC_CFG	A0h

Function

The Security Configuration Register is a read/write register used to control the test mode, programmability and state modes of the TRNG. Clears on asynchronous system hard reset or POR.

Diagram

Fields

Field	Function
31-3 —	Reserved
2 UNUSED2	This bit is unused. Ignore.
1 NO_PRGM	If set, the TRNG registers cannot be programmed regardless of the TRNG access mode in the TRNG Miscellaneous Control Register. 0b - Programability of registers controlled only by the Miscellaneous Control Register's access mode bit. 1b - Overrides Miscellaneous Control Register access mode and prevents TRNG register programming.
0 UNUSED0	This bit is unused. Ignore.

27.6.38 Interrupt Control Register (INT_CTRL)

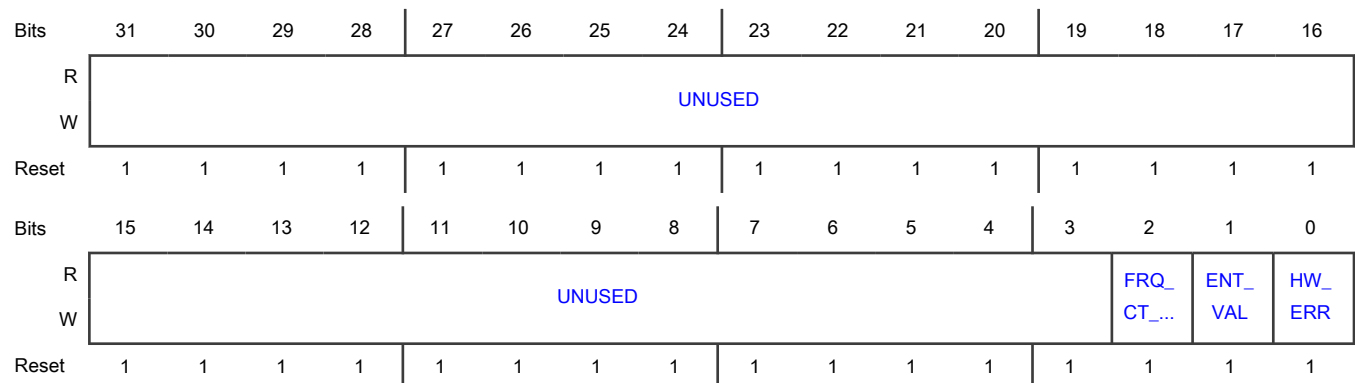
Offset

Register	Offset
INT_CTRL	A4h

Function

The Interrupt Control Register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG. See [INT_STATUS](#) register description. Each interrupt can be cleared by de-asserting the corresponding bit in the [INT_CTRL](#) register. Only a new interrupt will reassert the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the [MCTL](#) register.

Diagram



Fields

Field	Function
31-3 UNUSED	Reserved but writeable.
2 FRQ_CT_FAIL	Same behavior as bit 0 of this register. 0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
1 ENT_VAL	Same behavior as bit 0 of this register. 0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
0 HW_ERR	Bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted. 0b - Corresponding bit of INT_STATUS register cleared. 1b - Corresponding bit of INT_STATUS register active.

27.6.39 Mask Register (INT_MASK)

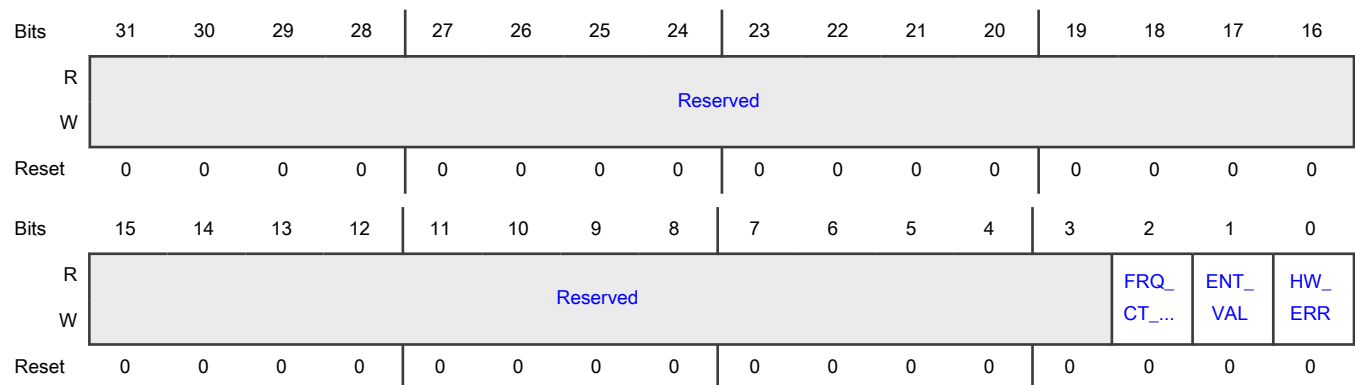
Offset

Register	Offset
INT_MASK	A8h

Function

The Interrupt Mask Register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG. See [INT_STATUS](#) register description. Each interrupt can be masked/disabled by de-asserting the corresponding bit in the [INT_MASK](#) register. Only setting this bit high will re-enable the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the [MCTL register](#).

Diagram



Fields

Field	Function
31-3 —	Reserved
2 FRQ_CT_FAIL	Same behavior as bit 0 of this register. 0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
1 ENT_VAL	Same behavior as bit 0 of this register. 0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
0 HW_ERR	Bit position that can be cleared or set to enable the corresponding bit of INT_STATUS to show interrupt status. 0b - Corresponding interrupt of INT_STATUS is masked. 1b - Corresponding bit of INT_STATUS is active.

27.6.40 Interrupt Status Register (INT_STATUS)

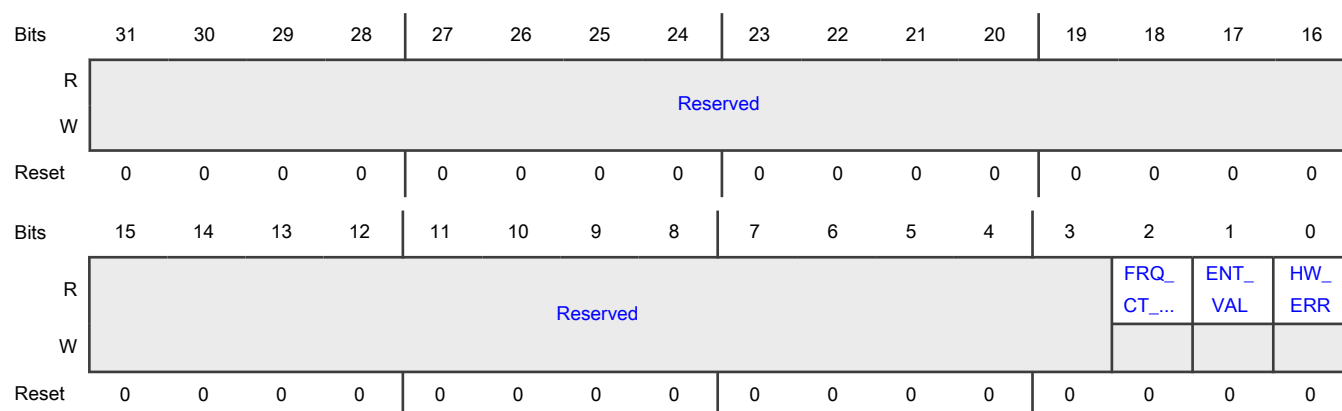
Offset

Register	Offset
INT_STATUS	ACh

Function

The Interrupt Status Register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The TRNG interrupt signal alerts that TRNG has either generated a Frequency Count Fail, Entropy Valid or Error Interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the INT_STATUS register. Each interrupt can be temporarily cleared by de-asserting the corresponding bit in the [INT_CTRL](#) register. To mask the interrupts, clear the corresponding bits in the [INT_MASK](#) register. The description of each of the 3 interrupts is defined in the Block Guide under the MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 FRQ_CT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. 0b - No hardware nor self test frequency errors. 1b - The frequency counter has detected a failure.
1 ENT_VAL	Read only: Entropy Valid. Will assert after an entropy value is generated and only if TRNG_ACC bit is set. Will be cleared when ENT15 is read. (ENT0 through ENT14 should be read before reading ENT15). 0b - Busy generation entropy. Any value read is invalid. 1b - TRNG can be stopped and entropy is valid if read.
0 HW_ERR	Read: Error status. 1 = error detected. 0 = no error. Any HW error in the TRNG will trigger this interrupt. 0b - no error 1b - error detected.

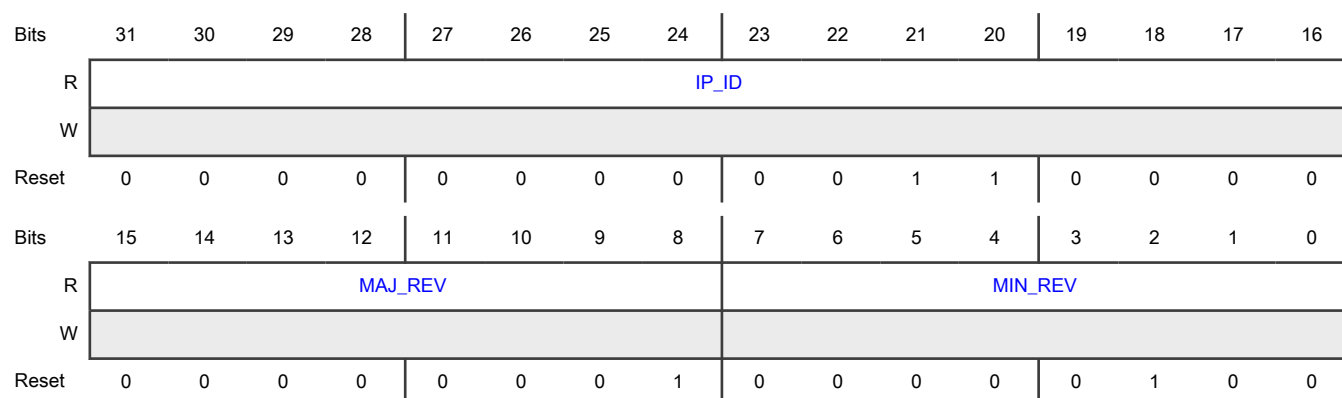
27.6.41 Version ID Register (MS) (VID1)

Offset

Register	Offset
VID1	F0h

Function

The Version ID Register (VID1) is a MSW read only register used to identify the version of the TRNG in use. This register as well as [VID2](#) should both be read to verify the expected version.

Diagram**Fields**

Field	Function
31-16 IP_ID	Shows the IP ID. 0000_0000_0011_0000b - ID for TRNG.
15-8 MAJ_REV	Shows the IP's Major revision of the TRNG. 0000_0001b - Major revision number for TRNG.
7-0 MIN_REV	Shows the IP's Minor revision of the TRNG. 0000_0100b - Minor revision number for TRNG.

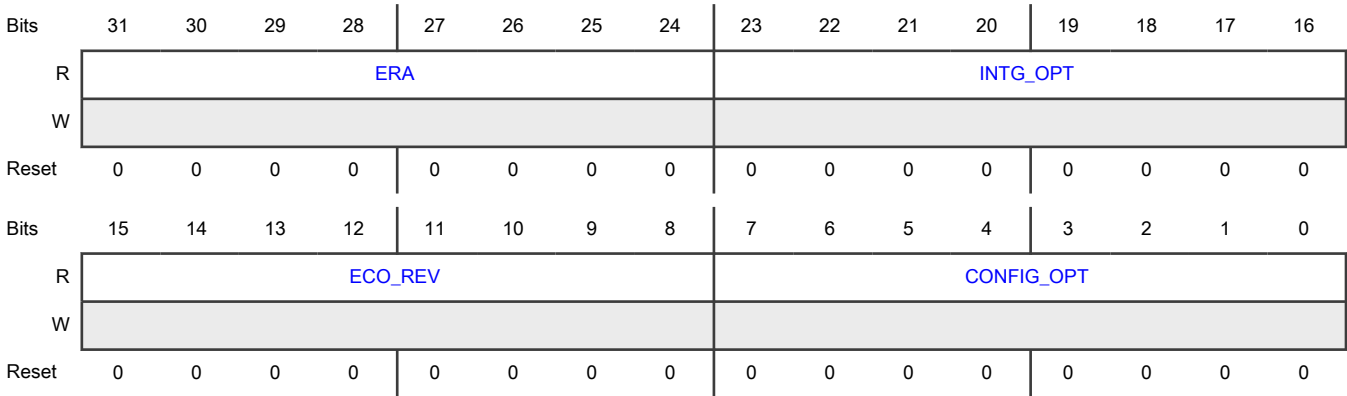
27.6.42 Version ID Register (LS) (VID2)**Offset**

Register	Offset
VID2	F4h

Function

The Version ID Register LSB is a read only register used to identify the architecture of the TRNG in use. This register as well as VID1 should both be read to verify the expected version.

Diagram



Fields

Field	Function
31-24 ERA	Shows the compile options for the TRNG. 0000_0000b - COMPILE_OPT for TRNG.
23-16 INTG_OPT	Shows the integration options for the TRNG. Used as REV_NUM and unique for each SoC type. NOTE (For 2019 or newer SoCs). 0000_0000b - INTG_OPT for TRNG.
15-8 ECO_REV	Shows the IP's ECO revision of the TRNG. 0000_0000b - TRNG_ECO_REV for TRNG.
7-0 CONFIG_OPT	Shows the IP's Configuration options for the TRNG. 0000_0000b - TRNG_CONFIG_OPT for TRNG.

Chapter 28

Real Time Clock (RTC)

28.1 Chip-specific RTC information

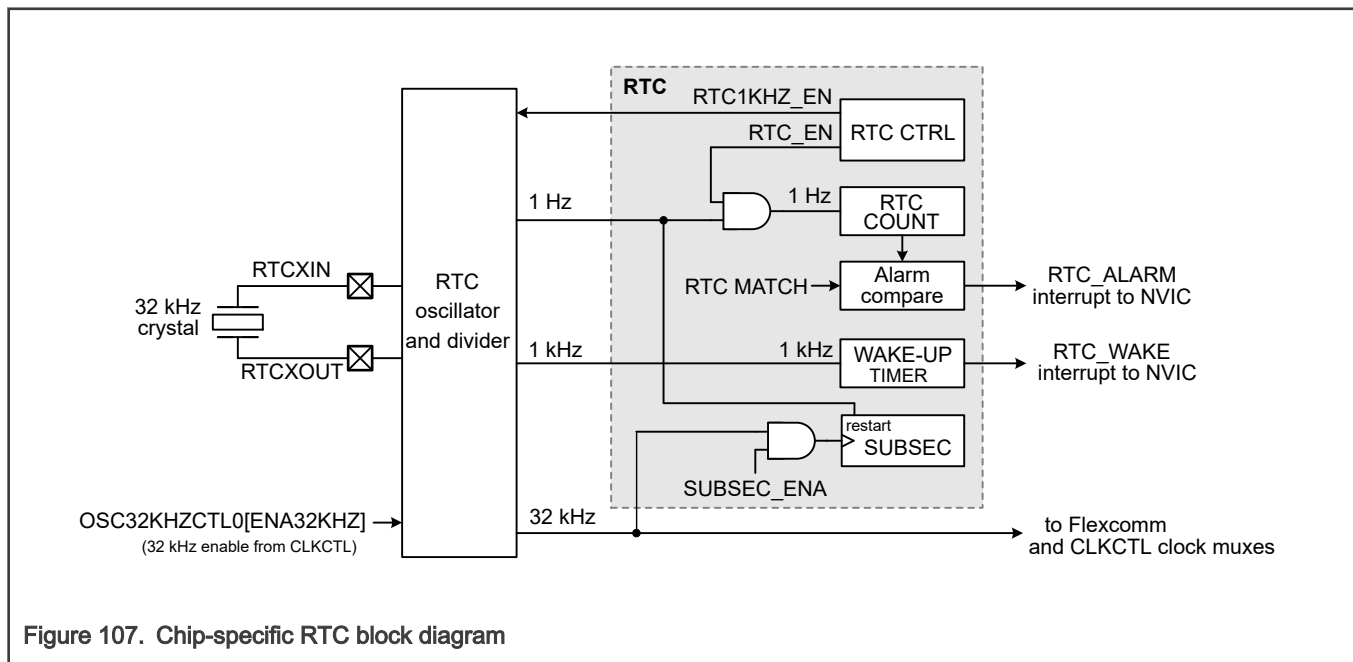
Table 388. Reference links to related information

Topic	Related module	Reference
Full description	RTC	RTC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

This RTC includes these features:

- A 15-bit, 32 kHz sub-second counter
- Eight 32-bit general purpose registers

Block diagram



Chip-specific RTC initialization

- Enable the clock to the RTC register interface and peripheral clock with the AHBCLKCTRL0 register in CLKCTL.
- Enable the RTC software reset in the RTC CTRL register. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, RTC is not initialized by other system resets.
- Enable the interrupts to the NVIC for the RTC_WAKE and RTC_ALARM functions. See the NVIC chapter.
- Enable the RTC interrupts for waking up the device from Deep-sleep mode by enabling the interrupts in the STARTEN1 register of SYSCTL and also in the NVIC.

- Enable the RTC interrupts for waking up from Deep Power-down mode using the RTC CTRL register.

If enabled, the RTC and RTC oscillator continue running in all low power modes as long as power is supplied to the device. Therefore, the 32 kHz output is always available to be enabled for SYSCCTL clock generation. Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT and 32KHZ_CLKOUT pins. The 1 Hz output is enabled in the RTC CTRL[RTC_EN] bit. Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL[RTC1KHZ_EN] bit.

If the 32 kHz output of the RTC is used by another part of the system, enable it via the RTC CTRL[RTC_EN].

28.1.1 Module instances

This device has one instance of the RTC module, RTC.

28.2 Overview

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, the RTC counts continuously at a 1 Hz clock rate as long as the device is powered and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the device is in one of the low power modes (such as Deep-sleep or Deep Power-down) an RTC alarm interrupt can also wake up the device to exit the power mode and begin normal operation.

28.2.1 Block diagram

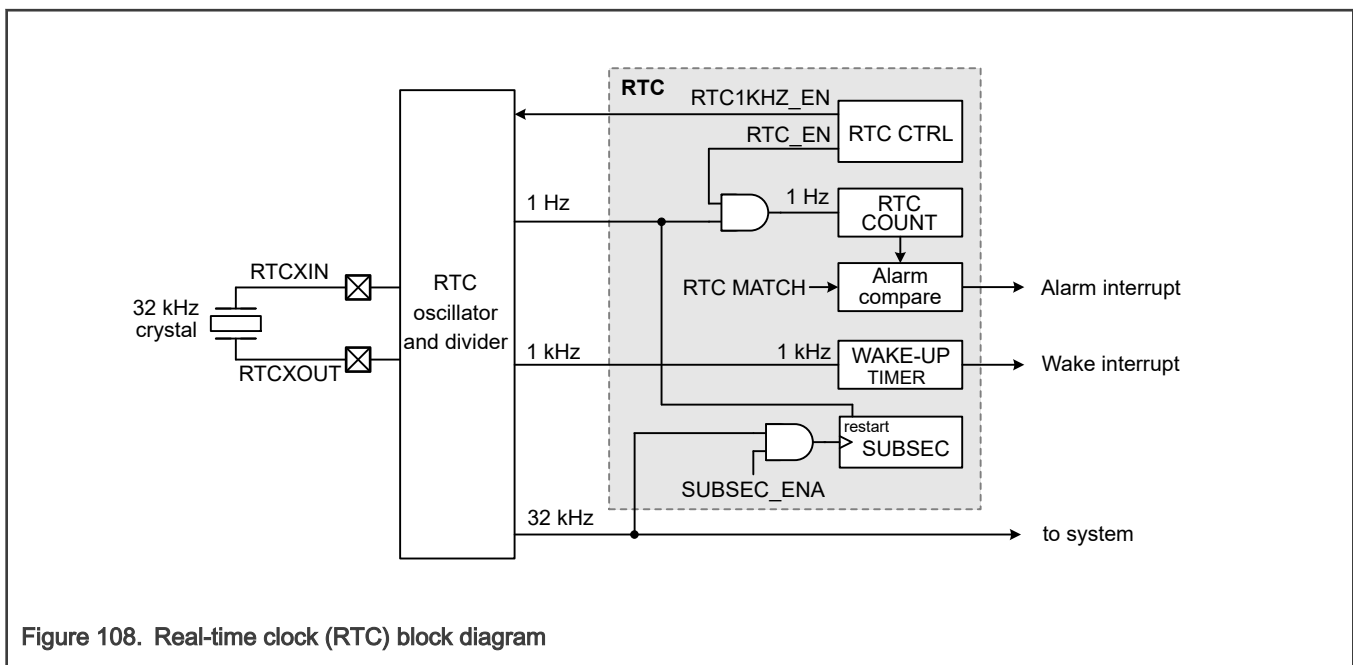


Figure 108. Real-time clock (RTC) block diagram

28.2.2 Features

- The RTC and the RTC independent oscillator operate directly from the device power pins, not using the on-chip regulator. The RTC oscillator has the following clock outputs:
 - 32 kHz clock, selectable for system use
 - 1 Hz clock for RTC timing
 - 1 kHz clock for high-resolution RTC timing
- 32-bit, 1 Hz RTC counter and associated MATCH register for alarm generation

- 15-bit, 32 kHz sub-second counter (May not be available on all RTC instances)
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution with a more than 1-minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests. Either time-out can wake up the device from any of the low power modes, including Deep Power-down mode.
- Eight 32-bit general purpose registers which can retain backup information through Deep Power-down mode or in the event of a power failure provided battery backup is present. (May not be available on all RTC instances).

28.3 Functional description

28.3.1 RTC timers

The RTC contains these timers:

- The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the device from any low power mode if enabled.
- The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot countdown timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the device from any low power mode, if enabled. This timer is intended to be used for timed wake-ups from Deep-sleep or Deep Power-down modes.

The high-resolution wake-up timer can be disabled to conserve power, if not used.

- The sub-second counter. When enabled, counts cycles of the 32 kHz oscillator, and is synchronized to the main RTC timer.

28.3.1.1 High-resolution/wake-up timer

The time interval required for many applications, including waking the device up from a low-power mode, often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-resolution/Wake-up Timer (WAKE) is a 16-bit countdown timer, which is clocked at a 1 kHz rate when enabled. Writing any non-zero value to this timer automatically enables the counter and launches a countdown sequence. When the counter is used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

High-Resolution/Wake-up Timer (WAKE) sequence:

- Load the starting count value. WAKE turns on, and begins the countdown from the pre-loaded value down to zero.
- When the value of WAKE reaches zero, that generates an interrupt and/or a wake-up command.
- WAKE then turns itself off until re-launched by a subsequent software write.

28.3.1.2 (Optional) Sub-second counter

Some instances of the Real Time Clock module include a 15-bit sub-second up-counter which is clocked at a 32-kHz rate. The 32-KHz clock must be enabled prior to using this feature.

The state of this counter may be read through the bus and combined with the main, one-second RTC count for a more precise time reading. The sub-second counter does not contribute to alarm, interrupt, or wakeup generation.

The sub-second counter is disabled whenever the RTC is in reset or whenever the main RTC 1 Hz counter is disabled. The sub-second counter must be independently enabled after the main counter is enabled.

To re-enable the sub-second counter:

- Set the main counter with the RTC enable bit equal to 1 (CTRL[RTC_EN] = 1)

- Set the 32-kHz Sub-second Counter Enable bit equal to 1 (CTRL[RTC_SUBSEC_ENA] = 1)

Once enabled, the counter waits until the start of the next one-second interval and then begins incrementing at a 32 KHz rate. The counter rolls-over to zero and resumes counting at the start of each one-second interval as long as the counter is enabled.

Unlike the rest of the RTC module, the sub-second counter does NOT reside in the “always-on” voltage domain so it is not available and does not retain its state during Deep Power-down mode. This counter must be re-enabled upon exiting Deep Power-down or following a loss of main chip power.

28.3.2 General purpose backup registers

Some instances of the RTC module include eight 32-bit general purpose registers. These registers may be used to store critical data through Deep Power-down mode or in the event of a main power failure provided the battery backup supply is present.

28.3.3 RTC power domain

With the exception of the sub-second counter, the RTC module and the 1Hz/1kHz oscillator that drives it, reside in the battery-backup “always-on” voltage domain. As a result, the RTC continues operating in Deep Power-down mode when power is removed from the rest of the chip. The sub-second counter does not function in Deep Power-down mode.

If a backup battery is employed the RTC continues to operate during power failure, until the backup battery runs out.

28.4 Signals

The RTC pins table gives a summary of the signals related to the RTC.

Table 389. RTC pins

Pin	Type	Description
RTCXIN	Input	RTC oscillator input. Can be input to external square wave when oscillator is bypassed.
RTCXOUT	Output	RTC oscillator output

28.5 Initialization

To Initialize the RTC:

- Enable the clock to the RTC register interface and peripheral clock.
- Enable the RTC CTRL register for RTC software reset.
- Enable the wake and alarm interrupts in the device's interrupt controller.
- Enable the RTC interrupts for waking up the device from Deep-sleep mode.
- Enable the RTC interrupts for waking up the device from Deep Power-down mode.

See Chip-specific RTC information.

28.6 Memory map and register definition

This section includes the RTC module memory map and detailed descriptions of all registers.

Reset values pertain to the initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

28.6.1 RTC register descriptions

28.6.1.1 RTC memory map

Reset values apply to the initial power-up of the device or the RTC software reset (except where noted). This block is not initialized by any other system reset.

RTC base address: 4003_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	RTC Control (CTRL)	32	RW	See section
4h	RTC Match (MATCH)	32	RW	FFFF_FFFFh
8h	RTC Counter (COUNT)	32	RW	0000_0000h
Ch	High-resolution/Wake-up Timer Control (WAKE)	32	RW	See section
10h	RTC Sub-second Counter (SUBSEC)	32	R	0000_0000h
40h - 5Ch	General Purpose (GPREG0 - GPREG7)	32	RW	0000_0000h

28.6.1.2 RTC Control (CTRL)

Offset

Register	Offset
CTRL	0h

Function

CTRL selects the clock that the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts for wake up of the device from Deep Power-down. To configure wake up of the chip from Deep-sleep mode, enable the RTC interrupts according to system configuration instructions. See the chip-specific RTC information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RTC_OSC_loadcap				Reserved											
W																
Reset	0	0	0	1	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				RTC_S UB...				Reserved				RTC_ OSC...			
W																
Reset	u	u	u	u	u	0	0	1	0	0	0	0	0	0	1	1

Fields

Field	Function																		
31-28 RTC_OSC_loadcap	Capacitive Load Selection RTC_OSC_loadcap allows programmable internal capacitance values in the range of 2 pF through 30 pF in 2 pF steps, eliminating the need for external capacitors on the RTCXIN and RTCXOUT pins. Choose half the required capacitance expressed as a binary value. <table> <tr> <th>Field value</th><th>Required cap load</th></tr> <tr> <td>0000</td><td>0 pF</td></tr> <tr> <td>0001</td><td>2 pF</td></tr> <tr> <td>0010</td><td>4 pF</td></tr> <tr> <td>0011</td><td>6 pF</td></tr> <tr> <td>0100</td><td>8 pF</td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td>1110</td><td>28 pF</td></tr> <tr> <td>1111</td><td>30 pF</td></tr> </table>	Field value	Required cap load	0000	0 pF	0001	2 pF	0010	4 pF	0011	6 pF	0100	8 pF	...	...	1110	28 pF	1111	30 pF
Field value	Required cap load																		
0000	0 pF																		
0001	2 pF																		
0010	4 pF																		
0011	6 pF																		
0100	8 pF																		
...	...																		
1110	28 pF																		
1111	30 pF																		
27-11 —	Reserved Read value is undefined, only zero should be written.																		
10 RTC_SUBSEC_ENA	32-KHz Sub-second Counter Enable NOTE RTC_SUBSEC_ENA can only be set after the RTC_EN has been set by a previous write operation. NOTE The RTC sub-second counter must be re-enabled whenever the chip exits Deep Power-down mode. 0b - Disable. The sub-second counter (if implemented) is disabled. RTC_SUBSEC_ENA is cleared by a system-level POR or BOD reset or by an RTC_EN reset. On modules not equipped with a sub-second counter, RTC_SUBSEC_ENA will always read as 0. 1b - Enable. The 32 KHz sub-second counter is enabled (if implemented). Counting commences on the start of the first one-second interval after RTC_SUBSEC_ENA is set.																		
9 —	Reserved																		
8 RTC_OSC_PD	The RTC Oscillator Enable 0b - Enable. The RTC oscillator is enabled. RTC_OSC_PD must be cleared for the RTC module to function.																		

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Shut Off. The RTC operation is disabled. The RTC oscillator is shut-off to limit power consumption.
7 RTC_EN	RTC enable RTC_EN must be set to initiate operation of the RTC. 0b - Disable. The RTC 1 Hz and 1 kHz clocks are shut down and the RTC operation is disabled. RTC_EN should be 0 when writing to load a value in the RTC counter (COUNT) register . 1b - Enable. The 1 Hz RTC clock is running and RTC operation is enabled. The first clock to the RTC counter occurs 1 s after RTC_EN is set. To also enable the high-resolution, 1 kHz clock, set CTRL[RTC1KHZ_EN] = 1.
6 RTC1KHZ_EN	RTC 1 kHz Clock Enable Set RTC1KHZ_EN to 0 to conserve power if the 1 kHz timer is not used. RTC1KHZ_EN has no effect when the RTC is disabled (CTRL[RTC_EN] = 0). 0b - Disable. A match on the 1 kHz RTC timer does not bring the device out of Deep Power-down mode. 1b - Enable. The 1 kHz RTC timer is enabled.
5 WAKEDPD_EN	RTC 1 kHz Timer Wake-up Enable for Deep Power-down 0b - Disable. A match on the 1 kHz RTC timer does not bring the device out of Deep Power-down mode. 1b - Enable. A match on the 1 kHz RTC timer brings the device out of Deep Power-down mode.
4 ALARM1HZ_EN	RTC 1 Hz Timer Alarm Enable for Deep Power-down 0b - Disable. A match on the 1 Hz RTC timer does not bring the device out of Deep Power-down mode. 1b - Enable. A match on the 1 Hz RTC timer brings the device out of Deep Power-down mode.
3 WAKE1KHZ	RTC 1 kHz Timer Wake-up Flag Status 0b - Run. The RTC 1 kHz timer is running. Writing a 0 has no effect. 1b - Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request RTC-WAKE which can also wake up the device from any low power mode. Write 1 to clear.
2 ALARM1HZ	RTC 1 Hz Timer Alarm Flag Status ALARM1HZ shows the results of a compare between the values of the RTC 1 Hz Timer (read from COUNT[VAL], which reflects the current value of the main, 1 Hz RTC timer) and the match value (from COUNT[VAL]). 0b - No match. No match condition has occurred on the 1 Hz RTC timer. Writing a 0 has no effect. 1b - Match. The value in MATCH[MATVAL] equals the value in COUNT[VAL]. This flag generates an RTC alarm interrupt/wake-up request, RTC_ALARM, which can also wake up the device from any low power mode. Write 1 to clear.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1	Reserved
—	Read value is undefined, only zero should be written.
0 SWRESET	Software Reset Control SWRESET must be cleared before writing to any register in the RTC, including writes to any other fields within this register. Do not write to this register at the same time that the IN_RESET is being cleared. 0b - Not in reset. The RTC is not held in reset. Clear SWRESET before configuring or initiating any operation of the RTC. 1b - In reset. The RTC is held in reset. All registers in the RTC change to their reset value except OFD.

28.6.1.3 RTC Match (MATCH)

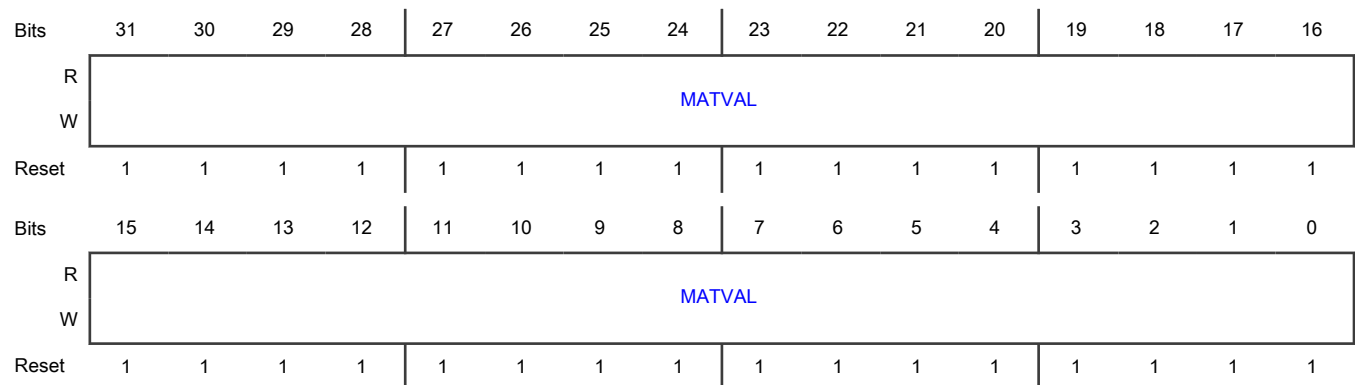
Offset

Register	Offset
MATCH	4h

Function

Contains the value that the 1 Hz RTC timer will be compared. If there is a match, then the alarm flag is set (CTRL[RTC_ALARM1HZ] = 1) and generates an alarm interrupt/wake-up request.

Diagram



Fields

Field	Function
31-0 MATVAL	Match Value

28.6.1.4 RTC Counter (COUNT)

Offset

Register	Offset
COUNT	8h

Function

The RTC counter counts up continuously at a 1 Hz rate once the RTC Software Reset is removed (CTRL[SWRESET] = 0).

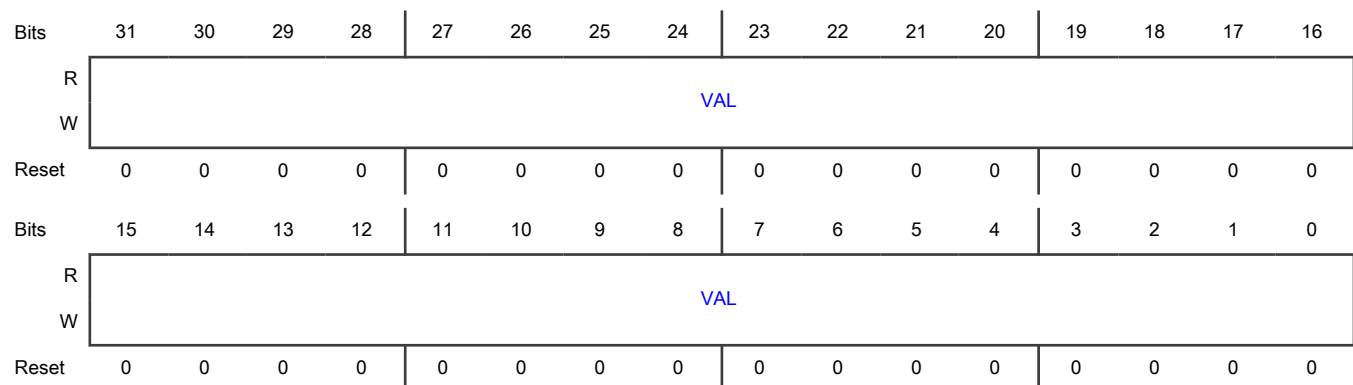
NOTE

No synchronization is provided to prevent a read of VAL during a count transition. The suggested method to read VAL is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result.

NOTE

Only write to this register when CTRL[RTC_EN] = 0. The counter increments one second after the CTRL[RTC_EN] = 1.

Diagram



Fields

Field	Function
31-0	Value
VAL	A read to VAL reflects the current value of the main, 1 Hz RTC timer. A write to VAL loads a new initial value into the timer.

28.6.1.5 High-resolution/Wake-up Timer Control (WAKE)

Offset

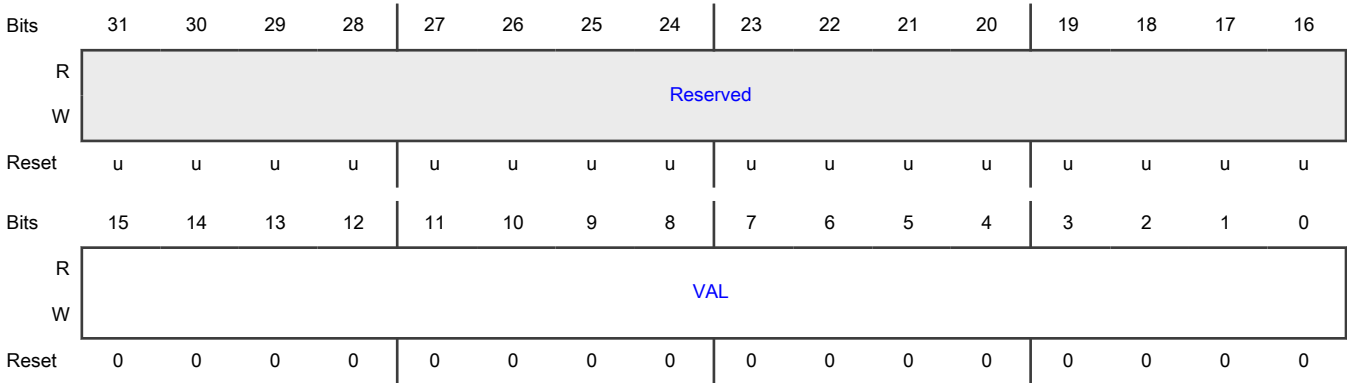
Register	Offset
WAKE	Ch

Function

Use the WAKE register to load a count value for the wake-up timer.

NOTE
Do not write to the WAKE register while counting is in progress.

Diagram



Fields

Field	Function
31-16	Reserved
—	Read value is undefined, only zero should be written.
15-0	Value
VAL	A read reflects the current value of High-resolution/Wake-up Timer (VAL). A write pre-loads a start count value into VAL and initializes a countdown sequence.

28.6.1.6 RTC Sub-second Counter (SUBSEC)

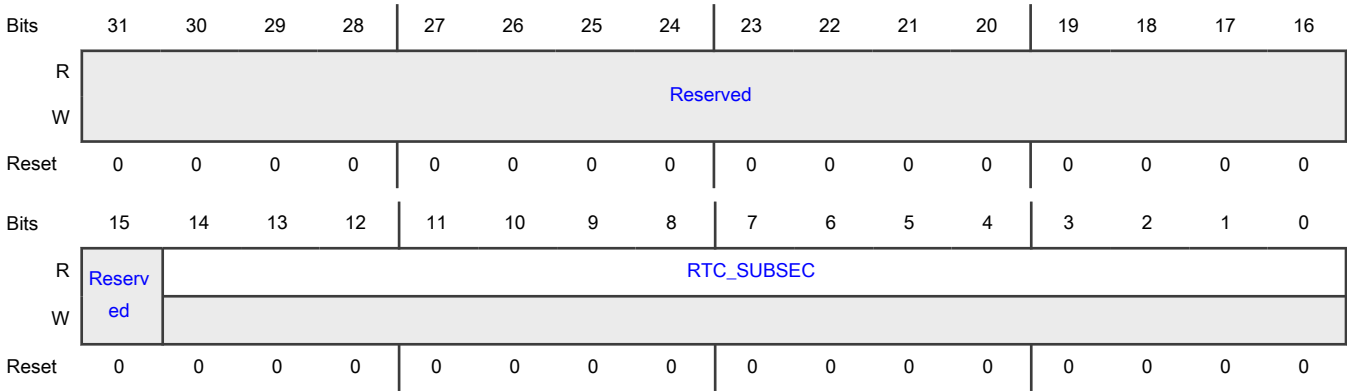
Offset

Register	Offset
SUBSEC	10h

Function

The SUBSEC register must be re-enabled after exiting Deep Power-down mode or after the main RTC module has been disabled and re-enabled.

Diagram



Fields

Field	Function
31-15	Reserved
—	Software should not write ‘1’s to unused bits.
14-0 RTC_SUBSEC	RTC Sub-second Counter A read reflects the current value of the 32 kHz sub-second counter. This counter is cleared whenever the Sub-second Enable bit is equal to zero (RTC_CONTROL[SUBSEC_ENA] = 0). Up-counting at a 32 kHz rate begins at the start of the next one-second interval after the Sub-second Enable bit is set (RTC_CONTROL[SUBSEC_ENA] = 1). On modules not equipped with a sub-second counter, this register reads back as all zeroes.

28.6.1.7 General Purpose (GPREG0 - GPREG7)

Offset

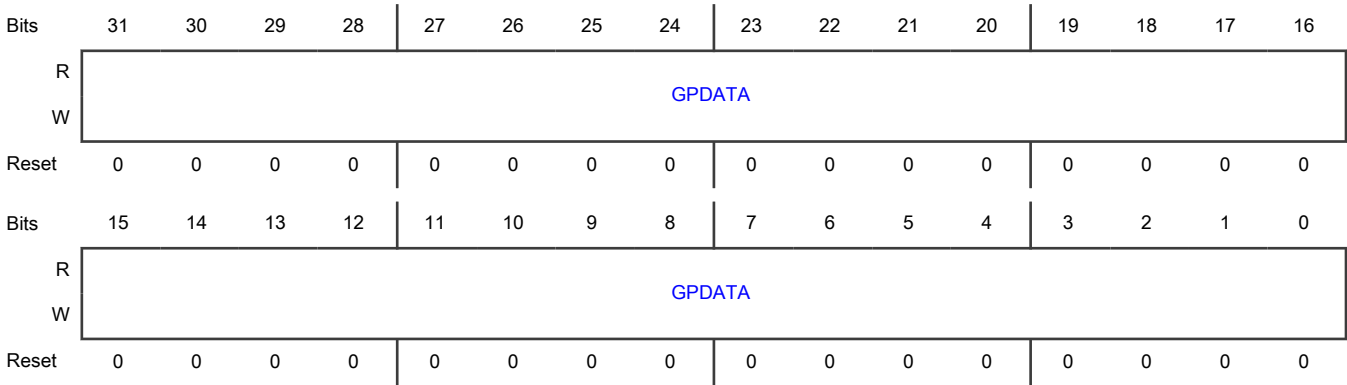
For a = 0 to 7:

Register	Offset
GPREGa	40h + (a × 4h)

Function

These registers retain contents even during Deep Power-down mode as long as device power is maintained. They can be used to preserve application data or a configuration that will always be available.

Diagram



Fields

Field	Function
31-0 GPDATA	General Purpose Data Data retained during Deep Power-down mode or loss of main power.

Chapter 29

CPU system tick timer (SYSTICK)

29.1 Overview

This section provides an introduction to the SYSTICK module.

29.1.1 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Selection of clocks, see Clock Control (CLKCTL).
- The Cortex-M33 includes a secure and non-secure systick timer.
 - The secure and non-secure systick timer each has its own registers. Which timer is accessed depends on the current security state
 - The secure and non-secure systick timer each has its own calibration register, see CLKCTL register section.

29.1.2 Block Diagram

SysTick timer block diagram is shown below . The secure and non-secure SysTick timers are functionally identical.

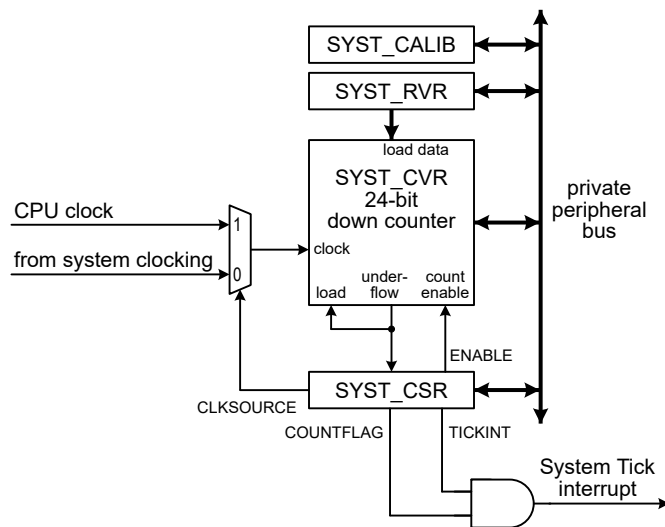


Figure 109.

29.2 Functional description

The SysTick timers are an integral part of the Cortex-M33. The feature is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timers are part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-based devices. The timers can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.

- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Clocking Diagram) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

29.3 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

$$\text{SYST\_RVR} = (\text{system clock frequency} \times 10 \text{ ms}) - 1 = (72 \text{ MHz} \times 10 \text{ ms}) - 1 = 720\,000 - 1 = 719\,999 = 0x000A\,FC7F$$

System tick timer clock = 24 MHz

Program the SYST_CSR register with the value 0x3 which selects the clock from the system tick clock divider (see SYSTICKFCLKDIV register in CLKCTL0 register section) as the clock source and enables the SysTick timer and the SysTick timer interrupt. Use DIV = 3.

$$\text{SYST\_RVR} = (\text{system tick timer clock frequency} \times 10 \text{ ms}) - 1 = (24 \text{ MHz} \times 10 \text{ ms}) - 1 = 240\,000 - 1 = 239\,999 = 0x0003\,A97F$$

System clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the FRO 12 MHz clock.

$$\text{SYST\_RVR} = (\text{system clock frequency} \times 10 \text{ ms}) - 1 = (12 \text{ MHz} \times 10 \text{ ms}) - 1 = 120\,000 - 1 = 119\,999 = 0x0001\,D4BF$$

29.4 Memory map and register description

29.4.1 SYSTICK register descriptions

29.4.1.1 SysTick Timer memory map

The systick timer registers are located on the private peripheral bus of each CPU. The appropriate systick timer is accessed depending on the current security state.

SYSTICK base address: E000_E010h

Offset	Register	Width (In bits)	Access	Reset value
10h	System Timer Control and status register (SYST_CSR)	32	RW	0000_0000h
14h	System Timer Reload value register (SYST_RVR)	32	RW	See section
18h	System Timer Current value register (SYST_CVR)	32	RW	See section
1Ch	System Timer Calibration value register (SYST_CALIB)	32	R	C000_0000h

29.4.1.2 System Timer Control and status register (SYST_CSR)

Offset

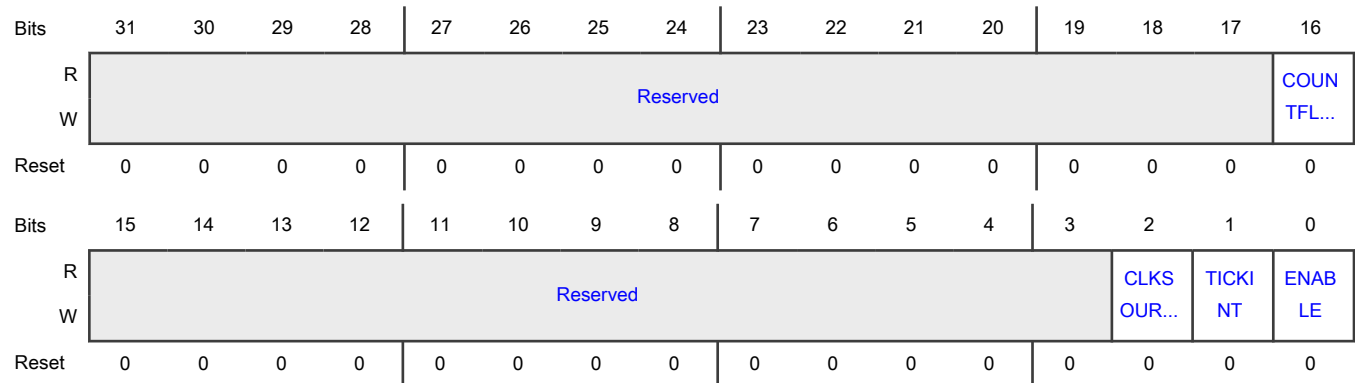
Register	Offset
SYST_CSR	10h

Function

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the CPU.

This register determines the clock source for the system tick timer.

Diagram



Fields

Field	Function
31-17 —	Reserved
16	Returns 1 if the SysTick timer counted to 0 since the last read of this register. Cleared automatically when read or when the counter is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
COUNTFLAG	
15-3 —	Reserved
2 CLKSOURCE	System Tick clock source selection When 1, the system clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKFCLKDIV) is selected as the reference clock. NOTE When the system tick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.
1 TICKINT	System Tick interrupt enable When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.
0 ENABLE	System Tick counter enable When 1, the counter is enabled. When 0, the counter is disabled.

29.4.1.3 System Timer Reload value register (SYST_RVR)

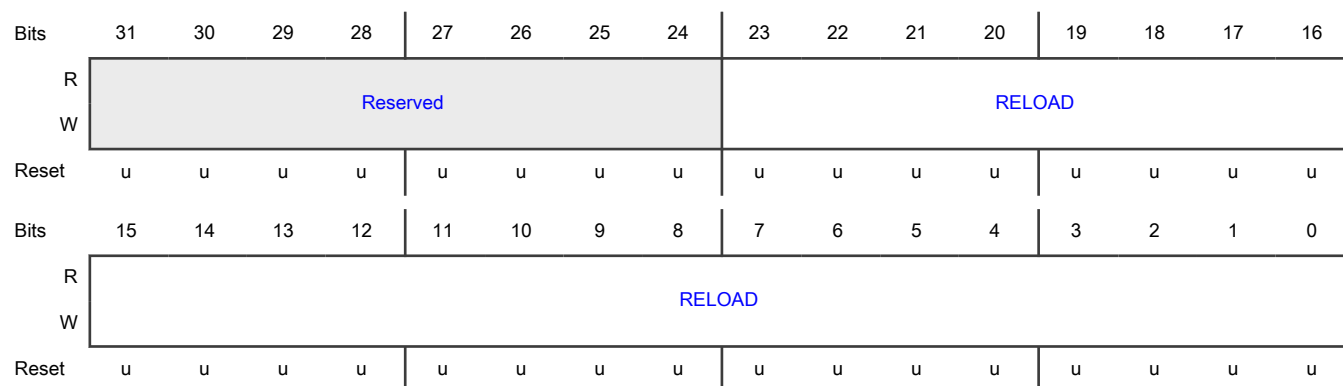
Offset

Register	Offset
SYST_RVR	14h

Function

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Diagram



Fields

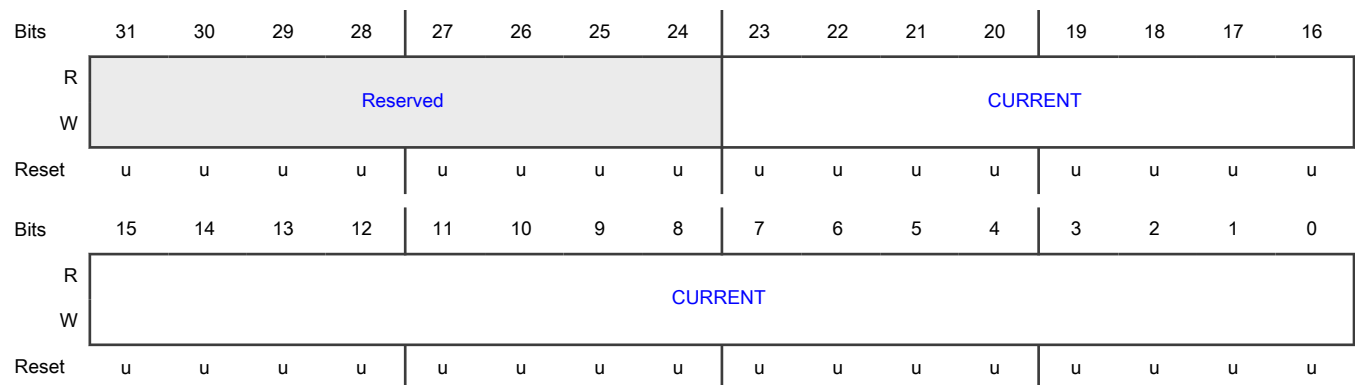
Field	Function
31-24 —	Reserved
23-0 RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.

29.4.1.4 System Timer Current value register (SYST_CVR)**Offset**

Register	Offset
SYST_CVR	18h

Function

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Diagram**Fields**

Field	Function
31-24 —	Reserved
23-0 CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in SYST_CSR.

29.4.1.5 System Timer Calibration value register (SYST_CALIB)

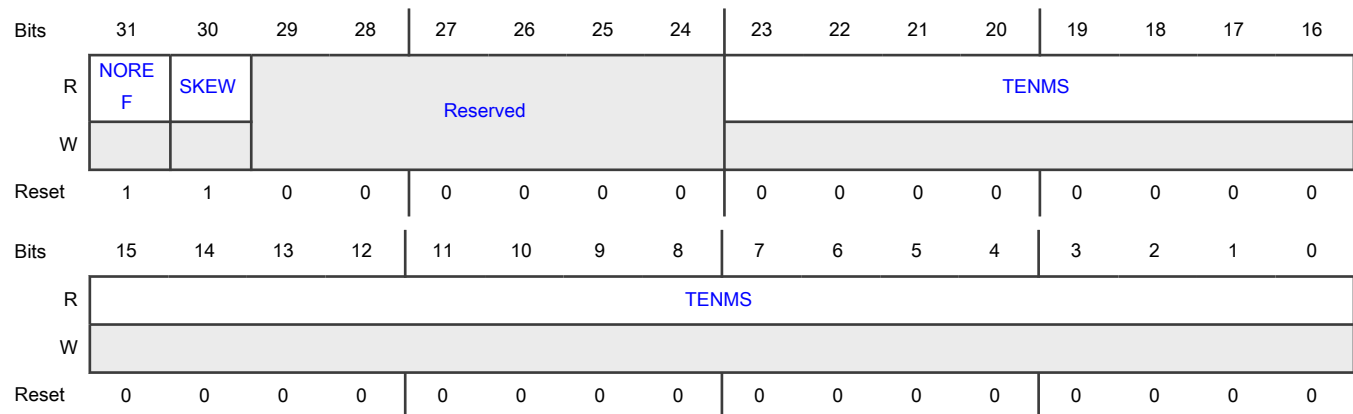
Offset

Register	Offset
SYST_CALIB	1Ch

Function

The value of the SYST_CALIB register is read-only and provides the SysTick calibration value and parameters.

Diagram



Fields

Field	Function
31 NOREF	Indicates whether an external reference clock is available. This bit is loaded from the SYSTCTL0 register SYSTEM_STICK_CALIB for the secure SysTick, or SYSTEM_NSTICK_CALIB for the non-secure SysTick. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.
30 SKEW	This bit is loaded from the SYSTCTL0 register SYSTEM_STICK_CALIB for the secure SysTick, or SYSTEM_NSTICK_CALIB for the non-secure SysTick. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.
29-24 —	Reserved
23-0 TENMS	Reload value from the a register in the SYSTCTL block. This field is loaded from the SYSTCTL0 register SYSTEM_STICK_CALIB for the secure SysTick, or SYSTEM_NSTICK_CALIB for the non-secure SysTick.

Chapter 30

Standard counter/timers (CTIMER)

30.1 Chip-specific CTIMER information

Table 390. Reference links to related information

Topic	Related module	Reference
Full description	CTIMER	CTIMER
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

The chip does not include timer-specific capture input pins. Instead there are 16 shared ct_in[0:15] functions, each of which may be brought in from a selection of pins. These 16 inputs go to all 5 CTIMERS via PMUXes (Peripheral input Muxes) in front of each CTIMER. So, user can elect to use any of these inputs for any of the timers. The inputs can be used for capture or clock.

30.1.1 Module instances

This device has five instances of the CTIMER module, CTIMER0, CTIMER1, CTIMER2, CTIMER3, CTIMER4.

30.1.2 Initialization

- Enable the clock to the CTIMER in the PSCCTL2 register (See CLKCTL). This enables the register interface and the peripheral function clock.
- Select a clock source for the CTIMER using the appropriate CT32BIT0FCLKSEL register.
- Clear the CTIMER peripheral reset in the PRSTCTL2 register by writing to the PRSTCTL2_CLR register (See RSTCTL).
- Each CTIMER provides interrupts to the NVIC. Please refer MCR and CCR in CTIMER register section for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, please refer NVIC Interrupt Assignments.
- Select timer pins and pin modes as needed through the relevant IOPCTL registers.
- The CTIMER DMA request lines are connected to the DMA trigger inputs via the DMAC0_ITRIG_SEL registers (See INPUTMUX). Note that timer DMA request outputs are connected to DMA trigger inputs.

30.2 Introduction

Each Counter/timer is designed to count cycles of the CTIMER function clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

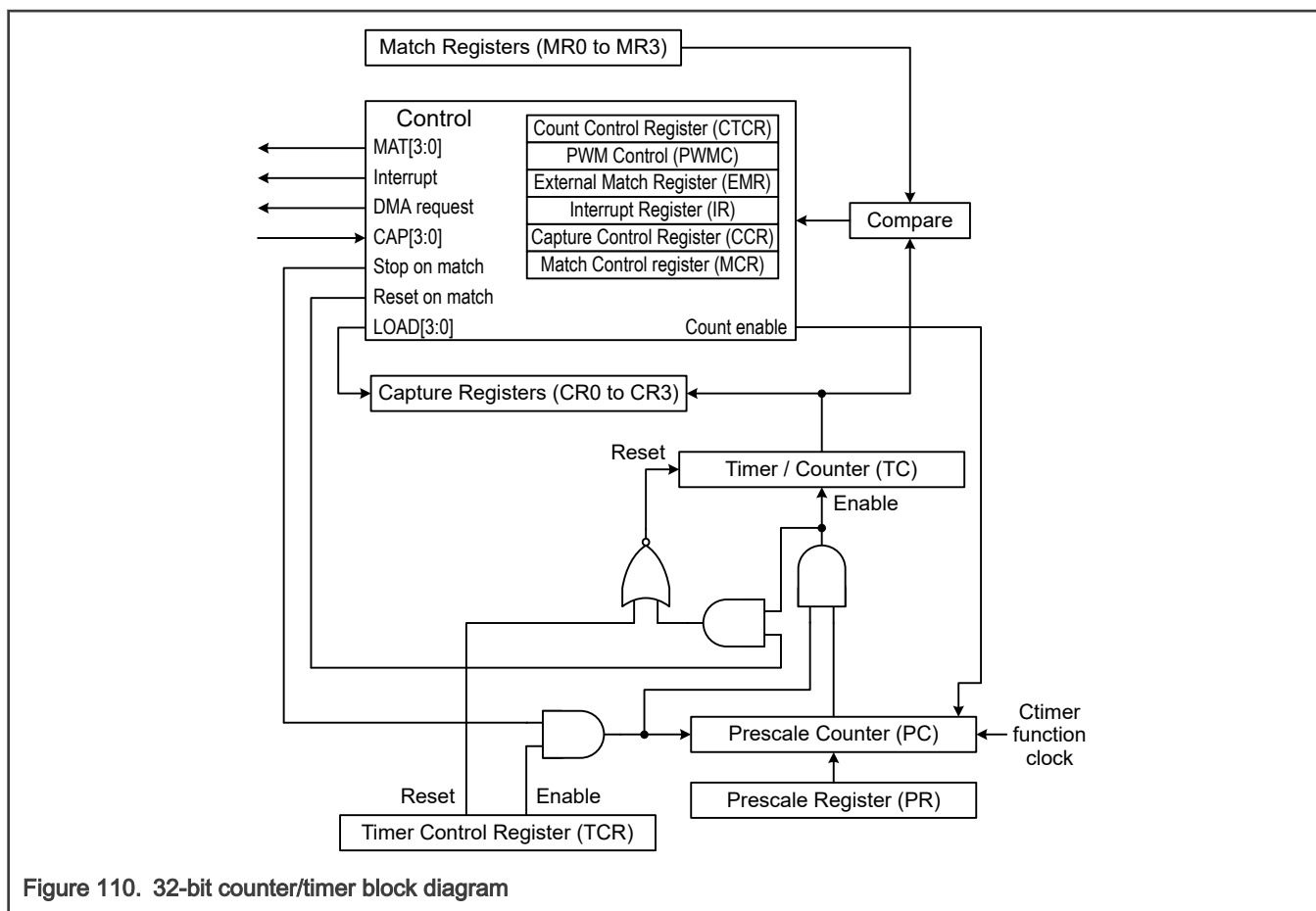
In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

30.2.1 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. The timers include external capture and match pin connections.
- Counter or timer operation.
- Each CTIMER has a selection of function clocks that may be asynchronous to other system clocks, please refer CLKCTL1_CT32BIT0CLKSEL and CLKCTL1_CT32BIT4CLKSEL.
- Up to four 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs for each timer that are actually available on device pins may vary by device.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
 - Continuous operation with optional interrupt generation on match.
 - Optional auto-reload from match shadow registers when counter is reset.
 - Stop timer on match with optional interrupt generation.
 - Reset timer on match with optional interrupt generation.
- For each timer, up to 4 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by device):
 - Set LOW on match.
 - Set HIGH on match.
 - Toggle on match.
 - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by device.)
- Up to 2 match registers can be used to generate DMA requests.

30.2.2 Block Diagram

The block diagram for the timers is shown below.



30.3 Signals

The tables below give a summary of each of the Timer/Counter related pins and Recommended IOPCTL settings. Also note that different part number and package variations may provide different CTIMER related pin functions.

Table 391. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER2_CAP3:0 CTIMER3_CAP3:0 CTIMER4_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Count Control Register (CTCR) .
CTIMER0_MAT3:0 CTIMER1_MAT3:0 CTIMER2_MAT3:0 CTIMER3_MAT3:0 CTIMER4_MAT3:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

Table 392. Suggested CTIMER timer pin settings

IOPCTL bit(s)	Name	Comment
3:0	FSEL	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless simulated open-drain output is desired.
11	IENA	Set to 0 (input function not inverted).

30.3.1 Multiple Capture (CAP) and Match (MAT) pins

Software can select from multiple pins for the CAP or MAT functions in the IOPCTL registers. Note that match conditions may be used internally without the use of a device pin.

30.3.1.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB CTIMER function clock. For more details see [Count Control Register \(CTCR\)](#).

30.3.1.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The [External Match Register \(EMR\)](#) and [PWM Control Register \(PWMC\)](#) control the functionality of this output.

30.4 Application information

- Interval Timer for counting internal events.
- PWM outputs
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

30.5 Functional description

The figure below shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

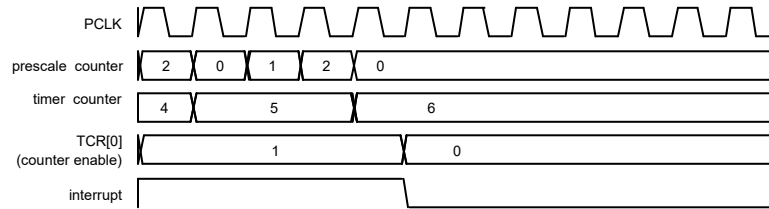


Figure 111. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

The figure below shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

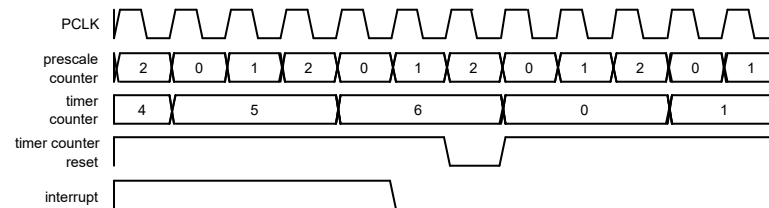


Figure 112. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

30.5.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

NOTE

When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

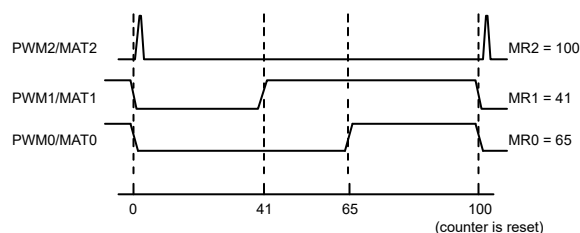


Figure 113. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

30.5.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See [Interrupt Register. \(IR\)](#). A DMA request will be cleared automatically when it is handled by the DMA controller.

NOTE

Timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

30.6 Memory map and Register definition

This section includes the CTIMER module memory map and detailed descriptions of all registers.

30.6.1 Counter/Timer register descriptions

30.6.1.1 CTIMER memory map

CTIMER0 base address: 4002_8000h

CTIMER1 base address: 4002_9000h

CTIMER2 base address: 4002_A000h

CTIMER3 base address: 4002_B000h

CTIMER4 base address: 4002_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Interrupt Register. (IR)	32	RW	See section
4h	Timer Control Register (TCR)	32	RW	See section
8h	Timer Counter (TC)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
Ch	Prescale Register (PR)	32	RW	0000_0000h
10h	Prescale Counter. (PC)	32	RW	0000_0000h
14h	Match Control Register. (MCR)	32	RW	See section
18h - 24h	Match Register (MR0 - MR3)	32	RW	0000_0000h
28h	Capture Control Register (CCR)	32	RW	See section
2Ch - 38h	Capture Register (CR0 - CR3)	32	R	0000_0000h
3Ch	External Match Register (EMR)	32	RW	See section
70h	Count Control Register (CTCR)	32	RW	See section
74h	PWM Control Register (PWMC)	32	RW	See section
78h - 84h	Match Shadow Register (MSR0 - MSR3)	32	RW	0000_0000h

30.6.1.2 Interrupt Register. (IR)

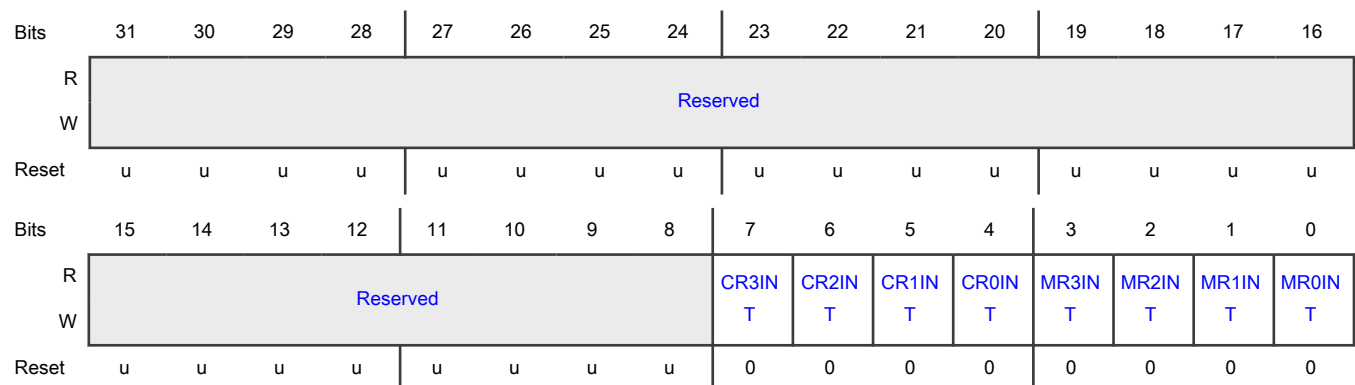
Offset

Register	Offset
IR	0h

Function

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.

Diagram



Fields

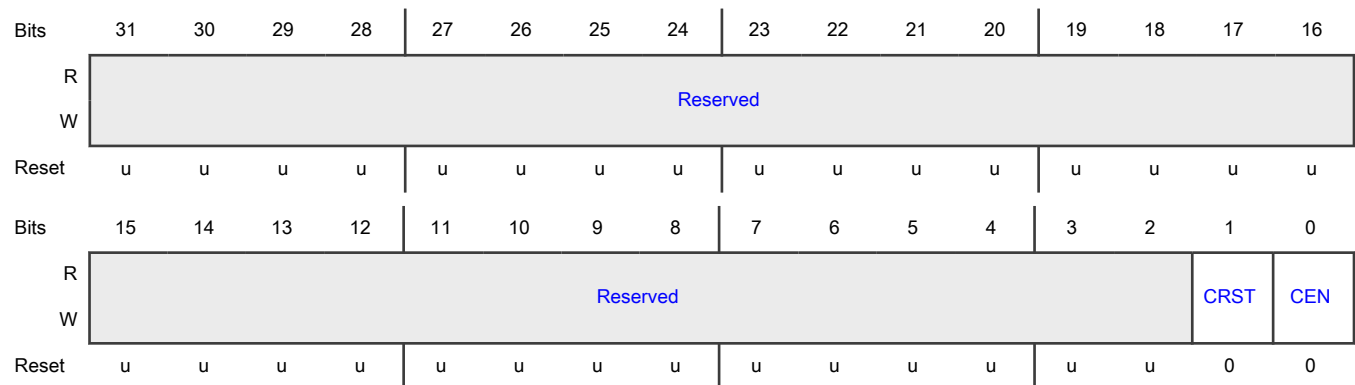
Field	Function
31-8 —	Reserved Read value is undefined, only zero should be written.
7-4 CRnINT	Interrupt flag for capture channel n event
3-0 MRnINT	Interrupt flag for match channel n

30.6.1.3 Timer Control Register (TCR)**Offset**

Register	Offset
TCR	4h

Function

The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.

Diagram**Fields**

Field	Function
31-2 —	Reserved Read value is undefined, only zero should be written.
1 CRST	Counter reset. 0b - Disabled. Do nothing.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the CTIMER function clock. The counters remain reset until TCR[1] is returned to zero.
0 CEN	Counter enable. 0b - Disabled. The counters are disabled. 1b - Enabled. The Timer Counter and Prescale Counter are enabled.

30.6.1.4 Timer Counter (TC)

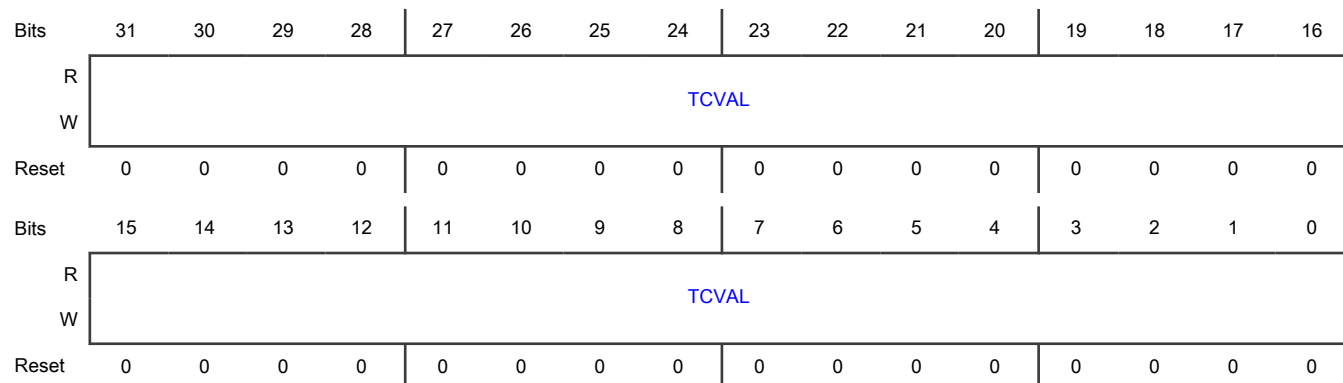
Offset

Register	Offset
TC	8h

Function

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed. The 32 bit TC is incremented every PR+1 cycles of the CTIMER function clock. The TC is controlled through the TCR.

Diagram



Fields

Field	Function
31-0 TCVAL	Timer counter value.

30.6.1.5 Prescale Register (PR)

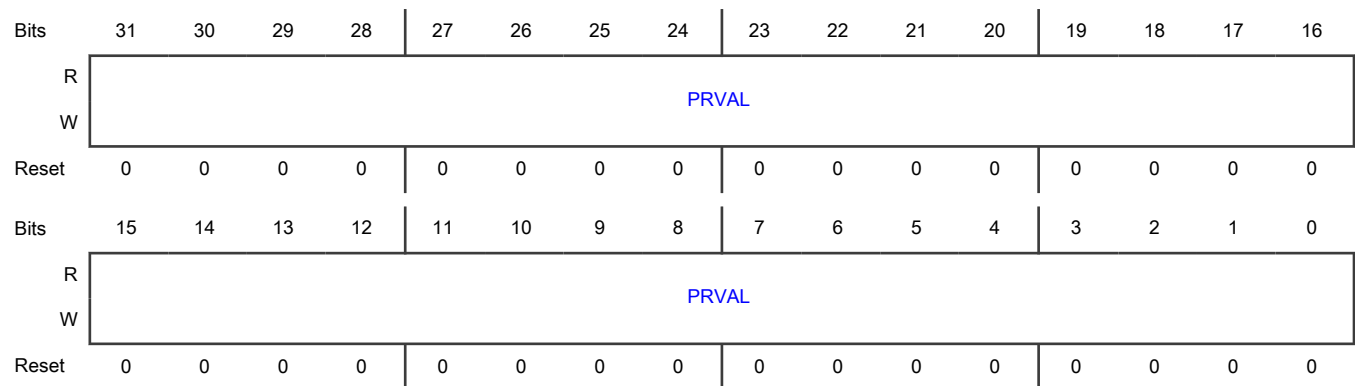
Offset

Register	Offset
PR	Ch

Function

When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.

Diagram



Fields

Field	Function
31-0 PRVAL	Prescale reload value.

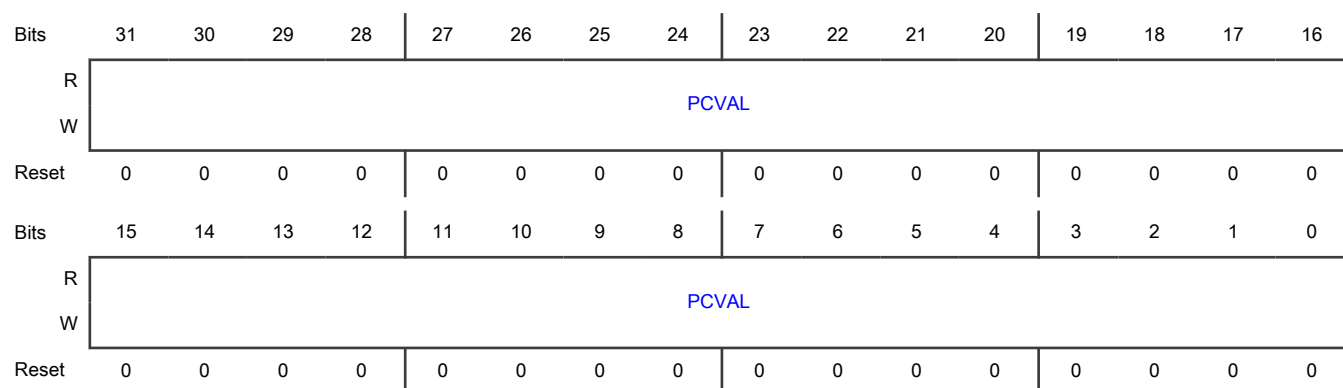
30.6.1.6 Prescale Counter. (PC)

Offset

Register	Offset
PC	10h

Function

The 32-bit Prescale Counter controls division of the CTIMER function clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every CTIMER function clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next CTIMER function clock. This causes the Timer Counter to increment on every CTIMER function clock when PR = 0, every 2 CTIMER function clocks when PR = 1, and so on. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.

Diagram**Fields**

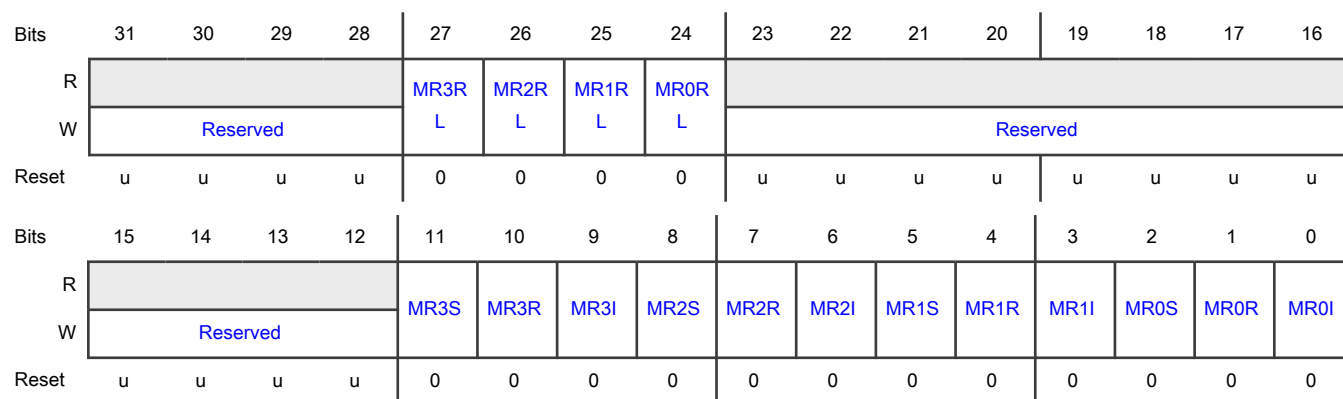
Field	Function
31-0 PCVAL	Prescale counter value.

30.6.1.7 Match Control Register. (MCR)**Offset**

Register	Offset
MCR	14h

Function

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Diagram

Fields

Field	Function
31-28 —	Reserved Read value is undefined, only zero should be written.
27-24 MRnRL	Reload MRn Reload with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. 0b - Disabled 1b - Enabled
23-12 —	Reserved Read value is undefined, only zero should be written. 0000_0000_0000b - Disabled 0000_0000_0001b - Enabled
11 MR3S	Stop on MR3 TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0b - Disabled 1b - Enabled
10 MR3R	Reset on MR3 TC will be reset if MR3 matches it. 0b - Disabled 1b - Enabled
9 MR3I	Interrupt on MR3 An interrupt is generated when MR3 matches the value in the TC. 0b - Disabled 1b - Enabled
8 MR2S	Stop on MR2 TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0b - Disabled 1b - Enabled
7 MR2R	Reset on MR2 TC will be reset if MR2 matches it. 0b - Disabled 1b - Enabled
6	Interrupt on MR2

Table continues on the next page...

Table continued from the previous page...

Field	Function
MR2I	An interrupt is generated when MR2 matches the value in the TC. 0b - Disabled 1b - Enabled
5 MR1S	Stop on MR1 TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0b - Disabled 1b - Enabled
4 MR1R	Reset on MR1 TC will be reset if MR1 matches it. 0b - Disabled 1b - Enabled
3 MR1I	Interrupt on MR1 An interrupt is generated when MR1 matches the value in the TC. 0b - Disabled 1b - Enabled
2 MR0S	Stop on MR0 TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0b - Disabled 1b - Enabled
1 MR0R	Reset on MR0 TC will be reset if MR0 matches it. 0b - Disabled 1b - Enabled
0 MR0I	Interrupt on MR0 An interrupt is generated when MR0 matches the value in the TC. 0b - Disabled 1b - Enabled

30.6.1.8 Match Register (MR0 - MR3)

Offset

Register	Offset
MR0	18h
MR1	1Ch
MR2	20h
MR3	24h

Function

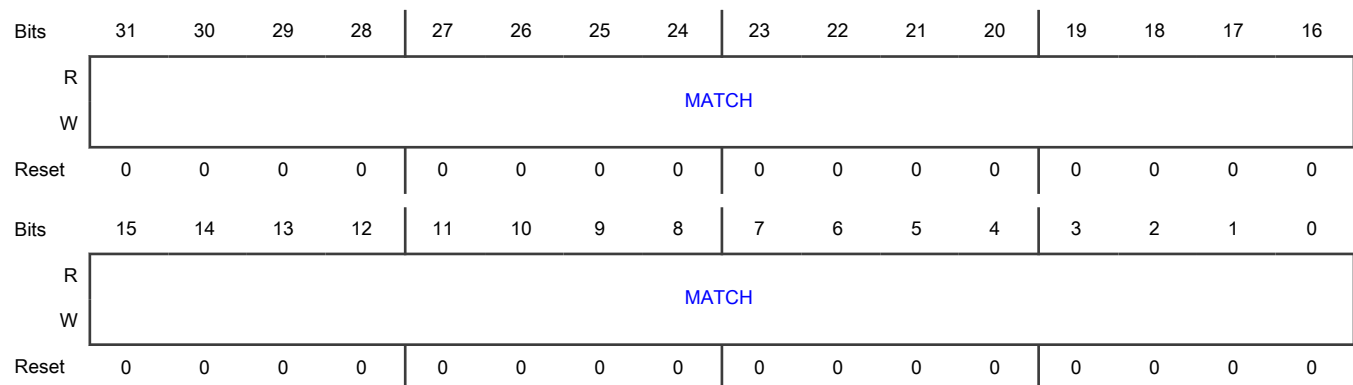
The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

If the associated MRxRL bit in the Match Control Register is set, the Match Register will be automatically reloaded with the current contents of its corresponding Match Shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that advances the TC to zero.

NOTE

The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the Timer Control Register.

Diagram



Fields

Field	Function
31-0 MATCH	Timer counter match value.

30.6.1.9 Capture Control Register (CCR)

Offset

Register	Offset
CCR	28h

Function

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

NOTE

If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				CAP3I	CAP3 FE	CAP3 RE	CAP2I	CAP2 FE	CAP2 RE	CAP1I	CAP1 FE	CAP1 RE	CAP0I	CAP0 FE	CAP0 RE
W																
Reset	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	Reserved Read value is undefined, only zero should be written.
11 CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt. 0b - Disabled 1b - Enabled
10 CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled
8 CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt. 0b - Disabled 1b - Enabled
7 CAP2FE	Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled
6 CAP2RE	Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled
5 CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt. 0b - Disabled 1b - Enabled
4 CAP1FE	Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled
3 CAP1RE	Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled
2 CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt. 0b - Disabled 1b - Enabled
1 CAP0FE	Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 CAP0RE	Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0b - Disabled 1b - Enabled

30.6.1.10 Capture Register (CR0 - CR3)

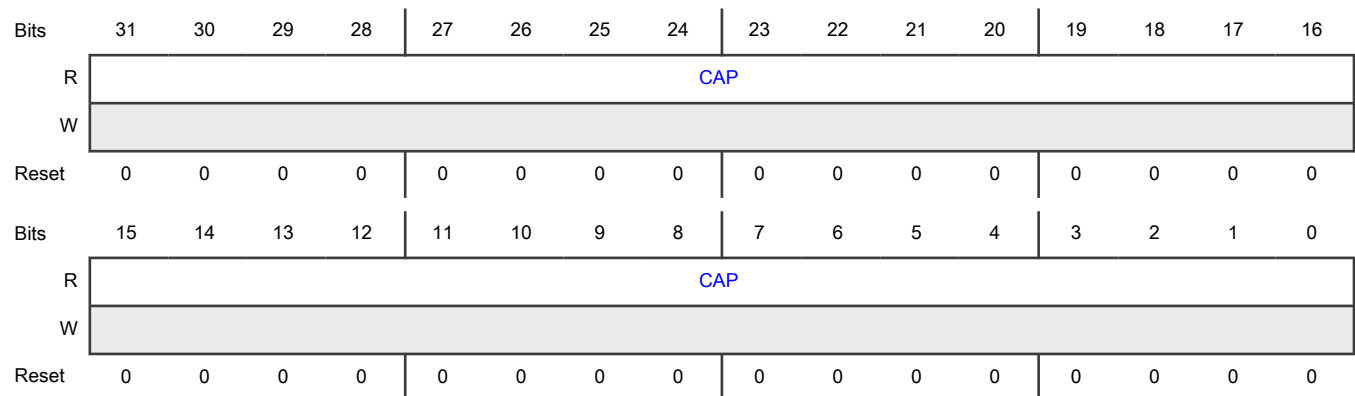
Offset

Register	Offset
CR0	2Ch
CR1	30h
CR2	34h
CR3	38h

Function

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges. The register is loaded with the value of TC when there is an event on the CAPn. input.

Diagram



Fields

Field	Function
31-0 CAP	Timer counter capture value.

30.6.1.11 External Match Register (EMR)

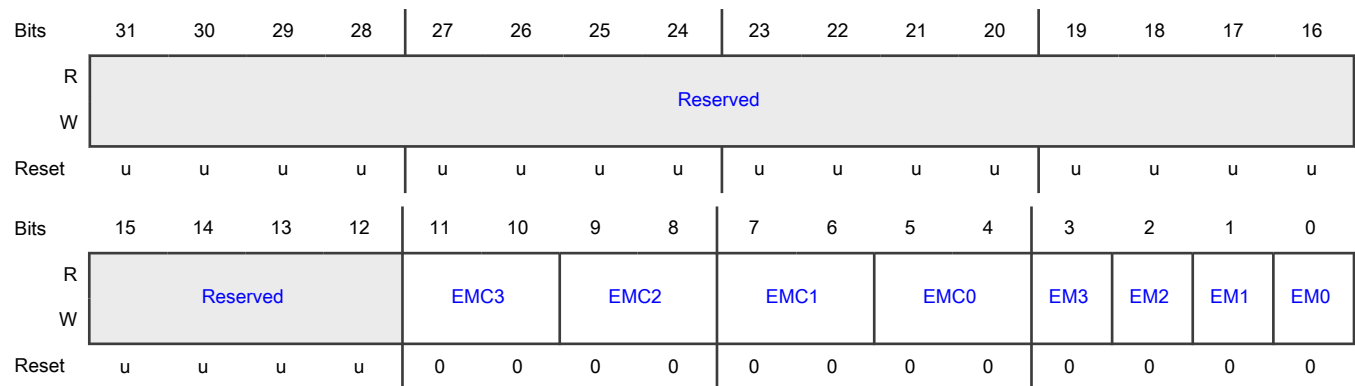
Offset

Register	Offset
EMR	3Ch

Function

The EMR controls the match function and the external match pins.

Diagram



Fields

Field	Function
31-12 —	Reserved Read value is undefined, only zero should be written.
11-10 EMC3	External Match Control 3 Determines the functionality of External Match 3. 00b - Do Nothing. 01b - Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out). 10b - Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out). 11b - Toggle. Toggle the corresponding External Match bit/output.
9-8 EMC2	External Match Control 2 Determines the functionality of External Match 2. 00b - Do Nothing. 01b - Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out). 11b - Toggle. Toggle the corresponding External Match bit/output.
7-6 EMC1	External Match Control 1 Determines the functionality of External Match 1. 00b - Do Nothing 01b - Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out). 10b - Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out). 11b - Toggle. Toggle the corresponding External Match bit/output.
5-4 EMC0	External Match Control 0 Determines the functionality of External Match 0. 00b - Do Nothing. 01b - Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out). 10b - Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out). 11b - Toggle. Toggle the corresponding External Match bit/output.
3 EM3	External Match 3 This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
2 EM2	External Match 2 This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
1 EM1	External Match 1 This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
0 EM0	External Match 0 This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.

30.6.1.12 Count Control Register (CTCR)

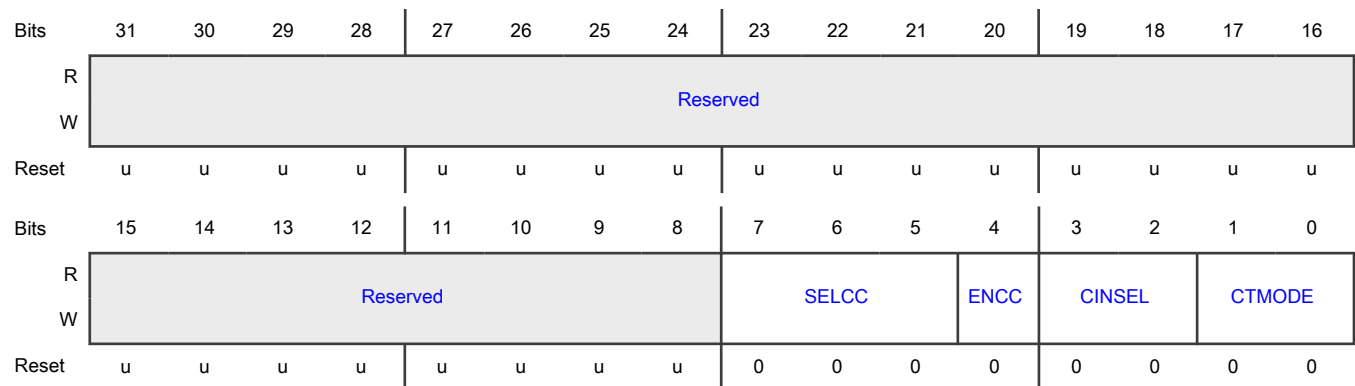
Offset

Register	Offset
CTCR	70h

Function

The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.

Diagram



Fields

Field	Function
31-8 —	Reserved Read value is undefined, only zero should be written.
7-5 SELCC	Edge select When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved. 000b - Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set). 001b - Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set). 010b - Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set). 011b - Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set). 100b - Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set). 101b - Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 ENCC	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.
3-2 CINSEL	Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. Note: If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer. 00b - Channel 0. CAPn.0 for CTIMERn 01b - Channel 1. CAPn.1 for CTIMERn 10b - Channel 2. CAPn.2 for CTIMERn 11b - Channel 3. CAPn.3 for CTIMERn
1-0 CTMODE	The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting. When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the CTIMER function clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented. Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the CTIMER function clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the CTIMER function clock. Consequently, duration of the HIGH/LOW levels on the same CAP input in this case cannot be shorter than a CTIMER function clock. Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software. 00b - Timer Mode. Incremented every rising CTIMER function clock edge. 01b - Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2. 10b - Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2. 11b - Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.

30.6.1.13 PWM Control Register (PWMC)

Offset

Register	Offset
PWMC	74h

Function

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR). Note that different part number and package variations may provide different match output pin functions.

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

The PWMCON enables PWM mode for the external match pins.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												PWME	PWME	PWME	PWME
W													N3	N2	N1	N0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0

Fields

Field	Function
31-4 —	Reserved Read value is undefined, only zero should be written.
3 PWME3	PWM mode enable for channel3. Note: It is recommended to use match channel 3 to set the PWM cycle. 0b - Match. CTIMERn_MAT3 is controlled by EM3. 1b - PWM. PWM mode is enabled for CT132Bn_MAT3.
2 PWME2	PWM mode enable for channel2. 0b - Match. CTIMERn_MAT2 is controlled by EM2. 1b - PWM. PWM mode is enabled for CTIMERn_MAT2.
1	PWM mode enable for channel1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
PWMEN1	0b - Match. CTIMERn_MAT01 is controlled by EM1. 1b - PWM. PWM mode is enabled for CTIMERn_MAT1.
0 PWMEN0	PWM mode enable for channel0. 0b - Match. CTIMERn_MAT0 is controlled by EM0. 1b - PWM. PWM mode is enabled for CTIMERn_MAT0.

30.6.1.14 Match Shadow Register (MSR0 - MSR3)

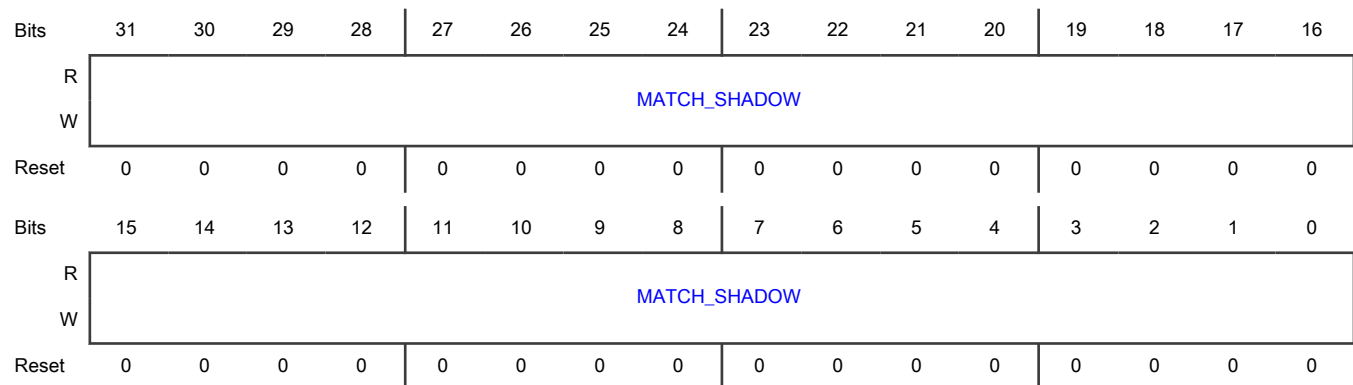
Offset

Register	Offset
MSR0	78h
MSR1	7Ch
MSR2	80h
MSR3	84h

Function

If enabled, the Match Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.

Diagram



Fields

Field	Function
31-0 MATCH_SHADOW	Timer counter match shadow value.

Chapter 31

OS Event Timer (OSTIMER)

31.1 Chip-specific OSTIMER information

Table 393. Reference links to related information

Topic	Related module	Reference
Full description	OSTIMER	OSTIMER
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

31.1.1 Module instances

This device has two instances of the OSTIMER module, OSTIMER_CM33 and OSTIMER_DSP.

31.2 Overview

OSTIMER includes a shared 64-bit timer and separate 64-bit match and capture functions for the Cortex-M33 core and DSP CPUs.

31.2.1 Block diagram

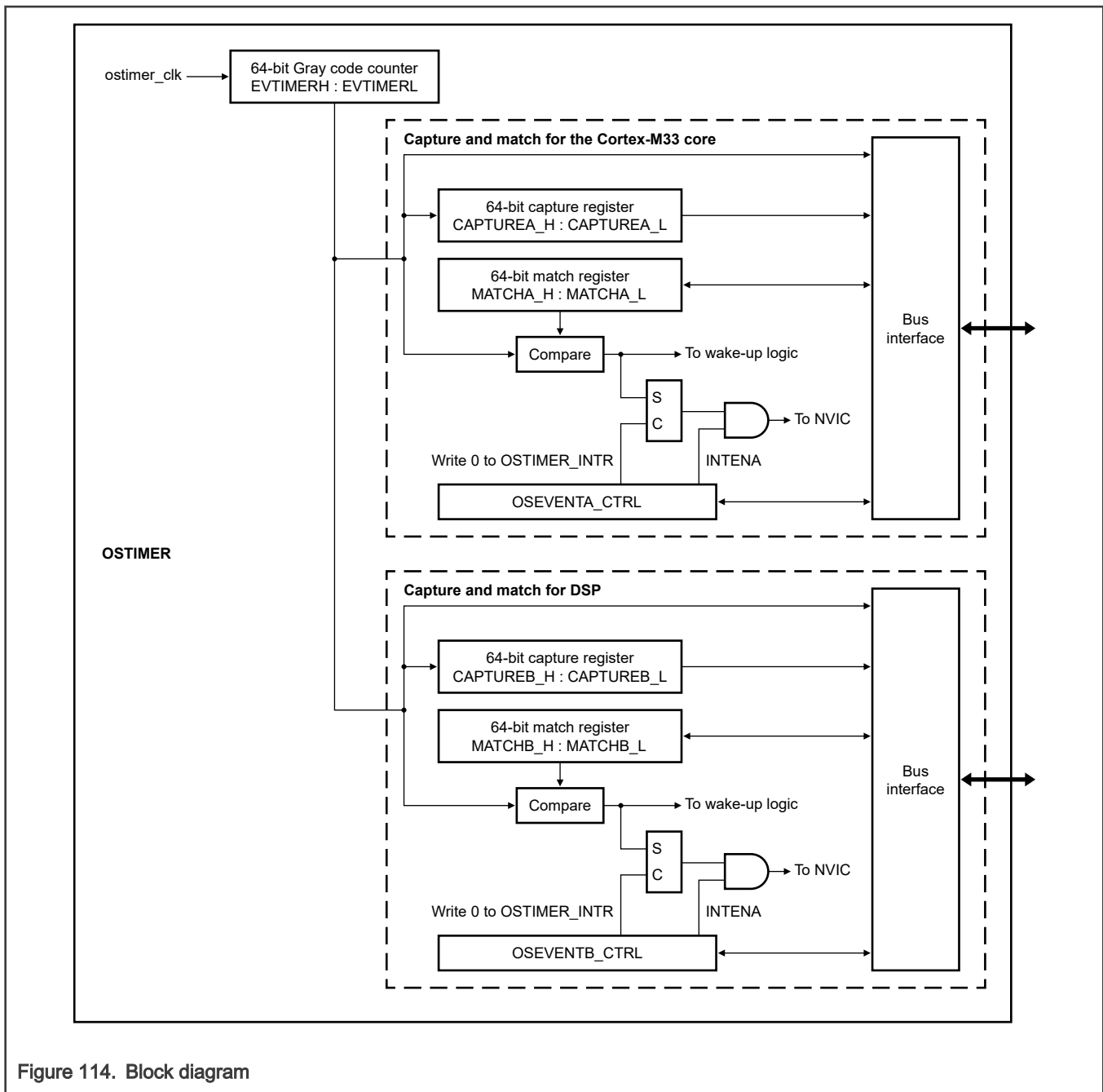


Figure 114. Block diagram

31.2.2 Features

- Provides a 64-bit Gray code counter. Using Gray code means that the timer can run at a frequency unrelated to any CPU clock and can still be read by a CPU without a synchronization delay. Gray code is a reflected binary code that changes in a single-bit position for each increment.
- Supports separate functions for each CPU:
 - A capture register can copy the main counter value when triggered by a CPU request.
 - A match register can be compared to the main counter and can optionally generate an interrupt or wake-up event.

31.3 Functional description

The following sections describe the functional details of this module.

31.3.1 Shared event or timestamp timer

The 64-bit shared EVTimer initializes after chip power-up and then counts up continuously. The typical clock for this timer is the 1 MHz low-power oscillator.

This timer is implemented as a Gray code counter that enables the counter to be read asynchronously by any of the processing domains or captured by a capture register running on an asynchronous clock. The output of this shared EVTimer is connected to the processor-specific submodules.

31.3.2 CPU-specific match, capture, and interrupt generation

The submodules associated with each CPU include:

- A separate bus interface.
- 64 bits of capture values, available in [CAPTURE_L](#) and [CAPTURE_H](#).
- 64 bits of match values, available in [MATCH_H](#) and [MATCH_L](#).
- A control register, which includes an interrupt request flag and an interrupt enable field.

Capture registers	64 bits of capture values are available in CAPTURE_L and CAPTURE_H for each CPU. When the CPU issues a capture command, capture registers store the current value of the main EVTimer.
Match registers and interrupt request	64 bits of match values are available in MATCH_H and MATCH_L for each bus interface. The EVTimer output is compared against this combined value for interrupt or wake-up generation. A match to this register pair sets OSEVENT_CTRL[OSTIMER_INTRFLAG] , which you can enable to generate an interrupt or wake-up request. You must write values to the match registers in Gray code. OSEVENT_CTRL[MATCH_WR_RDY] must be 0 before you write to the match registers. When you write a new value to the match registers, you must write to MATCH_L before the write to MATCH_H .

31.3.3 Reset

For OSTIMER, the reset function is more complicated as compared to most other functions.

- Only the POR and software reset can reset the shared EVTimer. "Software reset" refers to a complete reset of the module.
- The capture and match registers are reset by a system reset. "System reset" refers to a reset from the RESET n pin, SYSRESETREQ from the CPU, a POR, or any other internal low-voltage resets.
- The control register is reset by the system reset and by a software reset.

The following table shows a list of resets for OSTIMER registers.

Table 394. OSTIMER registers and their corresponding resets

Registers	POR	Software reset
EVTIMERL and EVTIMERH	Yes	Yes
CAPTURE_L and CAPTURE_H	Yes	No
OSEVENT_CTRL	Yes	Yes

31.4 Initialization

Perform this procedure to initialize OSTIMER:

1. Enable the OSTIMER clock. This enables the register interface and the peripheral function clock.
2. Select a clock source for OSTIMER.
3. Clear the OSTIMER peripheral reset.
4. Enable the interrupt that OSTIMER provides to the interrupt controller. This allows the chip to wake up from Deep Sleep mode. You can connect this interrupt to DSP alternatively.

31.5 Memory map and register definition

This section includes the module's memory map and detailed descriptions of all registers.

31.5.1 OSTIMER register descriptions

31.5.1.1 OSTIMER memory map

OSTIMER.OSTIMER_CM33 base address: 4011_3000h

Offset	Register	Width (In bits)	Access	Reset value
0h	EVTIMER Low (EVTIMERL)	32	R	0000_0000h
4h	EVTIMER High (EVTIMERH)	32	R	0000_0000h
8h	Local Capture Low for CPU (CAPTURE_L)	32	R	0000_0000h
Ch	Local Capture High for CPU (CAPTURE_H)	32	R	0000_0000h
10h	Local Match Low for CPU (MATCH_L)	32	RW	FFFF_FFFFh
14h	Local Match High for CPU (MATCH_H)	32	RW	FFFF_FFFFh
1Ch	OSTIMER Control for CPU (OSEVENT_CTRL)	32	RW	0000_0000h

31.5.1.2 EVTIMER Low (EVTIMERL)

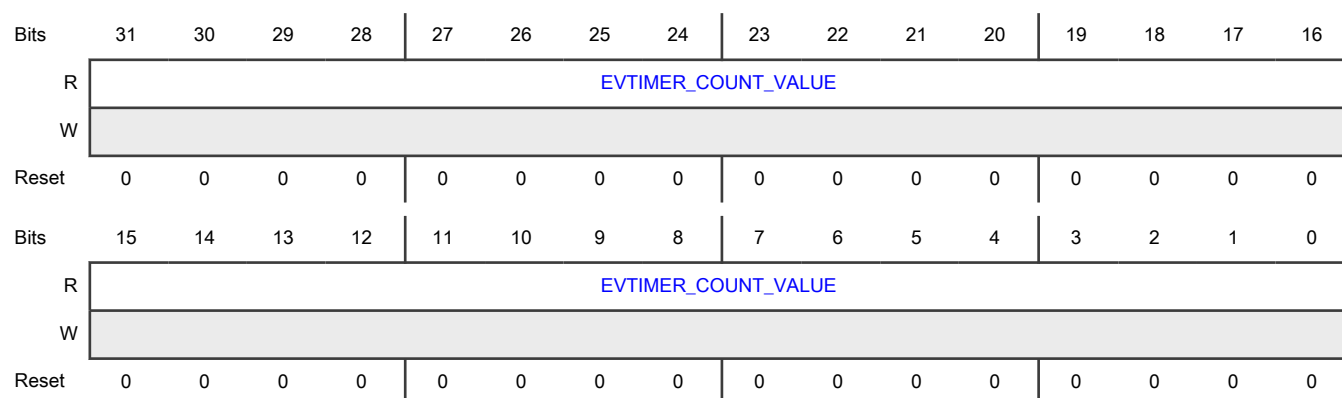
Offset

Register	Offset
EVTIMERL	0h

Function

Contains the immediate value of the central EVTimer. This is a Gray encoded counter register that is not synchronized with the bus clock.

Because two 32-bit reads are required to retrieve the entire counter value, it is possible for the counter to roll over between the reads of [EVTIMERL](#) and [EVTIMERH](#). Gray encoding ensures that even if this roll over occurs, the value read represents either the counter value immediately preceding or immediately following the rollover, if the bus clock rate is at least twice the module's clock rate (nominally 1 MHz), or if the bus is clocked by the same 1 MHz LP oscillator that provides the clock to the module.

Diagram**Fields**

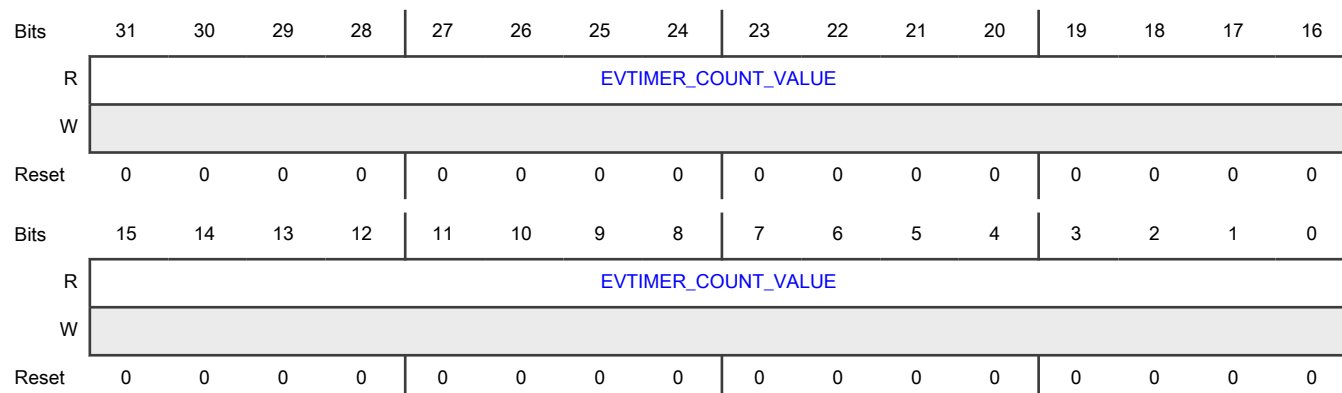
Field	Function
31-0	EVTimer Count Value
EVTIMER_COUNT_VALUE	Specifies the current value of the lower 32 bits of the central 64-bit EVTimer when you read this register.
NOTE Only one physical EVTimer exists and is readable from all domains.	

31.5.1.3 EVTIMER High (EVTIMERH)**Offset**

Register	Offset
EVTIMERH	4h

Function

Contains the higher value of the central EVTimer. This register resets only after POR or a software reset.

Diagram

Fields

Field	Function
31-0 EVTIMER_COUNT_VALUE	EVTimer Count Value Specifies the current value of the upper 32 bits of the central 64-bit EVTimer when you read this register.
NOTE Only one physical EVTimer exists and is readable from all domains.	

31.5.1.4 Local Capture Low for CPU (CAPTURE_L)

Offset

Register	Offset
CAPTURE_L	8h

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. A load is triggered by a load signal from the CPU (for example, an "arm_txev" pulse from the Cortex-M33 core CPU or a "tie_expstate" pulse from DSP). You must not execute a new capture command between reading [CAPTURE_L](#) and [CAPTURE_H](#), to avoid retrieving incoherent results.

For the instance associated with the DSP CPU: If a read of [CAPTURE_L](#) and [CAPTURE_H](#) closely follows the command to implement a capture, then you should insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Diagram

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

Fields

Field	Function
31-0	EVTimer Capture Value

Table continues on the next page...

Field	Function
CAPTURE_VALUE	Specifies the value of the lower 32 bits of the central 64-bit EVTimer when you read this register, at the time that the CPU generated the last capture signal. A separate pair of CAPTURE_L and CAPTURE_H is implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.

31.5.1.5 Local Capture High for CPU (CAPTURE_H)

Offset

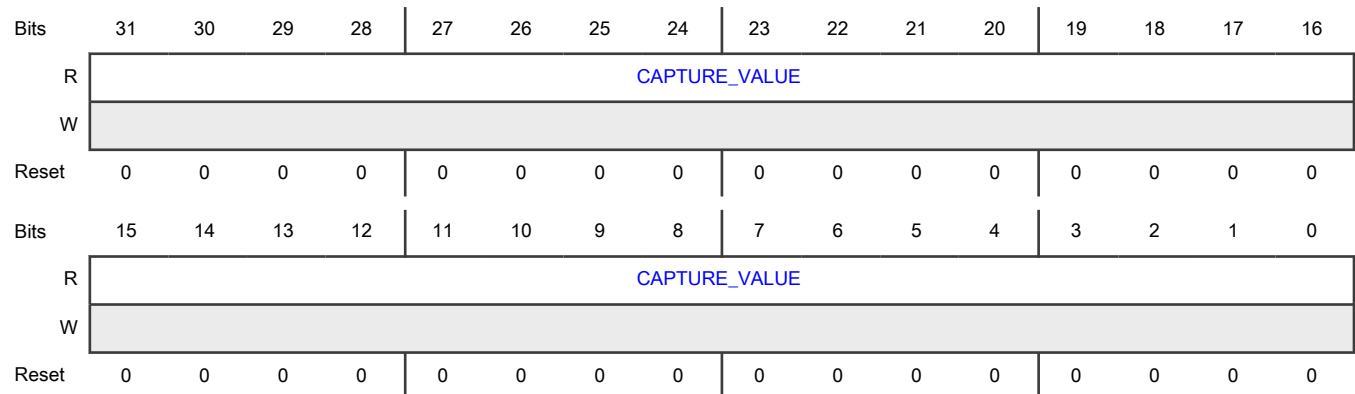
Register	Offset
CAPTURE_H	Ch

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. A load is triggered by a load signal from the CPU (for example, an "arm_txev" pulse from the Cortex-M33 core CPU or a "tie_expstate" pulse from DSP). You must not execute a new capture command between reading [CAPTURE_L](#) and [CAPTURE_H](#), to avoid retrieving incoherent results.

For the instance associated with the DSP CPU: If a read of [CAPTURE_L](#) and [CAPTURE_H](#) closely follows the command to implement a capture, then you must insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Diagram



Fields

Field	Function
31-0	EVTimer Capture Value
CAPTURE_VALUE	Specifies the value of the upper 32 bits of the central 64-bit EVTimer when you read this register, at the time that the CPU generated the last capture signal. A separate pair of CAPTURE_L and CAPTURE_H is implemented for each CPU reading its own capture value at the same pair of addresses.

31.5.1.6 Local Match Low for CPU (MATCH_L)

Offset

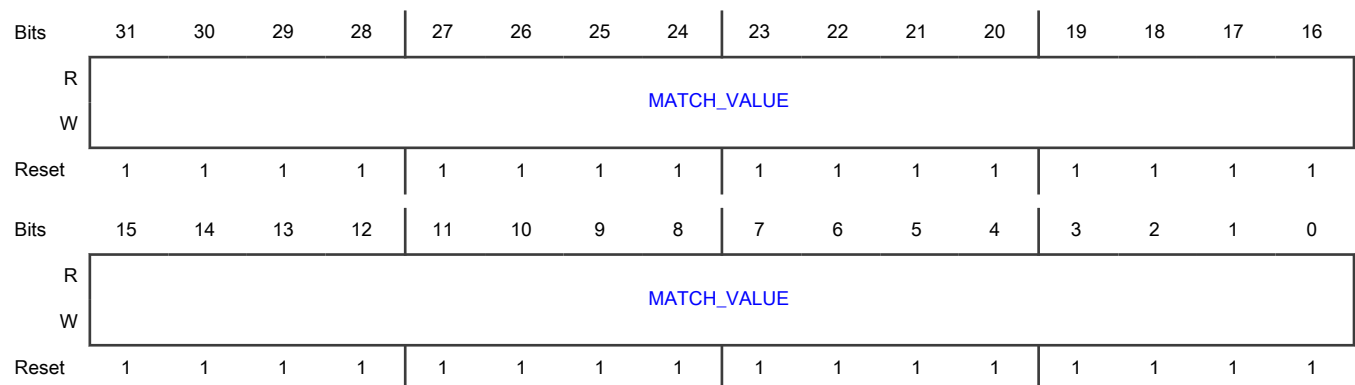
Register	Offset
MATCH_L	10h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to [MATCH_L](#) first, followed by the write to [MATCH_H](#).

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to [MATCH_H](#)). [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) indicates when it is safe to reload the match registers. You do not need to read [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram



Fields

Field	Function
31-0	EVTimer Match Value
MATCH_VALUE	Compares the value (lower 32 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled. A separate pair of MATCH_L and MATCH_H is implemented for each CPU reading its own local value at the same pair of addresses.

31.5.1.7 Local Match High for CPU (MATCH_H)

Offset

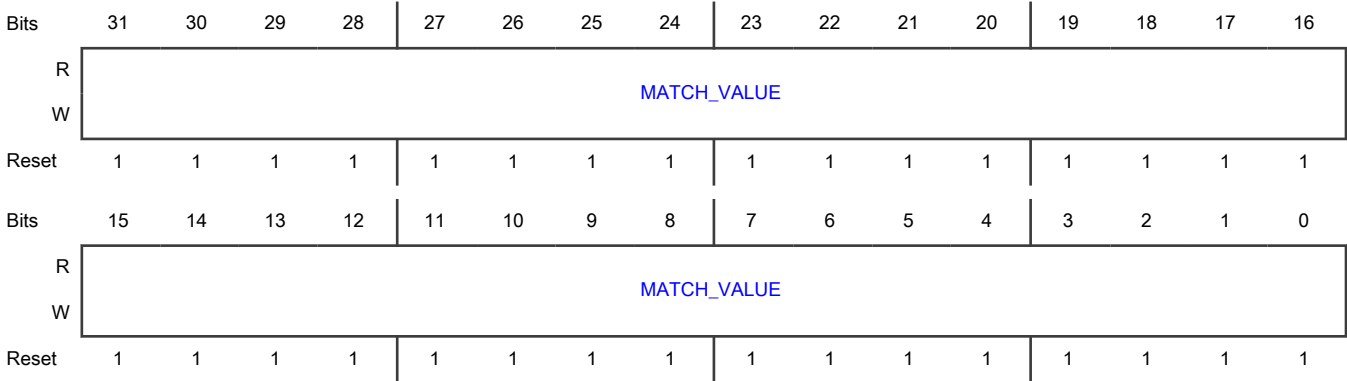
Register	Offset
MATCH_H	14h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to [MATCH_L](#) first, followed by the write to [MATCH_H](#).

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to [MATCH_H](#)). [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) indicates when it is safe to reload the match registers. You do not need to read [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram



Fields

Field	Function
31-0	EVTimer Match Value
MATCH_VALU E	Compares the value (upper 32 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled. A separate pair of MATCH_L and MATCH_H is implemented for each CPU reading its own local value at the same pair of addresses.

31.5.1.8 OSTIMER Control for CPU (OSEVENT_CTRL)

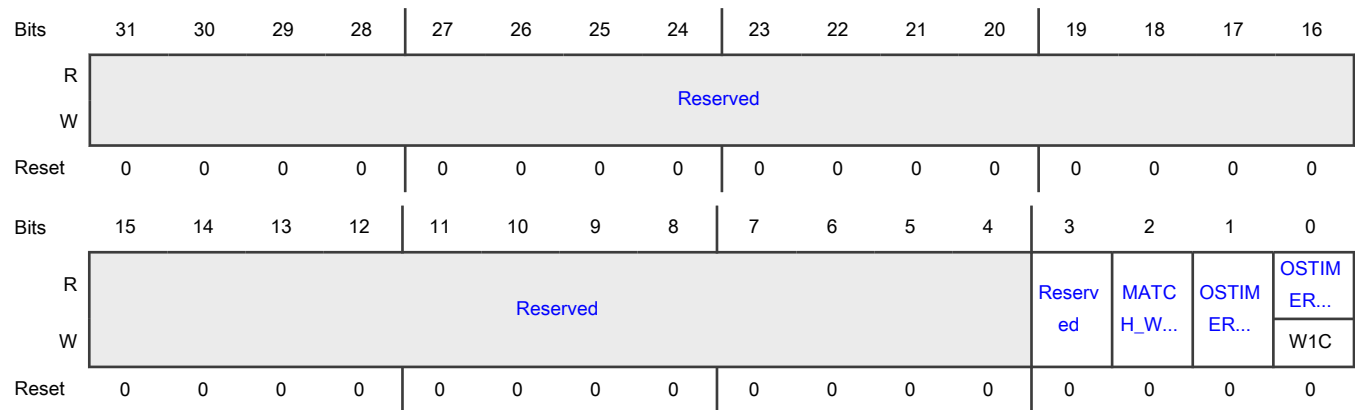
Offset

Register	Offset
OSEVENT_CTRL	1Ch

Function

Provides the interrupt flag and interrupt enable signals for each CPU. A separate OSEVENT_CTRL register is implemented for each CPU. Each CPU reads its own local value at the same address.

Diagram



Fields

Field	Function
31-4 —	Reserved
3 —	Reserved
2 MATCH_WR_RDY	EVTimer Match Write Ready Specifies that when this field is low, it is safe to reload (write) to the match registers. In typical applications, it is not necessary to read this field. The 64-bit match register value is transferred from a pair of shadow registers to the active match registers after you write to MATCH_H. You must not initiate a second write to MATCH_L and MATCH_H until after this transfer completes. This field becomes 0 after the transfer completes. It is not necessary to read this status field, if an interrupt has already occurred because of the first match value, or if it is certain (via some other means) that the required period of time (3 bus clocks) has elapsed.
1 OSTIMER_INT_ENA	Interrupt or Wake-Up Request Enables or blocks the interrupt or wake-up requests depending upon the value of OSEVENT_CTRL[OSTIMER_INTRFLAG]. If this value is 1, then this field asserts an interrupt or wake-up request to the domain processor. Otherwise, the interrupt or wake-up requests are blocked. 0b - Interrupts blocked 1b - Interrupts enabled
0 OSTIMER_INT_RFLAG	Interrupt Flag Sets an interrupt flag when a match occurs between the central 64-bit EVTimer and the value programmed in the MATCH_L and MATCH_H register pair for the associated CPU. Writes to clear this field are asynchronous. You must write 1 to this field before writing a new match value into MATCH_L and MATCH_H.

31.5.2 OSTIMER register descriptions

31.5.2.1 OSTIMER memory map

OSTIMER.OSTIMER_DSP base address: 4011_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	EVTIMER Low (EVTIMERL)	32	R	0000_0000h
4h	EVTIMER High (EVTIMERH)	32	R	0000_0000h
8h	Local Capture Low for CPU (CAPTURE_L)	32	R	0000_0000h
Ch	Local Capture High for CPU (CAPTURE_H)	32	R	0000_0000h
10h	Local Match Low for CPU (MATCH_L)	32	RW	FFFF_FFFFh
14h	Local Match High for CPU (MATCH_H)	32	RW	FFFF_FFFFh
1Ch	OSTIMER Control for CPU (OSEVENT_CTRL)	32	RW	0000_0000h

31.5.2.2 EVTIMER Low (EVTIMERL)

Offset

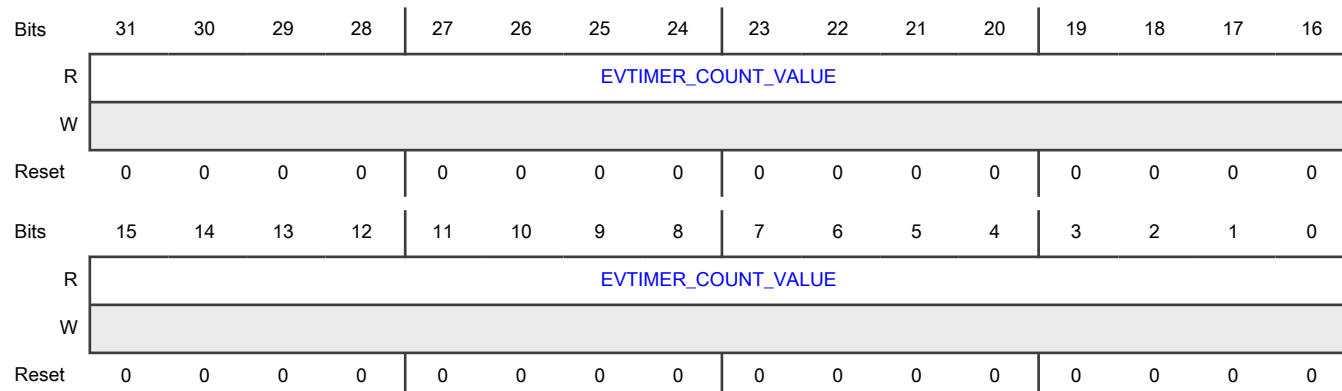
Register	Offset
EVTIMERL	0h

Function

Contains the immediate value of the central EVTimer. This is a Gray encoded counter register that is not synchronized with the bus clock.

Because two 32-bit reads are required to retrieve the entire counter value, it is possible for the counter to roll over between the reads of [EVTIMERL](#) and [EVTIMERH](#). Gray encoding ensures that even if this roll over occurs, the value read represents either the counter value immediately preceding or immediately following the rollover, if the bus clock rate is at least twice the module's clock rate (nominally 1 MHz), or if the bus is clocked by the same 1 MHz LP oscillator that provides the clock to the module.

Diagram



Fields

Field	Function
31-0 EVTIMER_COUNT_VALUE	EVTimer Count Value Specifies the current value of the lower 32 bits of the central 64-bit EVTimer when you read this register. NOTE Only one physical EVTimer exists and is readable from all domains.

31.5.2.3 EVTIMER High (EVTIMERH)**Offset**

Register	Offset
EVTIMERH	4h

Function

Contains the higher value of the central EVTimer. This register resets only after POR or a software reset.

Diagram

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

Fields

Field	Function
31-0 EVTIMER_COUNT_VALUE	EVTimer Count Value Specifies the current value of the upper 32 bits of the central 64-bit EVTimer when you read this register. NOTE Only one physical EVTimer exists and is readable from all domains.

31.5.2.4 Local Capture Low for CPU (CAPTURE_L)

Offset

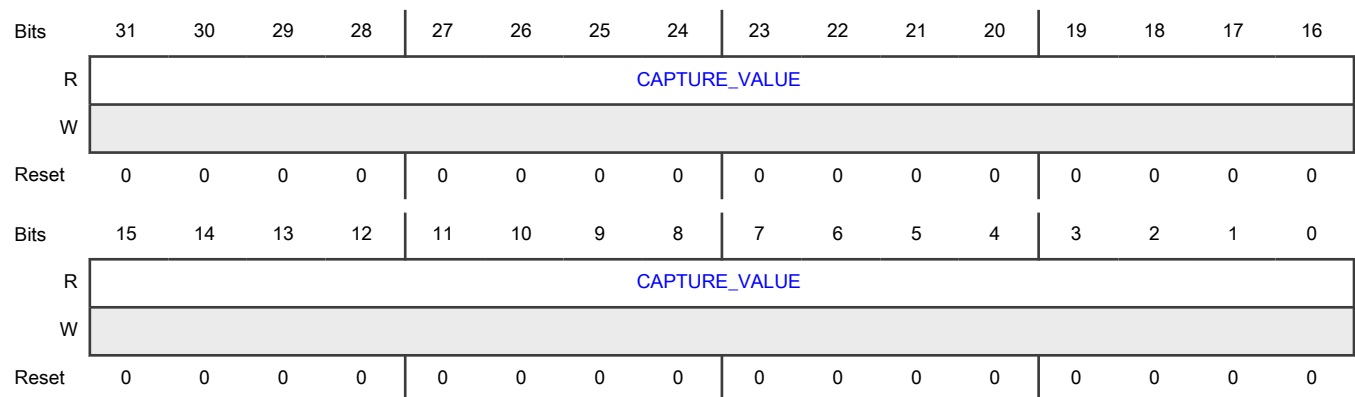
Register	Offset
CAPTURE_L	8h

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. A load is triggered by a load signal from the CPU (for example, an "arm_txev" pulse from the Cortex-M33 core CPU or a "tie_expstate" pulse from DSP). You must not execute a new capture command between reading [CAPTURE_L](#) and [CAPTURE_H](#), to avoid retrieving incoherent results.

For the instance associated with the DSP CPU: If a read of [CAPTURE_L](#) and [CAPTURE_H](#) closely follows the command to implement a capture, then you should insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Diagram



Fields

Field	Function
31-0	EVTimer Capture Value
CAPTURE_VALUE	Specifies the value of the lower 32 bits of the central 64-bit EVTimer when you read this register, at the time that the CPU generated the last capture signal. A separate pair of CAPTURE_L and CAPTURE_H is implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.

31.5.2.5 Local Capture High for CPU (CAPTURE_H)

Offset

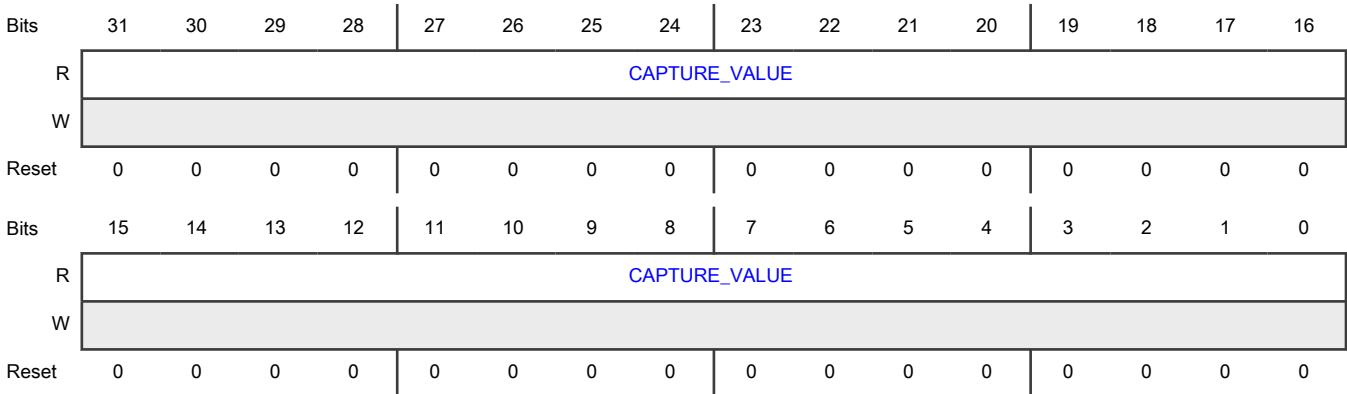
Register	Offset
CAPTURE_H	Ch

Function

Contains the value of the central EVTimer when captured by the CPU. This register is Gray encoded to match EVTimer. A load is triggered by a load signal from the CPU (for example, an "arm_txev" pulse from the Cortex-M33 core CPU or a "tie_expstate" pulse from DSP). You must not execute a new capture command between reading [CAPTURE_L](#) and [CAPTURE_H](#), to avoid retrieving incoherent results.

For the instance associated with the DSP CPU: If a read of [CAPTURE_L](#) and [CAPTURE_H](#) closely follows the command to implement a capture, then you must insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Diagram



Fields

Field	Function
31-0	EVTimer Capture Value
CAPTURE_VAL UE	Specifies the value of the upper 32 bits of the central 64-bit EVTimer when you read this register, at the time that the CPU generated the last capture signal. A separate pair of CAPTURE_L and CAPTURE_H is implemented for each CPU reading its own capture value at the same pair of addresses.

31.5.2.6 Local Match Low for CPU (MATCH_L)

Offset

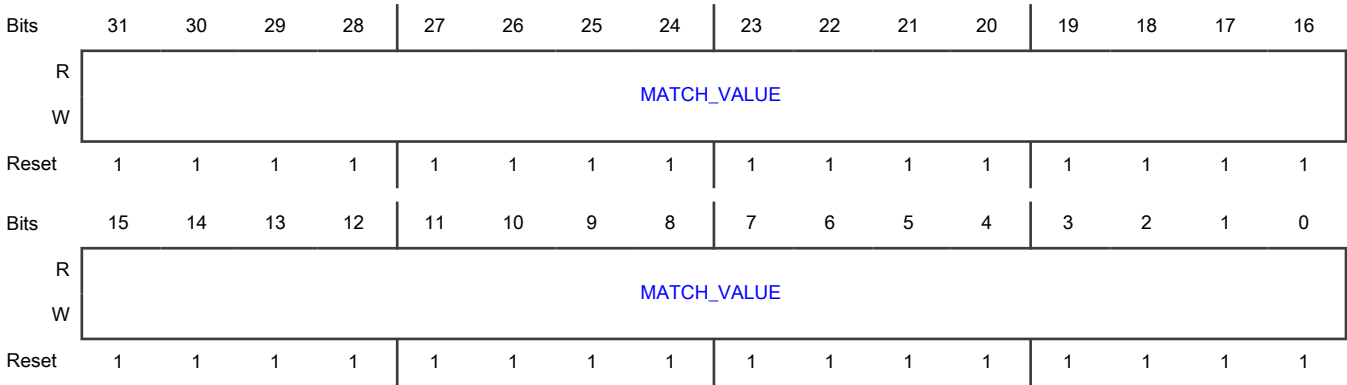
Register	Offset
MATCH_L	10h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to [MATCH_L](#) first, followed by the write to [MATCH_H](#).

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to [MATCH_H](#)). [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) indicates when it is safe to reload the match registers. You do not need to read [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram



Fields

Field	Function
31-0 MATCH_VALU E	EVTimer Match Value Compares the value (lower 32 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled. A separate pair of MATCH_L and MATCH_H is implemented for each CPU reading its own local value at the same pair of addresses.

31.5.2.7 Local Match High for CPU (MATCH_H)

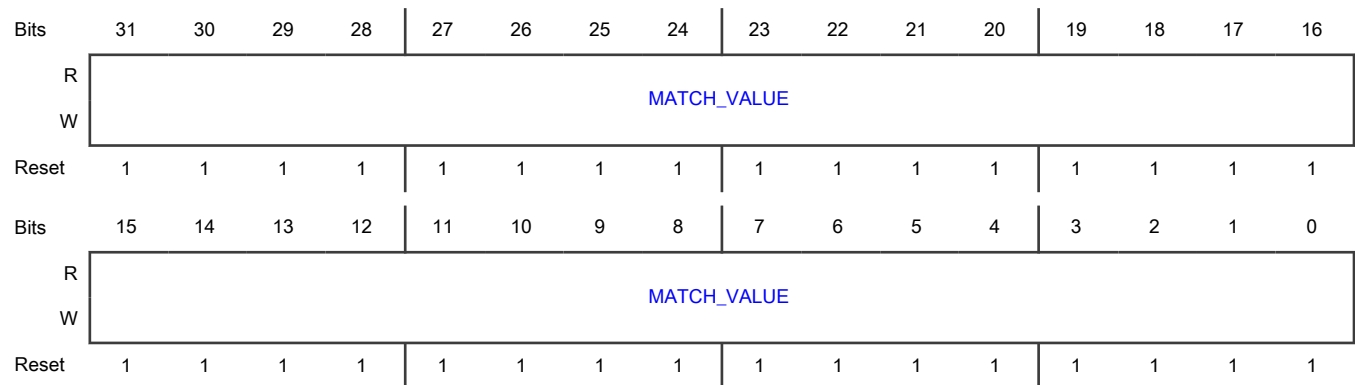
Offset

Register	Offset
MATCH_H	14h

Function

Compares the match value with the central EVTimer value. This register is Gray encoded to match EVTimer. The value in this pair of registers is stable, so there is no need for multiple reads. When writing to the match register pair, you must write to [MATCH_L](#) first, followed by the write to [MATCH_H](#).

You must not initiate a second write to the match register pair until the first write is complete (which is a minimum of three bus clocks followed by a write to [MATCH_H](#)). [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) indicates when it is safe to reload the match registers. You do not need to read [OSEVENT_CTRL\[MATCH_WR_RDY\]](#) if an interrupt has already occurred because of the previous match value, or the required time period has elapsed.

Diagram**Fields**

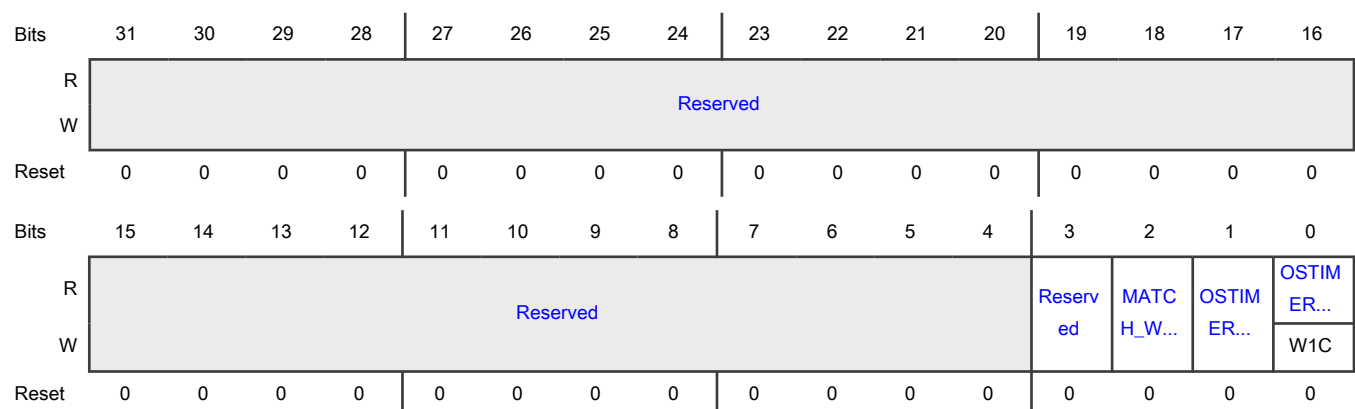
Field	Function
31-0 MATCH_VALUE	EVTimer Match Value Compares the value (upper 32 bits) of the MATCH_L and MATCH_H register pair with the central EVTimer value. If both the values match, the CPU generates an interrupt request if the interrupt is enabled. A separate pair of MATCH_L and MATCH_H is implemented for each CPU reading its own local value at the same pair of addresses.

31.5.2.8 OSTIMER Control for CPU (OSEVENT_CTRL)**Offset**

Register	Offset
OSEVENT_CTRL	1Ch

Function

Provides the interrupt flag and interrupt enable signals for each CPU. A separate OSEVENT_CTRL register is implemented for each CPU. Each CPU reads its own local value at the same address.

Diagram

Fields

Field	Function
31-4 —	Reserved
3 —	Reserved
2 MATCH_WR_RDY	EVTimer Match Write Ready Specifies that when this field is low, it is safe to reload (write) to the match registers. In typical applications, it is not necessary to read this field. The 64-bit match register value is transferred from a pair of shadow registers to the active match registers after you write to MATCH_H. You must not initiate a second write to MATCH_L and MATCH_H until after this transfer completes. This field becomes 0 after the transfer completes. It is not necessary to read this status field, if an interrupt has already occurred because of the first match value, or if it is certain (via some other means) that the required period of time (3 bus clocks) has elapsed.
1 OSTIMER_INT_ENA	Interrupt or Wake-Up Request Enables or blocks the interrupt or wake-up requests depending upon the value of OSEVENT_CTRL[OSTIMER_INTRFLAG]. If this value is 1, then this field asserts an interrupt or wake-up request to the domain processor. Otherwise, the interrupt or wake-up requests are blocked. 0b - Interrupts blocked 1b - Interrupts enabled
0 OSTIMER_INT_RFLAG	Interrupt Flag Sets an interrupt flag when a match occurs between the central 64-bit EVTimer and the value programmed in the MATCH_L and MATCH_H register pair for the associated CPU. Writes to clear this field are asynchronous. You must write 1 to this field before writing a new match value into MATCH_L and MATCH_H.

Chapter 32

SCTIMER

32.1 Chip-specific SCTimer information

Table 395. Reference links to related information

Topic	Related module	Reference
Full description	SCTIMER	SCTIMER
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

32.1.1 Module instances

This device contains one instance of the SCTIMER, SCT0.

32.1.2 Chip-specific signals

See the IOPCTL registers to assign the SCT functions to external pins.

SCT input can come from numerous places and are selected by the SCT Input Mux (see the Peripheral Input Mux Controller (INPUTMUX) chapter). SCT outputs can be routed to multiple places and can be connected (for example) to both a pin and an ADC trigger at the same time.

Table 396. SCT0 signals (inputs)

Type	Function	Connect to	Use register
external from pin	SCT0_GPI0	PIO0_5, PIO0_11, PIO0_14, PIO0_31, PIO1_18, PIO1_30	IOPCTL register for the related pin
	SCT0_GPI1	PIO0_6, PIO0_15, PIO1_0, PIO1_20, PIO1_31	
	SCT0_GPI2	PIO0_12, PIO0_16, PIO0_24, PIO1_1, PIO1_22, PIO2_0	
	SCT0_GPI3	PIO0_13, PIO0_17, PIO1_2, PIO1_24, PIO2_1	
	SCT0_GPI4	PIO0_7, PIO0_19, PIO1_6, PIO1_26, PIO2_6	
	SCT0_GPI5	PIO0_8, PIO0_20, PIO0_21, PIO1_7, PIO1_28, PIO2_7	
	SCT0_GPI6	PIO0_9, PIO0_18, PIO0_22, PIO0_26, PIO1_8, PIO2_10	
	SCT0_GPI7	PIO0_10, PIO0_23, PIO0_27, PIO1_9, PIO2_11	

Table 397. SCT0 signals (outputs)

Type	Function	Connect to	Use register
external to pin	SCT0_OUT0	PIO0_5, PIO0_14, PIO1_19, PIO2_2, PIO2_29	IOPCTL register for the related pin
	SCT0_OUT1	PIO0_6, PIO0_15, PIO1_21, PIO2_3	
	SCT0_OUT2	PIO0_12, PIO0_16, PIO1_23, PIO2_4	
	SCT0_OUT3	PIO0_13, PIO0_17, PIO1_25, PIO2_5, PIO2_30	
	SCT0_OUT4	PIO0_7, PIO0_19, PIO1_6, PIO1_27, PIO2_8	
	SCT0_OUT5	PIO0_8, PIO0_21, PIO1_7, PIO1_29, PIO2_9	
	SCT0_OUT6	PIO0_9, PIO0_18, PIO0_22, PIO0_31	
	SCT0_OUT7	PIO0_10, PIO1_0, PIO2_31	
	SCT0_OUT8	PIO0_11, PIO0_23, PIO2_14	
	SCT0_OUT9	PIO0_24, PIO2_15	

Table 398. Suggested SCT input pin settings

IOPCTL bit(s)	Type D pin
10	I2CFILTER: Set to 1
9	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.
7	DIGIMODE: Set to 1.
6	INVERT: Set to 0.
5	I2CSLEW: Set to 1.
4:3	Not used, set to 0.
2:0	FUNC: not used, set to 0. Specific pin inputs are directly connected to the SCT.

Table 399. Suggested SCT output pin settings

IOPCTL bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1

Table continues on the next page...

Table 399. Suggested SCT output pin settings (continued)

IOPCTL bit(s)	Type D pin	Type A pin	Type I pin
9	SLEW: Set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.
8	FILTEROFF: Set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
4:3	MODE: set to 0.	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for an SCT output.	A reasonable choice for an SCT output.	Not recommended for SCT outputs.

32.2 Overview

SCT (or PWM) is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention. It can operate as a single 32-bit counter or as two independent, 16-bit counters in Unidirectional or Bidirectional modes. Access to 16-bit registers, if you configure SCT as two 16-bit counters, has some limitations (see [Using SCT as two 16-bit counters](#)).

As with most timers, SCT supports:

- A selection of match registers against which you can compare the count value
- Capture registers where you can record the current count value if a predefined condition is detected

An additional feature contributing to the versatility of SCT is the concept of events. This module supports multiple separate events that you can define based on some combination of parameters including:

- A match on one of the match registers, and/or
- A transition on one of the SCT inputs or outputs
- The direction of count

Every action that SCT performs occurs in direct response to the events that you define, without any software overhead. You can enable an event to:

- Start, stop, or halt the counter
- Limit the counter, which refers to clearing the counter in Unidirectional mode, or changing the counter direction in Bidirectional mode
- Set, clear, or toggle any SCT output
- Force a capture of the count value into any of the capture registers
- Generate an interrupt or DMA request

SCT also allows you to group and filter events. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven and can therefore happen without your intervention. By defining these states, SCT or PWM provides the means to run state machines in hardware to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer or counter with capture and match capabilities, all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the state that does not change describes the system. This is the default configuration of SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. You can use an event that is enabled in both states to move from one state to the other and back while filtering out events in either of the states. In such a two-state system, you can create different waveforms at the SCT output, depending on the event history. Change between states is event-driven and happens without any CPU intervention.

Formally, you can program SCT or PWM as a state machine generator. With the unique ability to perform switching between groups of events, you can use SCT as a highly complex state machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides you with an extremely powerful control tool—particularly when the SCT inputs and outputs are connected to other on-chip resources.

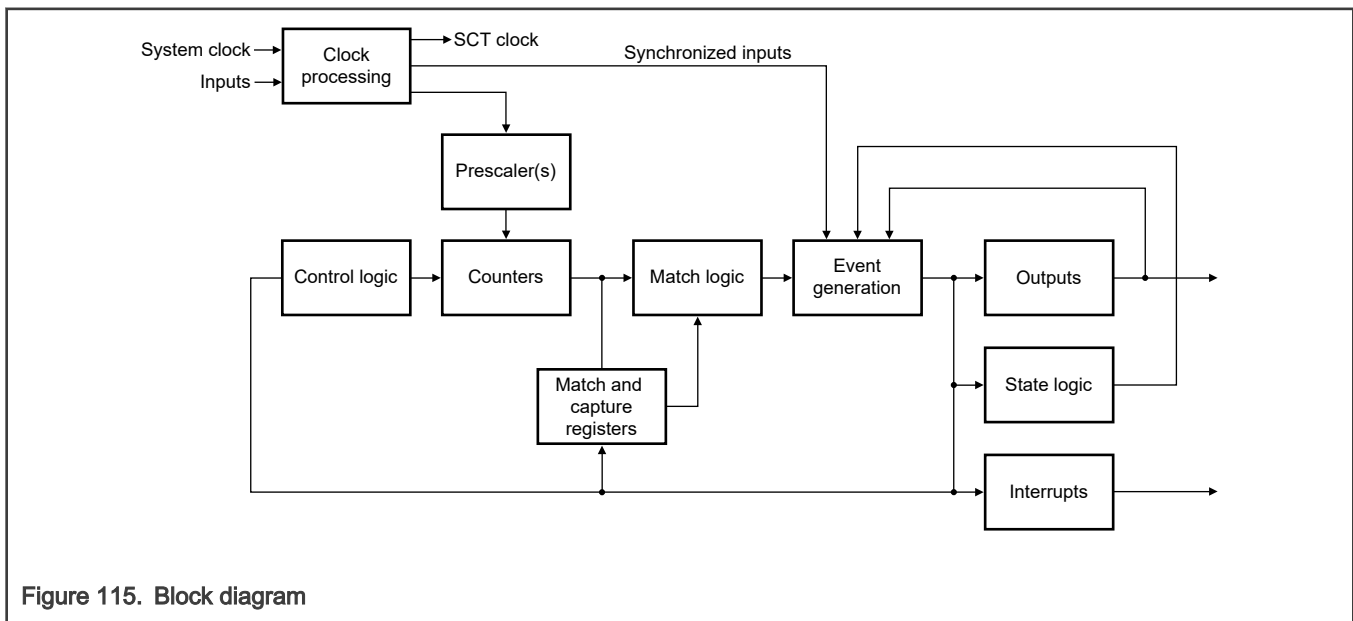
In addition to events and states, SCT (or PWM) provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode
- Selection of any SCT input as a clock source or a clock gate
- Capability of selecting a greater-than-or-equal-to match condition for the purpose of event generation.

NOTE

In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

32.2.1 Block diagram



32.2.2 Features

SCT (or PWM) supports:

- 8 inputs
- 10 outputs
- 16 Match and capture registers
- 16 events
- 16 states

Counter or timer features:

- Configurable as two 16-bit counters or one 32-bit counter (this configuration is possible for each SCT or PWM)
- Counters clocked by system clock or selected input
- Configurable as up counters or up-down counters
- Configurable number of match and capture registers
- Loadable counter value into capture registers, triggered by a match or I/O toggle
- Event creation possible for the following timer events for match and/or I/O transitions:
 - Interrupt
 - Stop
 - Limit
 - Halt
 - Change counting direction
 - Toggle outputs
 - Change the state

PWM features:

- Ability to use counters with match registers to toggle outputs and create time-proportioned PWM signals (PWM waveforms can change based on the current state)
- Up to eight single-edge or four dual-edge PWM outputs with an independent duty cycle and a common PWM cycle length

Event creation features:

- Ability to enable events in Bidirectional mode, based on the count direction. The following conditions define an event:
 - A counter match condition
 - An input (or output) condition such as a rising or falling edge or level
 - A combination of match and/or I/O conditions
- Ability for selected events to limit, halt, start, or stop a counter or change direction
- Ability for events to trigger state changes, output transitions, timer captures, interrupts, and DMA transactions
- Ability to use Match register 0 as an automatic limit
- Ability to hold match events until another qualifying event occurs

State control features:

- A state is defined by events that can happen in the state while the counter is running.
- A state transitions into another state as a result of an event.
- Each event can be assigned to one or more states.
- State variables allow sequencing across multiple counter cycles.

32.3 Functional description

32.3.1 SCT operation

In its simplest, single-state configuration, SCT operates as an event-controlled single- or bidirectional counter.

You can configure events as:

- Counter match events

- An input or output level
- Transitions on an input or output pin
- Combination of match and I/O behavior

In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. You can perform multiple, simultaneous actions for each event. Furthermore, any number of events can trigger a specific SCT action.

A single or multiple SCT actions uniquely define an event. Events that are enabled to trigger an SCT action or actions at any stage of the counter define a state. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that SCT can perform when an event occurs. If you configure an event to change the state, the new state defines a new set of events resulting in different SCT actions. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event-controlled transitions on the SCT outputs and/or interrupts.

After configuration, SCT can run continuously, without software intervention, and can generate multiple output patterns entirely under the control of events. See the following sections to perform various SCT actions:

- [Configuring SCT for multiple events and states](#) to configure SCT
- [Running SCT](#) to start, run, and stop SCT
- [Configuring SCT without using states](#) to configure SCT as simple, event-controlled counter or timer

32.3.2 Using SCT as two 16-bit counters

It is important to note that the SCT implementation on this chip has a limitation. It is not possible to do halfword writes to the upper half of registers, such as, CTRL_H. For the most part, you can deal with it by reading the entire word register, making changes to the upper halfword, and writing back the entire value. For registers CTRL_H, [STATE\[STATE_H\]](#), and [MATCH\[MATCH_H\]](#), this can only be done if both halves of the timer are in Halt mode (halted) (see [CONFLAG\[BUSERRH\]](#)).

32.3.3 Configuring SCT without using states

SCT can function as a standard counter or timer with external capture inputs and match outputs, without using the state logic. To use SCT without states, configure SCT as follows:

1. Write 0 to [State \(STATE\)](#) (0 is the default).
2. Write 0 to [EV_n_CTRL\[STATELD\]](#) and [EV_n_CTRL\[STATEV\]](#) for each event.
3. Write 0x1 to [EV_n_STATE\[STATEMSK_n\]](#) for each event. This enables the event.

In effect, the event occurs in a single state that never changes when the counter is running.

32.3.4 Configuring SCT for multiple events and states

32.3.4.1 Configure the counter

1. Configure the L and H counters in [SCTimer Configuration \(CONFIG\)](#) by selecting two independent 16-bit counters (L and H) or one combined 32-bit counter in [CONFIG\[UNIFY\]](#).
2. Select the SCT clock source using [CONFIG\[CLKMODE\]](#) and [CONFIG\[CKSEL\]](#) from any of the inputs or an internal clock.

32.3.4.2 Configure the match and capture registers

1. Select the number of match and capture registers that the application uses, by using [Match/Capture Mode \(REGMODE\)](#). For each of the match and capture register pairs, you can select whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:

- Each match register ([Match Value \(MATCH0 - MATCH15\)](#)) sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
- Each match reload register ([Match Reload Value \(MATCHRELO - MATCHREL15\)](#)) sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

32.3.4.3 Configure events and event responses

State variables allow SCT control across more than one cycle of the counter. Counter matches, I/O edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values. Perform this procedure to configure events and event responses:

1. Define what triggers each event, by using [Event n Control \(EV0_CTRL - EV15_CTRL\)](#) (one register per event):
 - Using [EVn_CTRL\[COMBMODE\]](#), select whether the event occurs:
 - On change of an input or output
 - On an input or output level
 - On a match condition of the counter
 - Or a combination of match and I/O conditions in field
 - For a match condition:
 - a. Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in [EVn_CTRL\[MATCHSEL\]](#).
 - b. Define, by using [EVn_CTRL\[HEVENT\]](#), whether the event occurs on matching the L or the H counter (when using L and H counters).
 - For an SCT input or output level or transition:
 - a. Select the input number or the output number that is associated with this event, using [EVn_CTRL\[IOSEL\]](#) and [EVn_CTRL\[OUTSEL\]](#).
 - b. Define how the selected input or output triggers the event (edge- or level-sensitive) using [EVn_CTRL\[IOCOND\]](#).
2. Define the effect of each event on SCT outputs, by using [Output n Set \(OUT0_SET - OUT9_SET\)](#) or [Output n Clear \(OUT0_CLR - OUT9_CLR\)](#) (up to the maximum number of outputs on this chip, one register per output): For each SCT output, select the events that set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:
 - a. Write 1 to the corresponding [LIMIT\[LIMMSK_H\]](#) or [LIMIT\[LIMMSK_L\]](#) for the event to set an upper limit for the counter.
 - When a limit event occurs in Unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.
 - When a limit event occurs in Bidirectional mode, the counter begins to count down from the current value on the next clock edge.
 - b. Write 1 to the corresponding [HALT\[HALTMSK_H\]](#) or [HALT\[HALTMSK_L\]](#) for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. You can restore the counter operation by writing 0 [CTRL\[HALT_H\]](#) and/or [CTRL\[HALT_L\]](#).
 - c. Write 1 to the corresponding [STOP\[STOPMSK_H\]](#) or [STOP\[STOPMSK_L\]](#) for the event to stop the counter. If the counter is stopped, it stops counting. However, if you configure an event as a transition on an I/O, it can restart the counter.
 - d. Write 1 to the corresponding [START\[STARTMSK_H\]](#) or [START\[STARTMSK_L\]](#) for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.

- Define the events that contribute to the SCT interrupt: write 1 to the corresponding [EVEN\[*IE**n*\]](#) and [EVFLAG\[*FLAG**n*\]](#) to enable the event to contribute to the SCT interrupt.

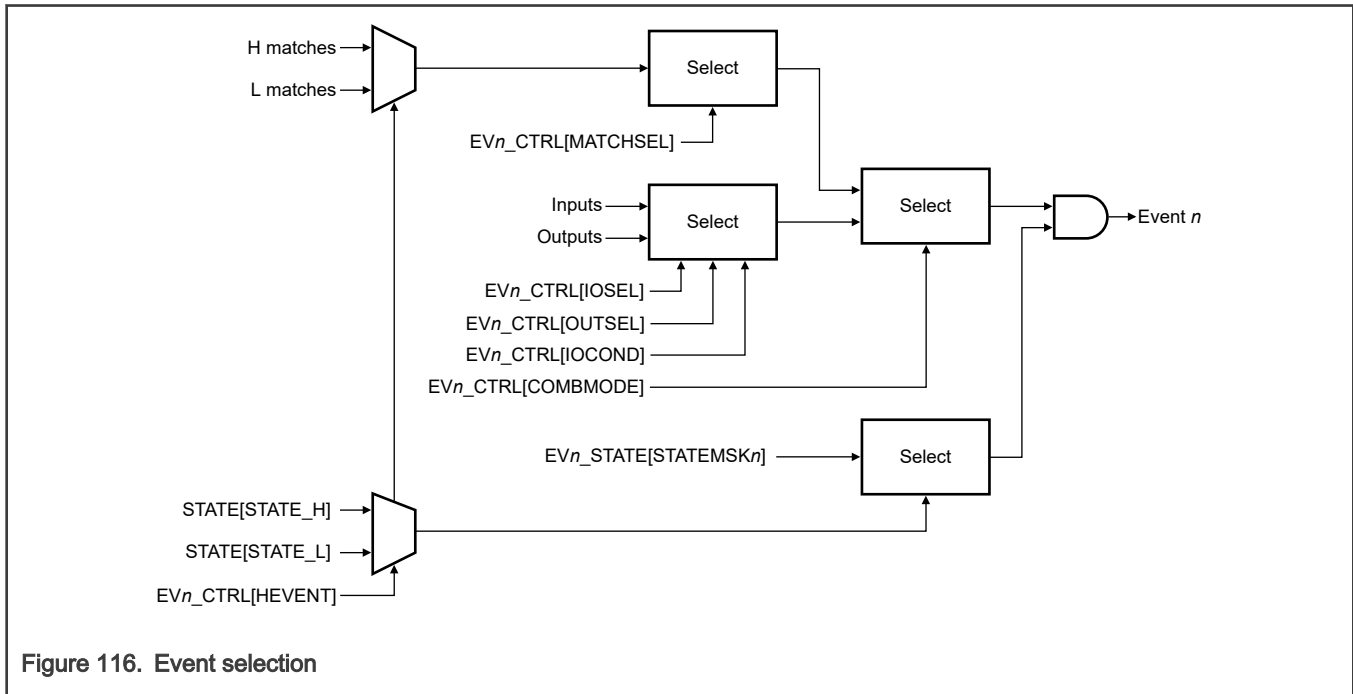


Figure 116. Event selection

32.3.4.4 Configure multiple states

- Select the state or states (up to 2), by using [Event *n* State \(EV0_STATE - EV15_STATE\)](#), for each event (up to the maximum number of events on this chip—one register per event), in which this event is allowed to occur. You can select each state for more than one event.
- Determine how the event affects the system state: in [Event *n* Control \(EV0_CTRL - EV15_CTRL\)](#) (up to the maximum number of events on this chip, one register per event), set the new state value in [EV_{*n*}_CTRL\[STATEV\]](#) for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the configuration of [EV_{*n*}_CTRL\[STATELD\]](#).

If [EV_{*n*}_CTRL\[STATEV\]](#) and [EV_{*n*}_CTRL\[STATELD\]](#) are 0, the state does not change.

NOTE

If there are higher numbered events in the current state, this event cannot change the state.

32.3.4.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. You can program each capture register to capture the counter contents when one or more events occur.
- If the counter is in Bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down, by writing to [Output Counter Direction Control \(OUTPUTDIRCTRL\)](#).

32.3.5 State logic

You can configure SCT as a timer or counter with multiple programmable states, which you can define through the events that can be captured in each particular state. A multi-state SCT can change from one state to another when an event that you define triggers a state change, which is triggered through each event's [Event *n* Control \(EV0_CTRL - EV15_CTRL\)](#) in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, SCT enters a locked state where all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event that moves SCT into a locked state, occurs. Later, while SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number.

If SCT registers an event that forces an abort, putting SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Because multiple states (any state number between the maximum implemented state and 31) are locked states, you can define multiple abort or error events, each incrementing the state number by a different value.

32.3.6 Clearing the prescaler

When enabled by a nonzero [CTRL\[PRE_L\]](#) and/or [CTRL\[PRE_H\]](#), the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- When you write to [Counter \(COUNT\)](#)
- When you write 1 to [CTRL\[CLRCTR\]](#)
- An event selected by a 1 in the counter limit register when [CTRL\[BIDIR\] = 0](#)

When [CTRL\[BIDIR\] = 0](#), a limit event caused by an I/O signal can clear a nonzero prescaler. However, a limit event caused by a Match clears only a nonzero prescaler in a special case, as described in [Match vs. I/O events](#).

A limit event, when [CTRL\[BIDIR\] = 1](#), does not clear the prescaler. Instead, it clears [CTRL\[DOWN\]](#) and decrements the counter on the same clock if the counter is enabled in that clock.

32.3.7 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, you can enable the prescaler and counter to count only when a selected edge is detected on a clock input:

- The prescaler is enabled when the clock mode is not 01, or when the input edge that [CONFIG\[CKSEL\]](#) selects is detected.
- The counter is enabled when the prescaler is enabled, and ([PRELIM = 0](#) or the prescaler is equal to the value of [PRELIM](#)).

An I/O component of an event can occur in any SCT clock when its counter, [CTRL\[HALT\]](#), is 0. In general, a Match component of an event can only occur in an SCT clock when its counter fields, [CTRL\[HALT\]](#) and [CTRL\[STOP\]](#), are both 0 and the counter is enabled.

The following table shows when the various kinds of events can occur.

Table 400. Event conditions

EV_n_CTRL[C OMBMODE]	EV_n_CTRL[I OCOND]	Event can occur on clock:
I/O	Any	Whenever CTRL[HALT] = 0 .
MATCH	Any	When CTRL[HALT] = 0 , CTRL[STOP] = 0 , and the counter is enabled.
OR	Any	• From the I/O component, whenever CTRL[HALT] = 0. • From the match component, when CTRL[HALT] = 0, CTRL[STOP] = 0, and the counter is enabled.

Table continues on the next page...

Table 400. Event conditions (continued)

EV_n_CTRL[C OMBMODE]	EV_n_CTRL[I OCOND]	Event can occur on clock:
AND	Low or High	When CTRL[HALT] = 0, CTRL[STOP] = 0, and the counter is enabled.
AND	Rise or Fall	Whenever CTRL[HALT] = 0.

32.4 External signals

SCT inputs can come from numerous places and are selected by the SCT input mux.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time.

Table 401. SCT signals

Signal	I/O	Description
SCT_IN _n	I	SCT input. Signals come from GPIO sources, each from a variety of pins selected via IOPCTL muxing.
SCT_OUT _n	O	SCT output. Signals go to IOPCTL muxing where they can be (optionally) output onto a selection of GPIO pins.

32.5 Initialization

To initialize SCT:

1. Enable the clock to SCT (or PWM).
2. Clear the SCT peripheral reset.
3. Configure the interrupts.
4. Connect the SCT outputs to external pins.
5. Configure the DMA request lines to connect to the DMA trigger inputs.

32.6 Application information

32.6.1 Running SCT

1. Initialize SCT (see [Initialization](#)).
2. Write to [State \(STATE\)](#) to define the initial state. By default, the initial state is 0.
3. To start SCT, write to [SCT Control \(CTRL\)](#):
 - a. Clear the counters.
 - b. Write 0 or 1 to [CTRL\[STOP_L\]](#) and/or [CTRL\[STOP_H\]](#).

NOTE

The counter starts counting after CTRL[STOP] becomes 0 as well. If CTRL[STOP] = 1, SCT, instead, waits for an event to occur that you can configure to start the counter.

- c. Select Unidirectional or Bidirectional counting mode ([CTRL\[BIDIR_L\]](#) and/or [CTRL\[BIDIR_H\]](#)) for each counter.
- d. Select the prescale factor for the counter clock ([CTRL\[PRE_L\]](#) or [CTRL\[PRE_H\]](#)).
- e. Write 0 to [CTRL\[HALT_L\]](#) and/or [CTRL\[HALT_H\]](#). By default, the counters halt and no events can occur.

4. Stop or halt the counters at any time by using CTRL[STOP_L] and/or CTRL[STOP_H], or CTRL[HALT_L] and/or CTRL[HALT_H], respectively).
 - After the counters stop, both you and an event configured to write 0 to CTRL[STOP_L] and/or CTRL[STOP_H] can start the counter again.
 - After the counter halts, only if you write 0 to CTRL[HALT_L] and/or CTRL[HALT_H] can start the counter again. No events can occur.
 - After the counters halt, you can set any SCT output as high or low by writing directly to the OUT register.

You can read the current SCT state at any time by using [State \(STATE\)](#).

To change the current state (that is, independently of any event occurring), write 1 to CTRL[HALT] and write to [State \(STATE\)](#) to change the state value. Writing to this register is allowed only when the counter halts (CTRL[HALT_L] and/or CTRL[HALT_H] = 1) and no events can occur.

32.6.2 SCT (or PWM) configuration example

This section discusses SCT configuration by using an example, showing the simple application of SCT with two sets of match events (EV0/1 and EV3/4) to set and clear SCT output 0. The timer automatically resets whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to high and match event EV1 clears output 0.

While monitoring SCT input 0:

- If input 0 is low at the next timer reset (EV2), the state changes to state 1, enabling events EV3 and EV4. This leads to the same output but with different match value triggers.
- If input 0 is high at the next timer reset, the associated event (EV5) causes the state to change back to the state enabling events EV0 and EV1.

The following figure uses this SCT configuration:

- One input
- One output
- Five match registers
- Six events and match 0 used with the auto limit function
- Two states

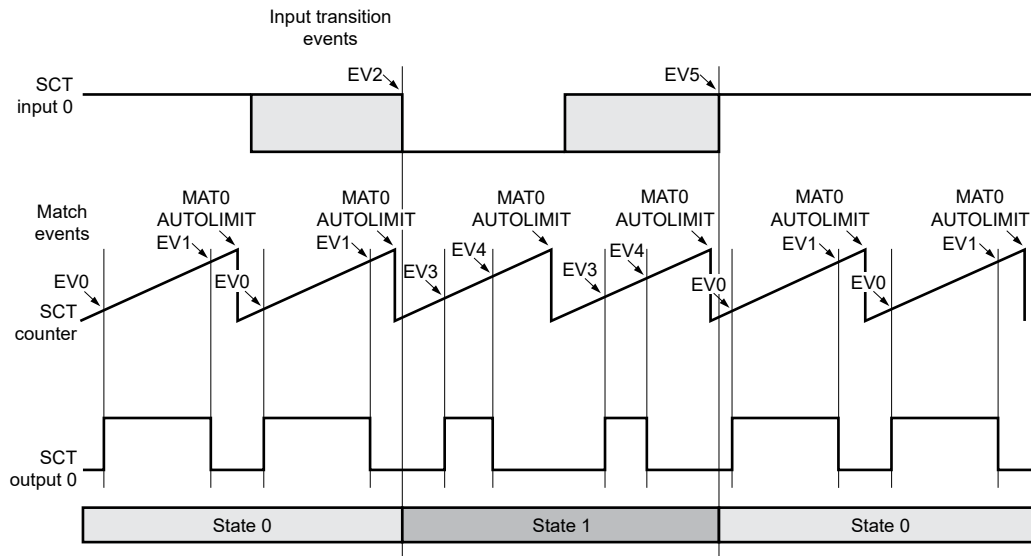


Figure 117. SCT configuration example: pictorial representation

This application of SCT uses the configuration listed in the following table (all register values not listed are their default values):

Table 402. SCT configuration example: settings

Configuration	Register	Setting
Counter	SCTimer Configuration n (CONFIG)	Uses one counter (CONFIG[UNIFY] = 1).
	SCTimer Configuration n (CONFIG)	Enable the auto limit for MAT0 (CONFIG[AUTOLIMIT_L] = 1).
	SCT Control (CTRL)	Uses unidirectional counter (CTRL[BIDIR_L] = 0).
Clock base	SCTimer Configuration n (CONFIG)	Uses default values for clock configuration.
Match and capture registers	Match/Capture Mode (REGMODE)	Configure one match register for each match event by setting bits 0, 1, 2, 3, 4 of REGMODE[REGMOD_Ln] to 0.
Define match values	Match Value (MATCH0 - MATCH15) (0/1/2/3/4)	Set a match value MATCH0/1/2/4_L in each register. The match 0 register serves as an automatic limit event that resets the counter, without using an event. To enable the automatic limit, write 1 to CONFIG[AUTOLIMIT].

Table continues on the next page...

Table 402. SCT configuration example: settings (continued)

Configuration	Register	Setting
Define match reload values	Match Reload Value (MATCHRELO - MATCHREL15) (0/1/2/3/4)	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EVO_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x1. Event 0 uses match condition only. - Set EV_n_CTRL[MATCHSEL] = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x1. Event 1 uses match condition only. - Set EV_n_CTRL[MATCHSEL] = 2. Select match value of match register 2. The match value of MAT2 is associated with event 1.
Define when event 2 occurs	EV2_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x3. Event 2 uses match condition and I/O condition. - Set EV_n_CTRL[IOSEL] = 0. Select input 0. - Set EV_n_CTRL[IOCOND] = 0x0. Input 0 is low. - Set EV_n_CTRL[MATCHSEL] = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set EV_n_CTRL[STATEV] to 1 and EV_n_CTRL[STATELD] to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x1. Event 3 uses match condition only. - Set EV_n_CTRL[MATCHSEL] = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x1. Event 4 uses match condition only. - Set EV_n_CTRL[MATCHSEL] = 0x4. Select match value of match register 4. The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	- Set EV_n_CTRL[COMBMODE] = 0x3. Event 5 uses match condition and I/O condition. - Set EV_n_CTRL[IOSEL] = 0. Select input 0. - Set EV_n_CTRL[IOCOND] = 0x3. Input 0 is high. - Set EV_n_CTRL[MATCHSEL] = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set EV_n_CTRL[STATEV] bits to 0 and EV_n_CTRL[STATELD] to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.

Table continues on the next page...

Table 402. SCT configuration example: settings (continued)

Configuration	Register	Setting
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states to enable event 0	OUT0_STATE	Set STATEMSK0 bit 0 to 1, enabling event 0 in state 0. Set all other bits to 0.
Configure states to enable event 1	EV1_STATE	Set STATEMSK1 bit 0 to 1, enabling event 1 in state 0. Set all other bits to 0.
Configure states to enable event 2	EV2_STATE	Set STATEMSK2 bit 0 to 1, enabling event 2 in state 0. Set all other bits to 0.
Configure states to enable event 3	EV3_STATE	Set STATEMSK3 bit 1 to 1, enabling event 3 in state 1. Set all other bits to 0.
Configure states to enable event 4	EV4_STATE	Set STATEMSK4 bit 1 to 1, enabling event 4 in state 1. Set all other bits to 0.
Configure states to enable event 5	EV5_STATE	Set STATEMSK5 bit 1 to 1, enabling event 5 in state 1. Set all other bits to 0.

32.7 Memory map and register definition

32.7.1 Register configuration

For most of the SCT registers, the register function depends on the setting of certain other register fields:

- [CONFIG\[UNIFY\]](#) determines whether SCT is used as one 32-bit register (for operation as one 32-bit counter or timer) or as two 16-bit counters or timers named L and H.
 - If [CONFIG\[UNIFY\]](#) = 1, only one register is used (for operation as one 32-bit counter or timer).
 - If [CONFIG\[UNIFY\]](#) = 0, you can access the L and H registers using a 32-bit read or write operation or you can read them individually. You can write to the L registers individually, but write to the H register as a word along with the L register.

Typically, you configure [CONFIG\[UNIFY\]](#) by writing to [SCTimer Configuration \(CONFIG\)](#) before accessing any other registers.

- [REGMOD\[REGMOD_H*n*\]](#) and [REGMOD\[REGMOD_L*n*\]](#) determine whether each set of match and capture registers uses the match or capture functionality:
 - [REGMODE*n*](#) = 0: Registers operate as match and reload registers.
 - [REGMODE*n*](#) = 1: Registers operate as capture and capture control registers.

32.7.2 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. The following figure shows the registers and register fields that you must configure for each event.

NOTE

In this figure, letters are used to represent the maximum quantity of certain SCT (or PWM) features:

- i = match/captures
- j = events
- k = states
- p = outputs
- q = inputs

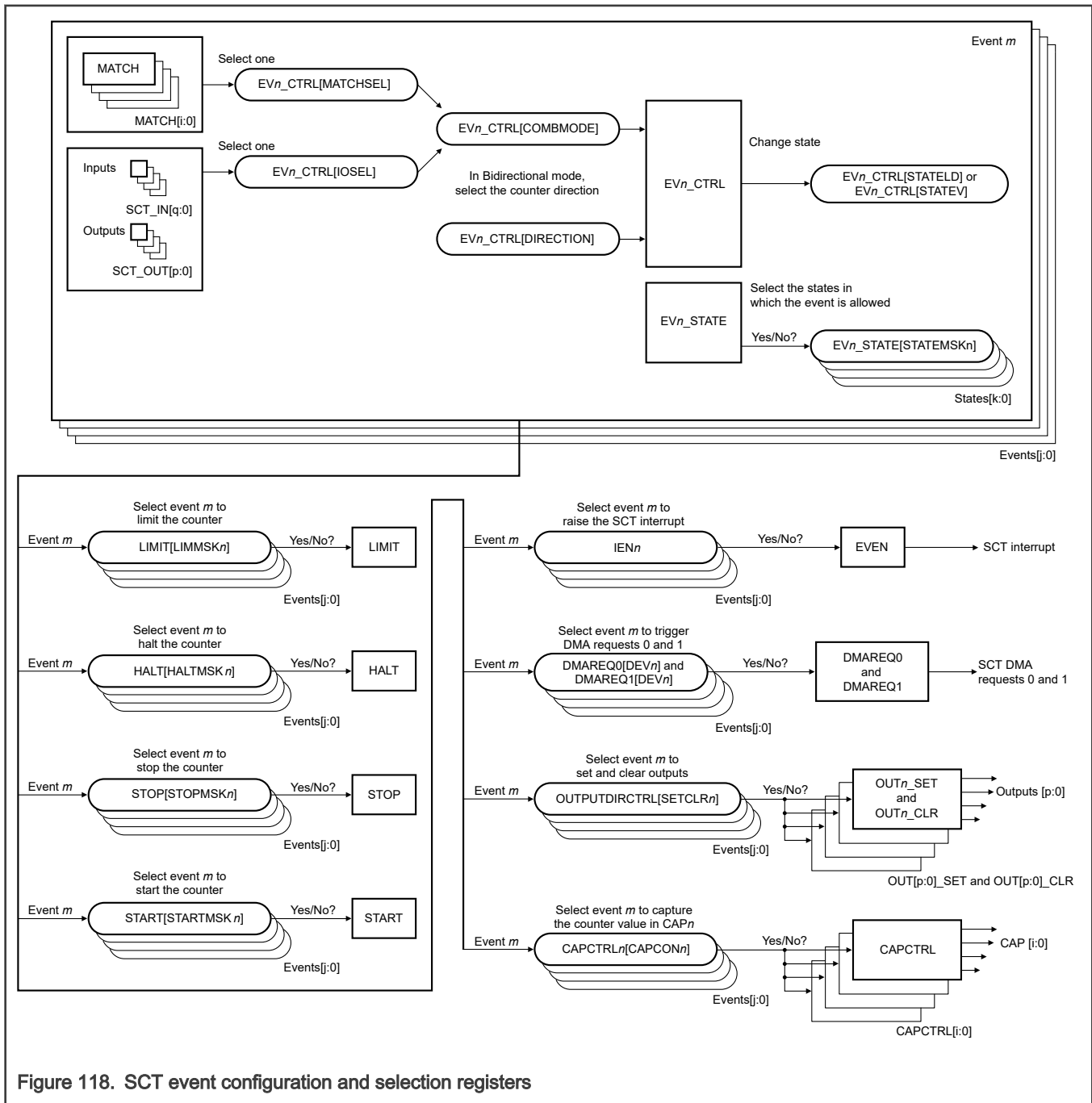


Figure 118. SCT event configuration and selection registers

32.7.2.1 Counter configuration and control registers

SCT contains these registers for configuration, monitoring, and control:

- **SCTimer Configuration (CONFIG)** configures SCT in single, 32-bit counter mode or in dual, 16-bit counter mode. The register helps you perform clocking, clock synchronization, configure automatic limits, and use reload registers.
- **SCT Control (CTRL)** helps you monitor and set the counter direction. Additionally, using this register, you can clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter in dual-counter mode.

32.7.2.2 Event configuration registers

Each event is associated with these registers:

- **Event n State (EV0_STATE - EV15_STATE)** for each event, to enable the event.
- **Event n Control (EV0_CTRL - EV15_CTRL)** for each event, to define what triggers the event.

32.7.2.3 Match and capture registers

SCT includes a set of registers to store the SCT match or capture values:

- Each **Match Value (MATCH0 - MATCH15)** is associated with a corresponding **Match Reload Value (MATCHRELO - MATCHREL15)**, which automatically reloads **Match Value (MATCH0 - MATCH15)** at the beginning of each counter cycle.
- One REGMODE register to configure each Match/Capture register for either storing a match value or a capture value.
- A set of match and capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter capture value.
- One **Capture Control (CAPCTRL0 - CAPCTRL15)** for each **Capture Value (CAP0 - CAP15)**.

Figure 119 shows the match logic for SCT.

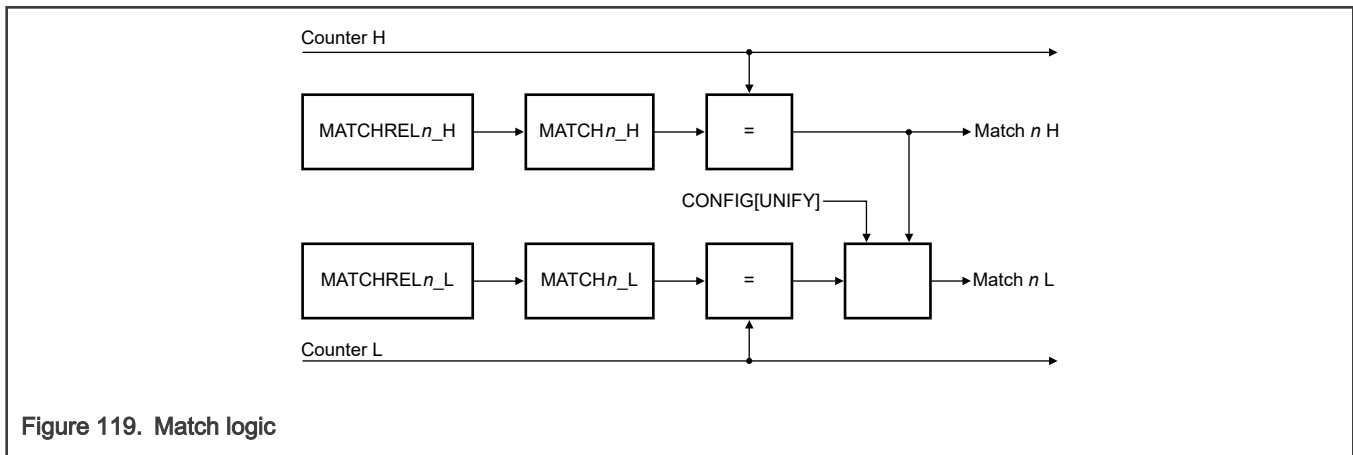
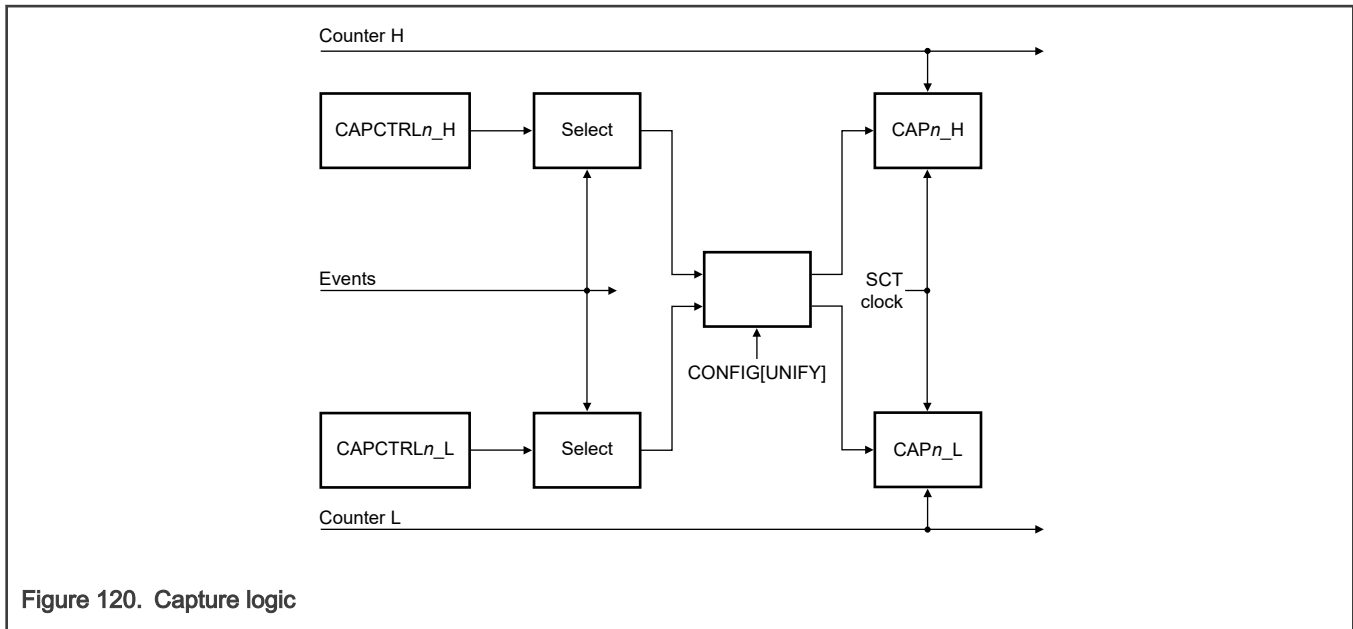


Figure 120 shows the capture logic for SCT.



32.7.2.4 Event select registers for counter operations

[SCT Limit Event Select \(LIMIT\)](#), [Halt Event Select \(HALT\)](#), [Stop Event Select \(STOP\)](#), and [Start Event Select \(START\)](#) help you select which events affect the counter. The counter actions are limit, halt, start, or stop, which apply to the unified counter or to the two 16-bit counters. Also included is the Counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

32.7.2.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One Output n Set register (OUTn_SET) for each output to select the events which set the output.
- One Output n Clear register (OUTn_CLR) for each output to select the events which clear the output.
- The Conflict Resolution register (RES) which defines an action when more than one event try to control an output at the same time.
- The Conflict Flag (CONFLAG) and Conflict Interrupt Enable (CONEN) registers that monitor interrupts arising from output set and clear conflicts.
- The Output Counter Direction Control register (OUTPUTDIRCTRL) that interchanges the set and clear output operation caused by an event in bidirectional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state for a given output n.

In the dual-counter mode, the events can be selected independently for each output.

Output slice n shows one output slice of the SCT.

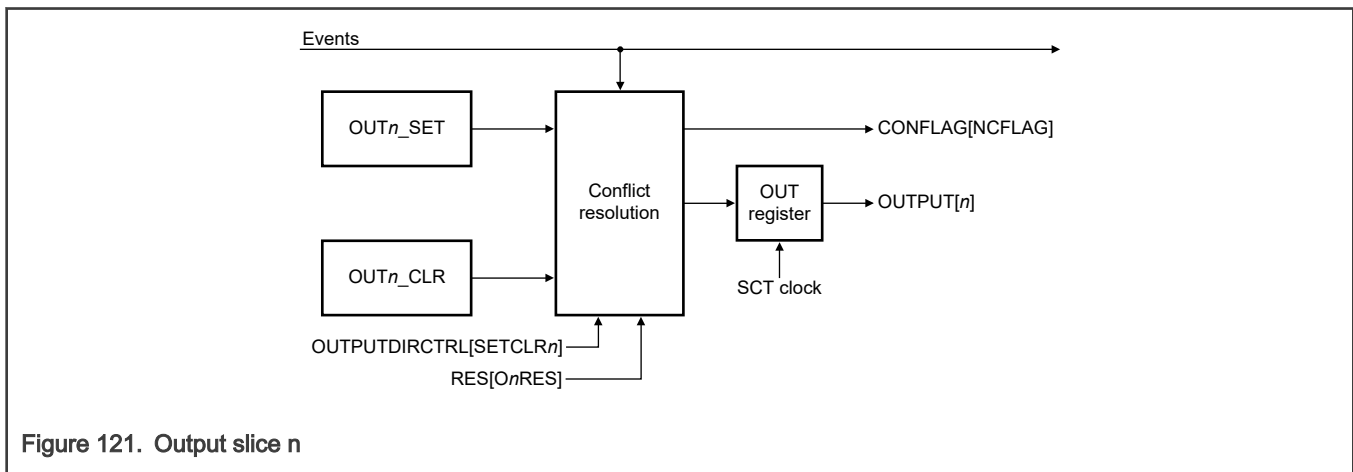


Figure 121. Output slice n

32.7.2.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAPn registers. Each Capture register n has one associated CAPCTRLn register which in turn selects the events to capture the counter value.

32.7.2.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can generate a DMA request.

The DMAREQn register contains one bit for each event that selects whether this event generates a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

32.7.2.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that request an interrupt.

- The Event Flag register (EVFLAG) provides one flag for each event that is set when the event occurs.
- The Event Interrupt Enable register (EVEN) provides one bit for each event to be enabled for the SCT interrupt.

This diagram shows an event triggering the SCT to generate one interrupt.

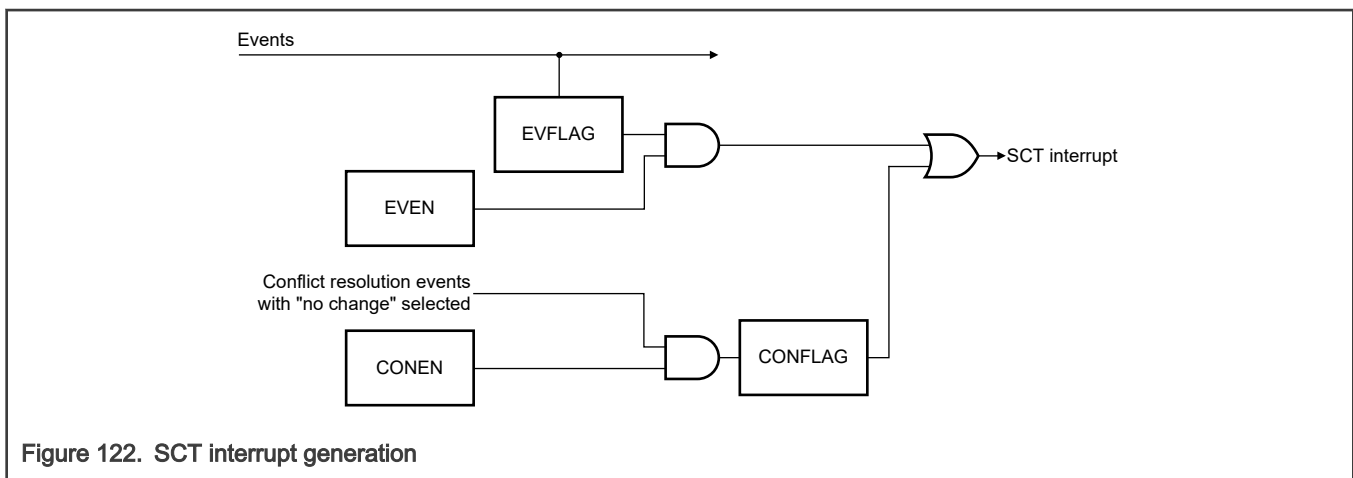


Figure 122. SCT interrupt generation

32.7.2.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT Input register (INPUT) to read the state of any of the SCT inputs
- The SCT Output register (OUTPUT) to set or clear any of the SCT outputs or to read the state of the outputs

32.7.3 SCTimer register descriptions

32.7.3.1 SCT memory map

SCT0 base address: 4014_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SCTimer Configuration (CONFIG)	32	RW	See section
4h	SCT Control (CTRL)	32	RW	See section
8h	SCT Limit Event Select (LIMIT)	32	RW	0000_0000h
Ch	Halt Event Select (HALT)	32	RW	0000_0000h
10h	Stop Event Select (STOP)	32	RW	0000_0000h
14h	Start Event Select (START)	32	RW	0000_0000h
40h	Counter (COUNT)	32	RW	0000_0000h
44h	State (STATE)	32	RW	See section
48h	Input (INPUT)	32	R	0000_0000h
4Ch	Match/Capture Mode (REGMODE)	32	RW	0000_0000h
50h	Output (OUTPUT)	32	RW	See section
54h	Output Counter Direction Control (OUTPUTDIRCTRL)	32	RW	0000_0000h
58h	Output Conflict Resolution (RES)	32	RW	0000_0000h
5Ch	DMA Request 0 (DMAREQ0)	32	RW	See section
60h	DMA Request 1 (DMAREQ1)	32	RW	See section
F0h	Event Interrupt Enable (EVEN)	32	RW	See section
F4h	Event Flag (EVFLAG)	32	RW	See section
F8h	Conflict Interrupt Enable (CONEN)	32	RW	See section
FCh	Conflict Flag (CONFLAG)	32	RW	See section
100h - 13Ch	Capture Value (CAP0 - CAP15)	32	RW	0000_0000h
100h - 13Ch	Match Value (MATCH0 - MATCH15)	32	RW	0000_0000h
200h - 23Ch	Capture Control (CAPCTRL0 - CAPCTRL15)	32	RW	0000_0000h
200h - 23Ch	Match Reload Value (MATCHREL0 - MATCHREL15)	32	RW	0000_0000h
300h	Event n State (EV0_STATE)	32	RW	See section
304h	Event n Control (EV0_CTRL)	32	RW	See section
308h	Event n State (EV1_STATE)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
30Ch	Event n Control (EV1_CTRL)	32	RW	See section
310h	Event n State (EV2_STATE)	32	RW	See section
314h	Event n Control (EV2_CTRL)	32	RW	See section
318h	Event n State (EV3_STATE)	32	RW	See section
31Ch	Event n Control (EV3_CTRL)	32	RW	See section
320h	Event n State (EV4_STATE)	32	RW	See section
324h	Event n Control (EV4_CTRL)	32	RW	See section
328h	Event n State (EV5_STATE)	32	RW	See section
32Ch	Event n Control (EV5_CTRL)	32	RW	See section
330h	Event n State (EV6_STATE)	32	RW	See section
334h	Event n Control (EV6_CTRL)	32	RW	See section
338h	Event n State (EV7_STATE)	32	RW	See section
33Ch	Event n Control (EV7_CTRL)	32	RW	See section
340h	Event n State (EV8_STATE)	32	RW	See section
344h	Event n Control (EV8_CTRL)	32	RW	See section
348h	Event n State (EV9_STATE)	32	RW	See section
34Ch	Event n Control (EV9_CTRL)	32	RW	See section
350h	Event n State (EV10_STATE)	32	RW	See section
354h	Event n Control (EV10_CTRL)	32	RW	See section
358h	Event n State (EV11_STATE)	32	RW	See section
35Ch	Event n Control (EV11_CTRL)	32	RW	See section
360h	Event n State (EV12_STATE)	32	RW	See section
364h	Event n Control (EV12_CTRL)	32	RW	See section
368h	Event n State (EV13_STATE)	32	RW	See section
36Ch	Event n Control (EV13_CTRL)	32	RW	See section
370h	Event n State (EV14_STATE)	32	RW	See section
374h	Event n Control (EV14_CTRL)	32	RW	See section
378h	Event n State (EV15_STATE)	32	RW	See section
37Ch	Event n Control (EV15_CTRL)	32	RW	See section
500h	Output n Set (OUT0_SET)	32	RW	See section
504h	Output n Clear (OUT0_CLR)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
508h	Output n Set (OUT1_SET)	32	RW	See section
50Ch	Output n Clear (OUT1_CLR)	32	RW	See section
510h	Output n Set (OUT2_SET)	32	RW	See section
514h	Output n Clear (OUT2_CLR)	32	RW	See section
518h	Output n Set (OUT3_SET)	32	RW	See section
51Ch	Output n Clear (OUT3_CLR)	32	RW	See section
520h	Output n Set (OUT4_SET)	32	RW	See section
524h	Output n Clear (OUT4_CLR)	32	RW	See section
528h	Output n Set (OUT5_SET)	32	RW	See section
52Ch	Output n Clear (OUT5_CLR)	32	RW	See section
530h	Output n Set (OUT6_SET)	32	RW	See section
534h	Output n Clear (OUT6_CLR)	32	RW	See section
538h	Output n Set (OUT7_SET)	32	RW	See section
53Ch	Output n Clear (OUT7_CLR)	32	RW	See section
540h	Output n Set (OUT8_SET)	32	RW	See section
544h	Output n Clear (OUT8_CLR)	32	RW	See section
548h	Output n Set (OUT9_SET)	32	RW	See section
54Ch	Output n Clear (OUT9_CLR)	32	RW	See section

32.7.3.2 SCTimer Configuration (CONFIG)

Offset

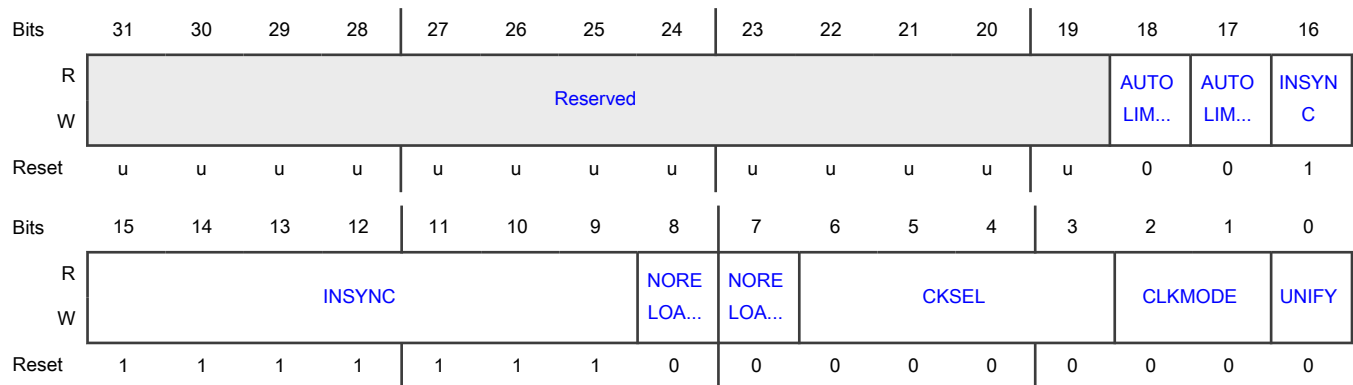
Register	Offset
CONFIG	0h

Function

Configures the overall operation of the SCT. Write to the CONFIG register before writing to any other registers.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-19 —	Reserved
18 AUTOLIMIT_H	Auto Limit Higher A one in this bit will cause a match on match register 0 to be treated as the LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set. 0b - Disable. 1b - Enable. A match on match register 0 is the LIMIT condition. No need to define an associated event.
17 AUTOLIMIT_L	Auto Limit Lower As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. AUTOLIMIT_L applies to both the higher and lower registers when the UNIFY bit is set. 0b - Disable. 1b - Enable. A match on match register 0 is the LIMIT condition. No need to define an associated event.
16-9 INSYNC	Input Synchronization Synchronization for input N (bit 9 = input 0, bit 10 = input 1, ... bit 16 = input 7). A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is known to already be synchronous with the SCT clock, INSYNC may be set to 0 for faster input response.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The SCT clock is the system clock for CKMODEs 0-2. It is the selected, asynchronous SCT input clock for CKMODE3. The INSYNC field only affects inputs used for event generation. INSYNC does not apply to the clock input specified in the CKSEL field.
8 NORELOAD_H	No Reload Higher Match Setting NORELOAD_H eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY = 1. 0b - Reload. The default setting. 1b - No Reload. Prevents the higher match registers from being reloaded from their respective reload registers.
7 NORELOAD_L	No Reload Lower Match Setting NORELOAD_L eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear NORELOAD_L at any time. This bit applies to both the higher and lower registers when the UNIFY = 1. 0b - Reload. The default setting. 1b - No Reload. Prevents the lower match registers from being reloaded from their respective reload registers.
6-3 CKSEL	SCT Clock Select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register. 0000b - Rising edges on input 0 0001b - Falling edges on input 0 0010b - Rising edges on input 1 0011b - Falling edges on input 1 0100b - Rising edges on input 2 0101b - Falling edges on input 2 0110b - Rising edges on input 3 0111b - Falling edges on input 3 1000b - Rising edges on input 4 1001b - Falling edges on input 4 1010b - Rising edges on input 5 1011b - Falling edges on input 5 1100b - Rising edges on input 6 1101b - Falling edges on input 6

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1110b - Rising edges on input 7 1111b - Falling edges on input 7
2-1 CLKMODE	SCT Clock Mode 00b - System Clock Mode. The system clock clocks the entire SCT module including all counters and counter prescalers. 01b - Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode. 10b - SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including all counters and prescalers, after first being synchronized to the system clock. The minimum pulse width on the clock input is 1 bus clock period. This mode is the low-power, sampled-clock mode. 11b - Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.
0 UNIFY	SCT Operation The UNIFY bit determines whether the SCT is used as a single 32-bit counter/timer or as two independent 16-bit counter/timers named L or H. Choose the setting for the UNIFY bit before accessing other registers. 0b - Dual counter. The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H. 1b - Unified counter. The SCT operates as a unified 32-bit counter.

32.7.3.3 SCT Control (CTRL)

Offset

Register	Offset
CTRL	4h

Function

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the CTRL register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All bits in CTRL can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

NOTE

If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

NOTE

If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved				PRE_H								BIDIR_H	CLRC_TR_H	HALT_H	STOP_H	DOWN_H
W																	
Reset	u	u	u	0	0	0	0	0	0	0	0	0	0	1	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserved				PRE_L								BIDIR_L	CLRC_TR_L	HALT_L	STOP_L	DOWN_L
W																	
Reset	u	u	u	0	0	0	0	0	0	0	0	0	0	1	0	0	

Fields

Field	Function
31-29 —	Reserved
28-21 PRE_H	Prescaler for High Counter Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PREL_H + 1. NOTE Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.
20 BIDIR_H	Bidirectional Select High High direction select 0b - Up. The H counter counts up to its limit condition, then is cleared to zero. 1b - Up-down. The H counter counts up to its limit, then counts down to a limit condition or to 0.
19 CLRCTR_H	Clear Counter High Writing a 1 to CLRCTR_H clears the H counter. CLRCTR_H always reads as 0.
18 HALT_H	Halt Counter High

Table continues on the next page...

Table continued from the previous page...

Field	Function
	When HALT_H = 1, the H counter does not run and no events can occur. When the HALT_H = 1, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. NOTE Once set, HALT_H can only be cleared by software to restore counter operation. HALT_H = 1 on reset. 0b - Disable 1b - Enable
17 STOP_H	Stop Counter High When STOP_H = 1 and HALT = 0, the H counter does not run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, STOP_H is cleared and counting resumes. 0b - Disable 1b - Enable
16 DOWN_H	Down Counter High Hardware sets DOWN_H when the counter is counting up, a counter limit condition occurs, and BIDIR = 1. Hardware clears DOWN_H when the counter is counting down and a limit condition occurs or when the counter reaches 0. 0b - Up. The H counter is counting up. 1b - Down. The H counter is counting down.
15-13 —	Reserved
12-5 PRE_L	Prescaler for Low Counter Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L + 1. NOTE Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.
4 BIDIR_L	Bidirectional Select Low L or unified counter direction select 0b - Up. The counter counts up to a limit condition, then is cleared to zero. 1b - Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.
3 CLRCTR_L	Clear Counter Low Writing a 1 to this bit clears the L or unified counter. CLRCTR_L always reads as 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 HALT_L	Halt Counter Low When HALT_L = 1, the L or unified counter does not run and no events can occur. When the HALT_L = 1, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. Once set, only software can clear HALT_L to restore counter operation. HALT_L = 1 on reset. 0b - Disable 1b - Enable
1 STOP_L	Stop Counter Low When STOP_L = 1 and HALT = 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, STOP_L is cleared and counting resumes. 0b - Disable 1b - Enable
0 DOWN_L	Down Counter Low Hardware sets DOWN_L when the counter is counting up, counter limit occurs, and BIDIR = 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0. 0b - Up. The L or unified counter is counting up. 1b - Down. The L or unified counter is counting down.

32.7.3.4 SCT Limit Event Select (LIMIT)

Offset

Register	Offset
LIMIT	8h

Function

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes count direction in bi-directional mode and keeps running. To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

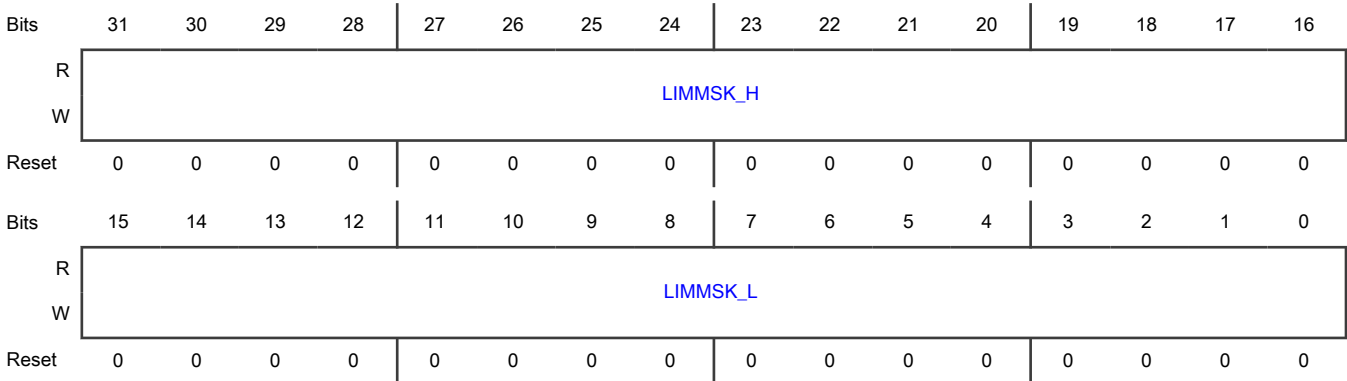
Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

In addition to using LIMIT to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature.

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the LIMIT register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Diagram



Fields

Field	Function
31-16 LIMMSK_H	Limit Event Counter High If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 LIMMSK_L	Limit Event Counter Low If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.5 Halt Event Select (HALT)

Offset

Register	Offset
HALT	Ch

Function

The running counter can be disabled (halted) by an event. When any of the events selected in the HALT register occur, the counter stops running and all further events are disabled.

Each bit of the HALT register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes the associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

NOTE

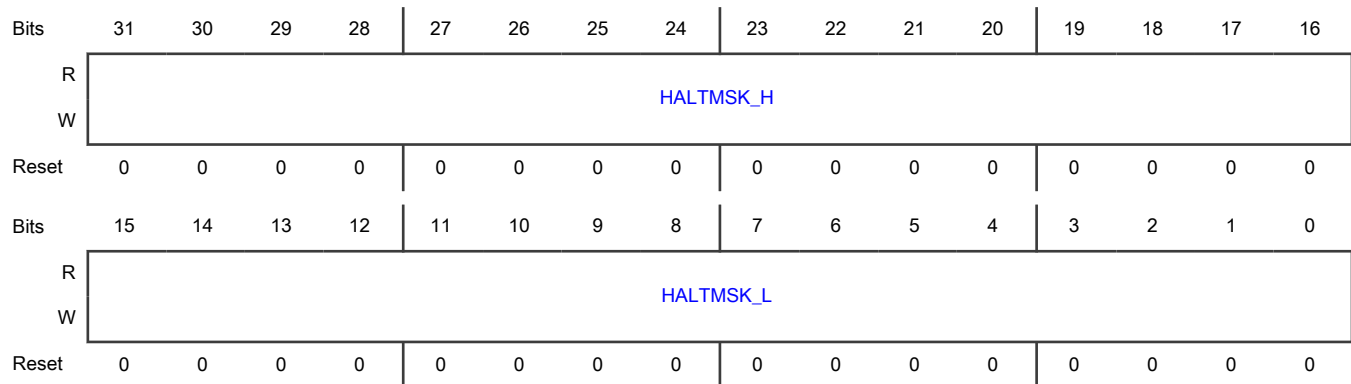
A HALT condition can only be removed when software clears the CTRL[HALT] bit.

If CONFIG[UNIFY] = 1, only the L bits are used.

If CONFIG[UNIFY] = 0, the HALT register can be written to as two registers HALT_L and HALT_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Diagram



Fields

Field	Function
31-16 HALTMSK_H	Halt Event High If bit n is one, event n sets CTRL[HALT_H] = 1 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 HALTMSK_L	Halt Event Low If bit n is one, event n sets CTRL[HALT_L] = 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.6 Stop Event Select (STOP)

Offset

Register	Offset
STOP	10h

Function

The running counter can be stopped by an event. When any of the events selected in STOP occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

The STOP register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

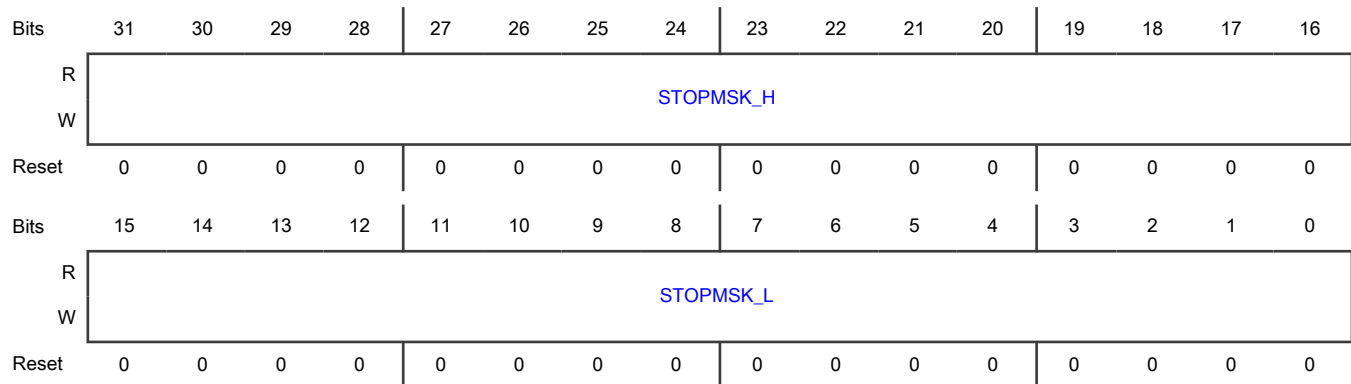
Software can stop and restart the counter by writing to the CTRL register.

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the STOP register can be written to as two registers STOP_L and STOP_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Diagram



Fields

Field	Function
31-16 STOPMSK_H	Stop Event High If bit n is one, event n sets the CTRL[STOP_H] = 1 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 STOPMSK_L	Stop Event Low If bit n is one, event n sets the CTRL[STOP_L] = 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.7 Start Event Select (START)

Offset

Register	Offset
START	14h

Function

The stopped counter can be re-started by an event. When any of the events selected in the START register occur, counting restarts from the current counter value.

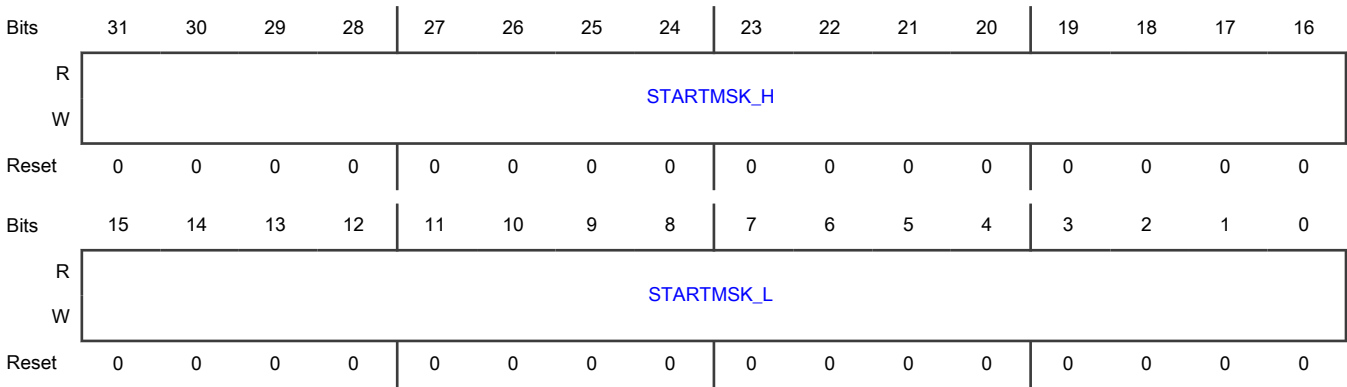
Each bit of the START register associates with a different event (bit 0 with event 0, etc.). Setting a bit causes the associated event to serve as a START event. When any START event occurs, hardware clears the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the START register can be written to as two registers START_L and START_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Diagram



Fields

Field	Function
31-16 STARTMSK_H	If bit n is one, event n clears the CTRL[STOP_H] = 0 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 STARTMSK_L	If bit n is one, event n clears the CTRL[STOP_L] = 0 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.8 Counter (COUNT)

Offset

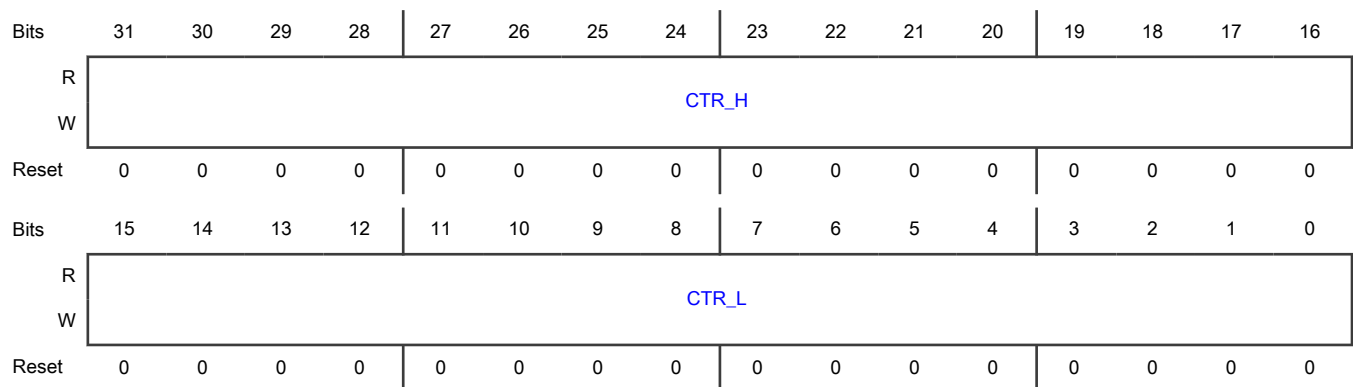
Register	Offset
COUNT	40h

Function

If CONFIG[UNIFY] = 1, the counter is a unified 32-bit register and both the _L and _H bits are used.

If CONFIG[UNIFY] = 0, the COUNT register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (CTRL[HALT] = 1). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Diagram**Fields**

Field	Function
31-16 CTR_H	Counter High When CONFIG[UNIFY] = 0, read or write the 16-bit H counter value. When CONFIG[UNIFY] = 1, read or write the upper 16 bits of the 32-bit unified counter.
15-0 CTR_L	Counter Low When CONFIG[UNIFY] = 0, read or write the 16-bit L counter value. When CONFIG[UNIFY] = 1, read or write the lower 16 bits of the 32-bit unified counter.

32.7.3.9 State (STATE)**Offset**

Register	Offset
STATE	44h

Function

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

NOTE

The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). STATE_H can only be written as a word along with STATE_L, and both counters must be halted.

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- Set and clear outputs
- Limit, stop, and start the counter

- Cause interrupts and DMA requests
- Modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

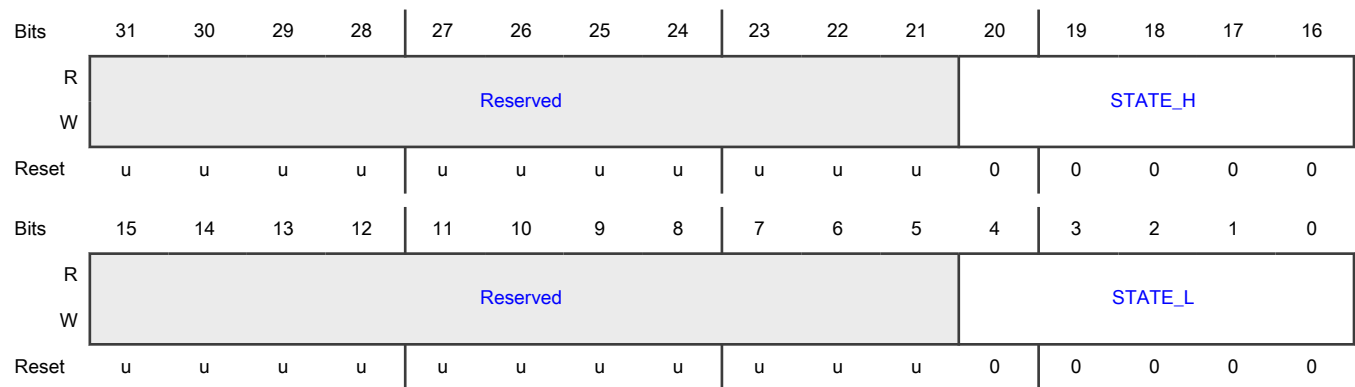
A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the STATE register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Diagram



Fields

Field	Function
31-21 —	Reserved
20-16 STATE_H	State variable
15-5 —	Reserved
4-0 STATE_L	State variable

32.7.3.10 Input (INPUT)

Offset

Register	Offset
INPUT	48h

Function

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the CONFIG[INSYNC] field:
 - If CONFIG[INSYNC] = 1, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
 - If CONFIG[INSYNC] = 0, the SIN bit value is identical to the AIN bit value.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SIN15	SIN14	SIN13	SIN12	SIN11	SIN10	SIN9	SIN8	SIN7	SIN6	SIN5	SIN4	SIN3	SIN2	SIN1	SIN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AIN15	AIN14	AIN13	AIN12	AIN11	AIN10	AIN9	AIN8	AIN7	AIN6	AIN5	AIN4	AIN3	AIN2	AIN1	AIN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SIN15	Input 15 state. Input 15 state following the synchronization specified by INSYNC.
30 SIN14	Input 14 state. Input 14 state following the synchronization specified by INSYNC.
29 SIN13	Input 13 state. Input 13 state following the synchronization specified by INSYNC.
28	Input 12 state. Input 12 state following the synchronization specified by INSYNC.

Table continues on the next page...

Table continued from the previous page...

Field	Function
SIN12	
27 SIN11	Input 11 state. Input 11 state following the synchronization specified by INSYNC.
26 SIN10	Input 10 state. Input 10 state following the synchronization specified by INSYNC.
25 SIN9	Input 9 state. Input 9 state following the synchronization specified by INSYNC.
24 SIN8	Input 8 state. Input 8 state following the synchronization specified by INSYNC.
23 SIN7	Input 7 state. Input 7 state following the synchronization specified by INSYNC.
22 SIN6	Input 6 state. Input 6 state following the synchronization specified by INSYNC.
21 SIN5	Input 5 state. Input 5 state following the synchronization specified by INSYNC.
20 SIN4	Input 4 state. Input 4 state following the synchronization specified by INSYNC.
19 SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC.
18 SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC.
17 SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC.
16 SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC.
15 AIN15	Input 15 state. Input 15 state on the last SCT clock edge.
14 AIN14	Input 14 state. Input 14 state on the last SCT clock edge.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 AIN13	Input 13 state. Input 13 state on the last SCT clock edge.
12 AIN12	Input 12 state. Input 12 state on the last SCT clock edge.
11 AIN11	Input 11 state. Input 11 state on the last SCT clock edge.
10 AIN10	Input 10 state. Input 10 state on the last SCT clock edge.
9 AIN9	Input 9 state. Input 9 state on the last SCT clock edge.
8 AIN8	Input 8 state. Input 8 state on the last SCT clock edge.
7 AIN7	Input 7 state. Input 7 state on the last SCT clock edge.
6 AIN6	Input 6 state. Input 6 state on the last SCT clock edge.
5 AIN5	Input 5 state. Input 5 state on the last SCT clock edge.
4 AIN4	Input 4 state. Input 4 state on the last SCT clock edge.
3 AIN3	Input 3 state. Input 3 state on the last SCT clock edge.
2 AIN2	Input 2 state. Input 2 state on the last SCT clock edge.
1 AIN1	Input 1 state. Input 1 state on the last SCT clock edge.
0 AIN0	Input 0 state. Input 0 state on the last SCT clock edge.

32.7.3.11 Match/Capture Mode (REGMODE)

Offset

Register	Offset
REGMODE	4Ch

Function

If CONFIG[UNIFY] = 1, only the _L bits of the REGMODE register are used. In this case, REGMODE_H is not used.

If CONFIG[UNIFY] = 0, the REGMODE register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. The _L bits/registers control the L Match/Capture registers, and the _H bits/registers control the H Match/Capture registers.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register or as a Capture register. Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register or as a Capture-Control register when the register is used as a Capture register. REGMODE_H is used only when CONFIG[UNIFY] = 0.

Diagram

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

Fields

Field	Function
31-16 REGMOD_Hn	Register Mode High n REGMOD_Hn bit controls Match/Capture register n. 0b - Match. Register n operates as a match register 1b - Capture. Register n operates as a capture register
15-0 REGMOD_Ln	Register Mode Low n REGMOD_Ln bit controls Match/Capture register n. 0b - Match. Register n operates as a match register 1b - Capture. Register n operates as a capture register

32.7.3.12 Output (OUTPUT)

Offset

Register	Offset
OUTPUT	50h

Function

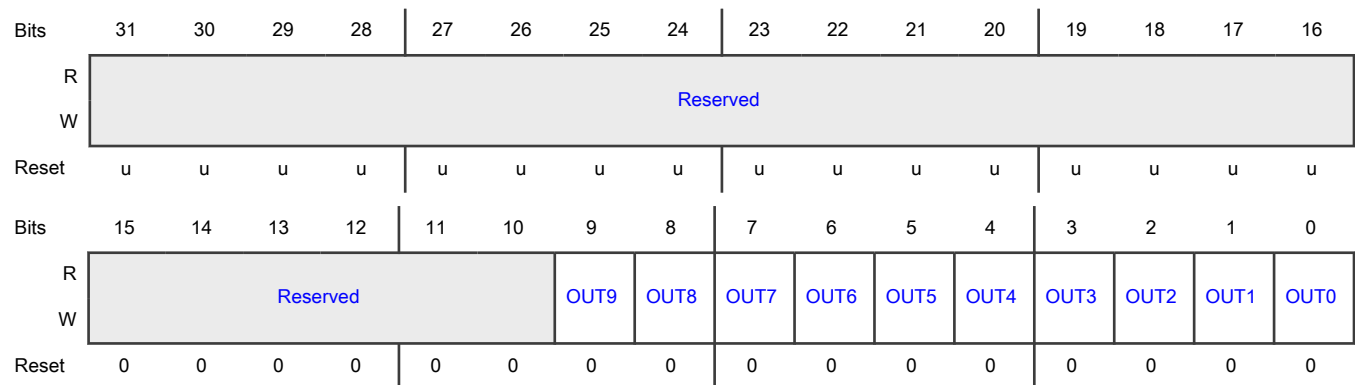
Each SCT output has a corresponding bit in the OUTPUT register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using the OUTPUT register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (CTRL[HALT] = 1).

Software can read this register at any time to sense the state of the outputs.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-10 —	Reserved
9-0 OUTn	Output n The number of bits equals the number of outputs in this SCT. 0b - Writing a 0 forces the corresponding output low 1b - Writing a 1 forces the corresponding output high

32.7.3.13 Output Counter Direction Control (OUTPUTDIRCTRL)

Offset

Register	Offset
OUTPUTDIRCTRL	54h

Function

For bi-directional mode, the OUTPUTDIRCTRL register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output. The purpose of the OUTPUTDIRCTRL register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												SETCLR9		SETCLR8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SETCLR7		SETCLR6		SETCLR5		SETCLR4		SETCLR3		SETCLR2		SETCLR1		SETCLR0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-18: SETCLR9	Set/Clear Operation on Output n 00b - Set and clear do not depend on the direction of any counter. 01b - Set and clear are reversed when counter L or the unified counter is counting down. 10b - Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. 11b - Reserved. Do not program this value.
17-16: SETCLR8	
15-14: SETCLR7	
13-12: SETCLR6	
11-10: SETCLR5	
9-8: SETCLR4	
7-6: SETCLR3	

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-4: SETCLR2 3-2: SETCLR1 1-0: SETCLR0	

32.7.3.14 Output Conflict Resolution (RES)

Offset

Register	Offset
RES	58h

Function

The Output Conflict Resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												O9RES		O8RES	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	O7RES		O6RES		O5RES		O4RES		O3RES		O2RES		O1RES		O0RES	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19-18: O9RES 17-16: O8RES	Effect of simultaneous set and clear on output n 00b - No change

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-14: O7RES	01b - Set output (or clear based on the OUTPUTDIRCTRL[SETCLRn] field)
13-12: O6RES	10b - Clear output (or set based on the OUTPUTDIRCTRL[SETCLRn] field)
11-10: O5RES	11b - Toggle output
9-8: O4RES	
7-6: O3RES	
5-4: O2RES	
3-2: O1RES	
1-0: O0RES	

32.7.3.15 DMA Request 0 (DMAREQ0)

Offset

Register	Offset
DMAREQ0	5Ch

Function

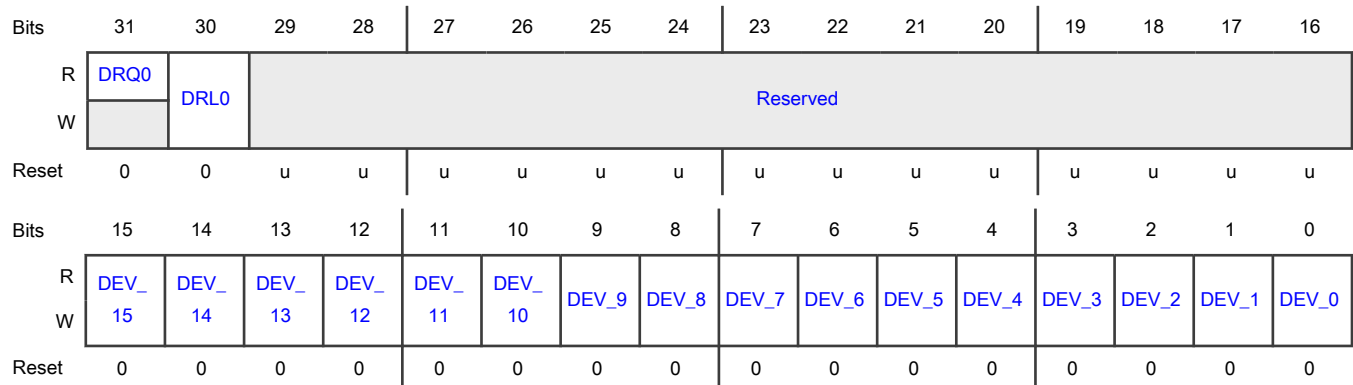
The SCT includes two DMA request outputs. These registers enable the DMA requests to be generated when a particular event occurs or when Counter Match registers are loaded from the Reload registers.

The DMA Request registers are word-write only. Attempting to write a halfword value to these registers results in a bus error.

Note that on this device, SCT DMA requests are connected to the DMA trigger inputs, and controlled by the DMA trigger input mux registers.

Event-generated DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Diagram



Fields

Field	Function
31 DRQ0	DMA Request 0 State This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag. DRQ0 will be cleared rapidly by the DMA service. If the flag remains set there could be an issue with DMA setup.
30 DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCHn_L/Unified registers from the RELOADn_L/Unified registers.
29-16 —	Reserved
15-0 DEV_n	DMA Request Event n If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.16 DMA Request 1 (DMAREQ1)

Offset

Register	Offset
DMAREQ1	60h

Function

This is the DMA request register 1.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DRQ1	DRL1	Reserved													
W																
Reset	0	0	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEV_15	DEV_14	DEV_13	DEV_12	DEV_11	DEV_10	DEV_9	DEV_8	DEV_7	DEV_6	DEV_5	DEV_4	DEV_3	DEV_2	DEV_1	DEV_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 DRQ1	DMA Request 1 State This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag. DRQ1 will be cleared rapidly by the DMA service. If the flag remains set there could be an issue with DMA setup.
30 DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.
29-16 —	Reserved
15-0 DEV_n	DMA Request Event n If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

32.7.3.17 Event Interrupt Enable (EVEN)

Offset

Register	Offset
EVEN	F0h

Function

The EVEN register enables flags to request an interrupt if the FLAGn bit in the SCT Event Flag register is also set.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0 IENn	Event Interrupt Enable n 0b - Disable 1b - Enable

32.7.3.18 Event Flag (EVFLAG)

Offset

Register	Offset
EVFLAG	F4h

Function

The EVFLAG register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all of the enabled flag register bits are zero.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLAG1	FLAG1	FLAG1	FLAG1	FLAG1	FLAG1	FLAG9	FLAG8	FLAG7	FLAG6	FLAG5	FLAG4	FLAG3	FLAG2	FLAG1	FLAG0
W	5	4	3	2	1	0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-0	Event Flag n

Table continues on the next page...

Table continued from the previous page...

Field	Function
FLAGn	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT. 0b - No Flag 1b - Event n Flag

32.7.3.19 Conflict Interrupt Enable (CONEN)

Offset

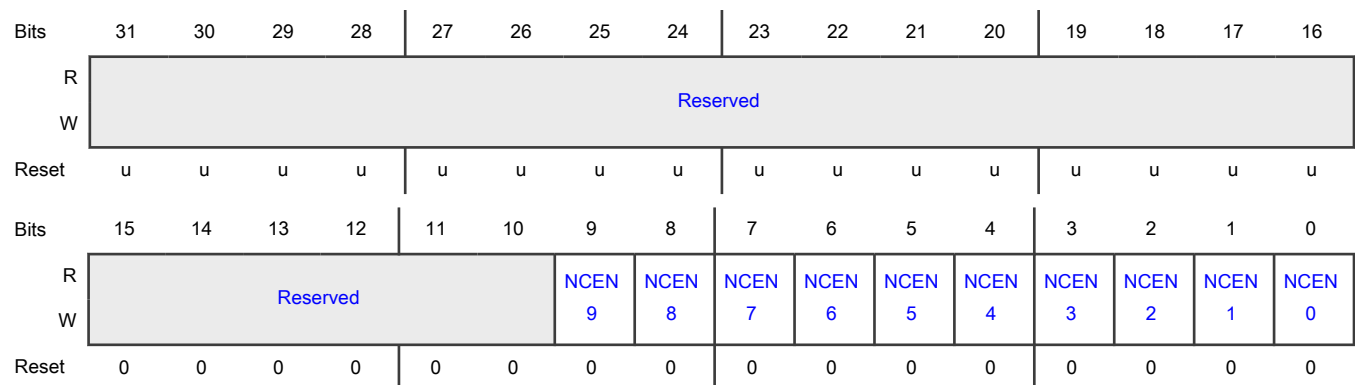
Register	Offset
CONEN	F8h

Function

The CONEN register enables the no-change conflict events specified in the SCT Conflict Resolution register to generate an interrupt request.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-10 —	Reserved
9-0 NCENn	No Change Conflict Event/Interrupt Enable The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, etc.).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No interrupt 1b - Interrupt

32.7.3.20 Conflict Flag (CONFLAG)

Offset

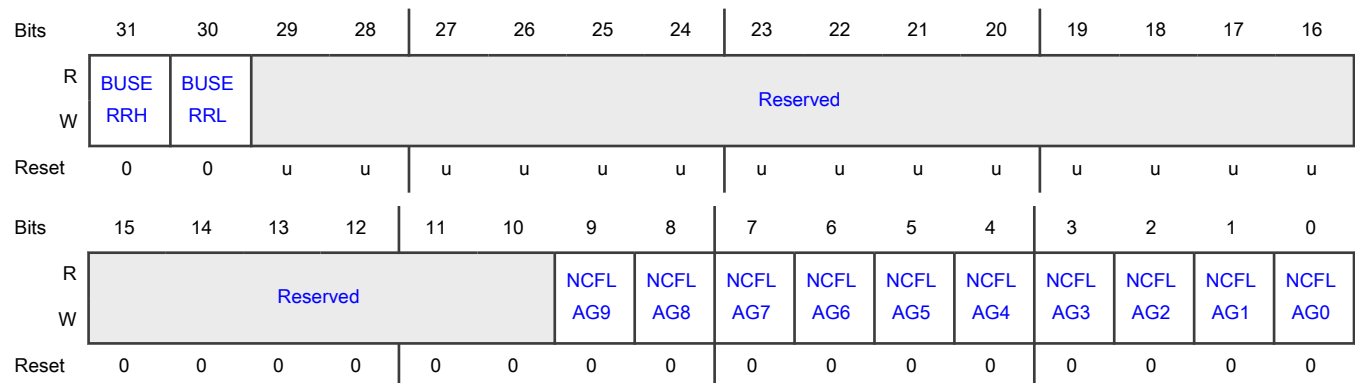
Register	Offset
CONFLAG	FCh

Function

The CONFLAG register records a no-change conflict occurrence and also provides details of any bus errors. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31 BUSERRH	Bus Error High The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.
30 BUSERRL	Bus Error Low/Unified The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.
29-10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
9-0 NCFLAGn	No Change Conflict Event Flag 0b - No Conflict Event 1b - A No Change Conflict Event occurred

32.7.3.21 Capture Value (CAP0 - CAP15)

Offset

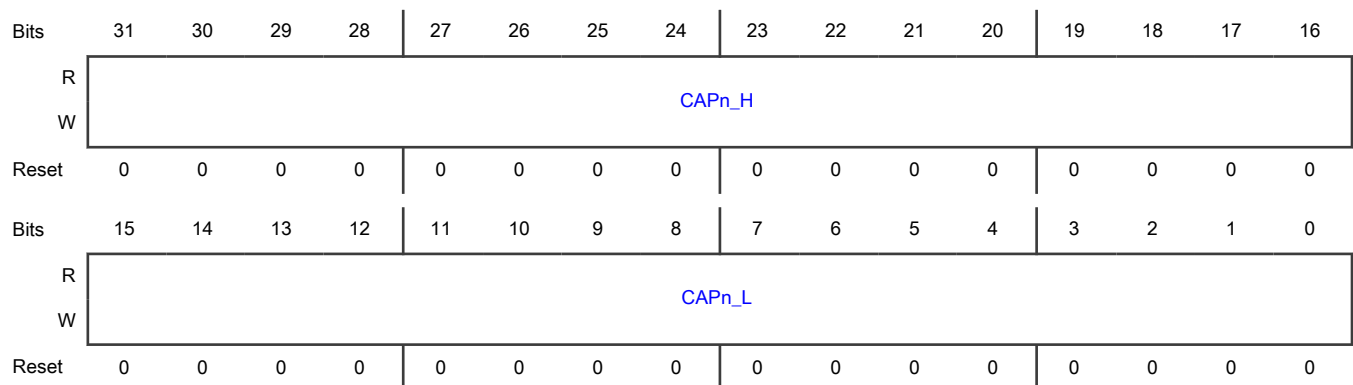
For n = 0 to 15:

Register	Offset
CAPn	100h + (n × 4h)

Function

The CAPn registers allow software to record the counter values when the events selected by the corresponding Capture Control registers occur.

Diagram



Fields

Field	Function
31-16 CAPn_H	Capture High When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.
15-0 CAPn_L	Capture Low When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value where this register was last captured.

32.7.3.22 Match Value (MATCH0 - MATCH15)

Offset

For $n = 0$ to 15:

Register	Offset
MATCHn	$100h + (n \times 4h)$

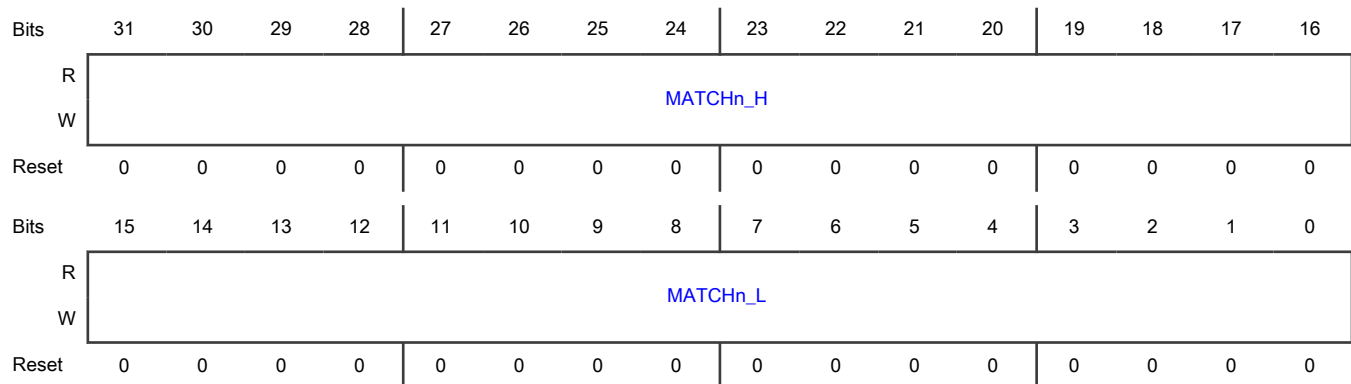
Function

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCHn_L or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). MATCHn_H can only be written as a word along with MATCHn_L, and both counters must be halted. Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a match event limits its counter, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Diagram



Fields

Field	Function
31-16 MATCHn_H	Match High When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.
15-0 MATCHn_L	Match Low When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.

32.7.3.23 Capture Control (CAPCTRL0 - CAPCTRL15)

Offset

For n = 0 to 15:

Register	Offset
CAPCTRLn	200h + (n × 4h)

Function

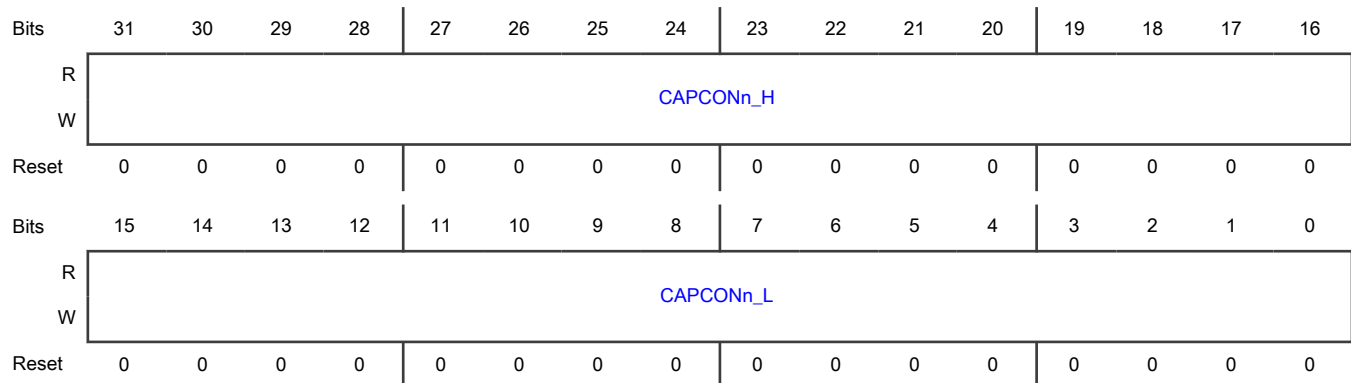
If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Based on a selected event, the Capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Diagram



Fields

Field	Function
31-16 CAPCONn_H	Capture Control High If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of match/captures in this SCT.
15-0 CAPCONn_L	Capture Control Low If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of match/captures in this SCT.

32.7.3.24 Match Reload Value (MATCHREL0 - MATCHREL15)

Offset

For n = 0 to 15:

Register	Offset
MATCHRELn	200h + (n × 4h)

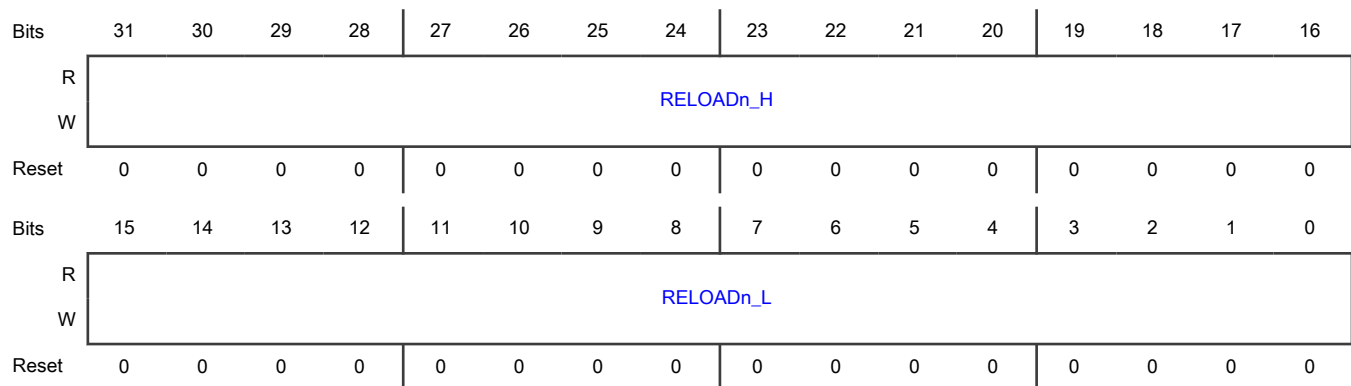
Function

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- When CTRL[BIDIR] = 0 and the counter is cleared to zero upon reaching its limit condition.

When CTRL[BIDIR] = 1 and the counter counts down to 0, unless CFG[NORELOAD] = 1.

Diagram



Fields

Field	Function
31-16 RELOADn_H	Reload High When CONFIG[UNIFY] = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When CONFIG[UNIFY] = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.
15-0 RELOADn_L	Reload Low When CONFIG[UNIFY] = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When CONFIG[UNIFY] = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.

32.7.3.25 Event n State (EV0_STATE - EV15_STATE)

Offset

For n = 0 to 15:

Register	Offset
EVn_STATE	300h + (n × 8h)

Function

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRLn register has one associated Event Enable register that can enable or disable the event for each available state.

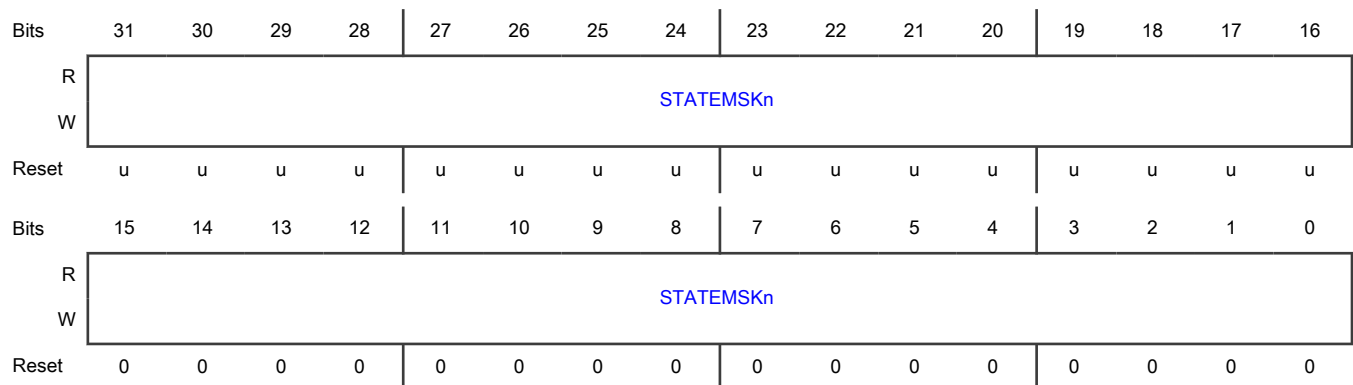
Each event has one associated SCT Event State Mask register that allows this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state does not change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-0	Event State Mask
STATEMSKn	If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1 = bit 1, etc.). The number of bits = number of states in this SCT.

32.7.3.26 Event n Control (EV0_CTRL - EV15_CTRL)

Offset

For n = 0 to 15:

Register	Offset
EVn_CTRL	304h + (n × 8h)

Function

This register defines the conditions for an event to occur based on the counter values or input and output states. Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATEn register. To block all events for a given counter, set the CTRL[HALT] bit equal to one or select an event to halt the counter.

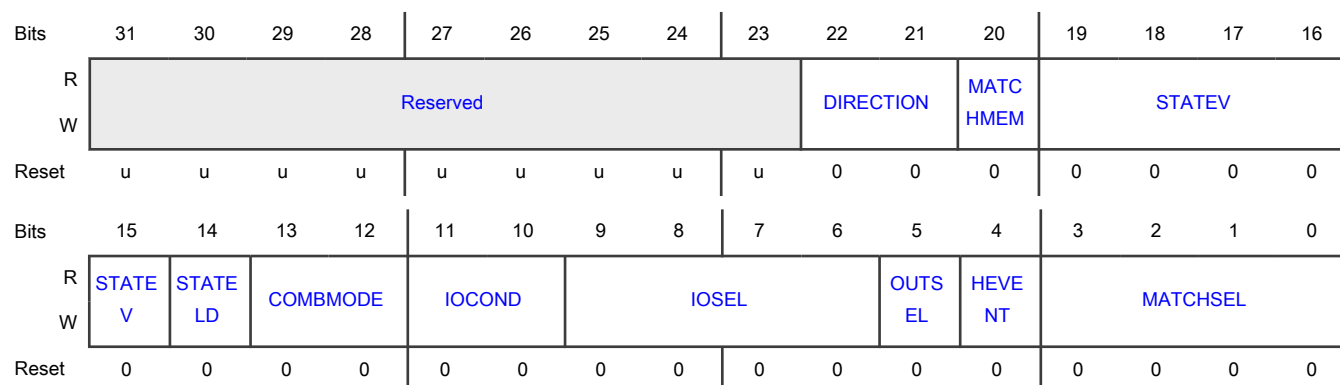
An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected Match register. In bidirectional mode, events can also be enabled based on the direction of count.

When CONFIG[UNIFY] = 0, each event is associated with a particular counter by the HEVENT bit in its Event Control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

Diagram



Fields

Field	Function
31-23 —	Reserved
22-21 DIRECTION	Direction Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. 00b - Direction independent. This event is triggered regardless of the count direction. 01b - Counting up. This event is triggered only during up-counting when BIDIR = 1. 10b - Counting down. This event is triggered only during down-counting when BIDIR = 1. 11b - Reserved
20 MATCHMEM	Match Mem If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.
19-15 STATEV	State Value This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 STATELD	State Load This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state. 0b - Add. STATEV value is added into STATE (the carry-out is ignored). 1b - Load. STATEV value is loaded into STATE.
13-12 COMBMODE	Combination Mode Selects how the specified match and I/O condition are used and combined. 00b - OR. The event occurs when either the specified match or I/O condition occurs. 01b - MATCH. Uses the specified match only. 10b - IO. Uses the specified I/O condition only. 11b - AND. The event occurs when the specified match and I/O condition occur simultaneously.
11-10 IOCOND	Input/Output Condition Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period . 00b - Low 01b - Rise 10b - Fall 11b - High
9-6 IOSEL	Input/Output Signal Select Selects the input or output signal number associated with this event (if any). Do not select an input in this register if CKMODE is 1x. In this case the clock input is an implicit ingredient of every event.
5 OUTSEL	Input/Output Select 0b - Selects the inputs selected by IOSEL. 1b - Selects the outputs selected by IOSEL.
4 HEVENT	High Event Select L/H counter. Do not set this bit if UNIFY = 1. 0b - Low Counter. Selects the L state and the L match register selected by MATCHSEL. 1b - High Counter. Selects the H state and the H match register selected by MATCHSEL.
3-0 MATCHSEL	Match Select Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.

32.7.3.27 Output n Set (OUT0_SET - OUT9_SET)

Offset

For n = 0 to 9:

Register	Offset
OUTn_SET	500h + (n × 8h)

Function

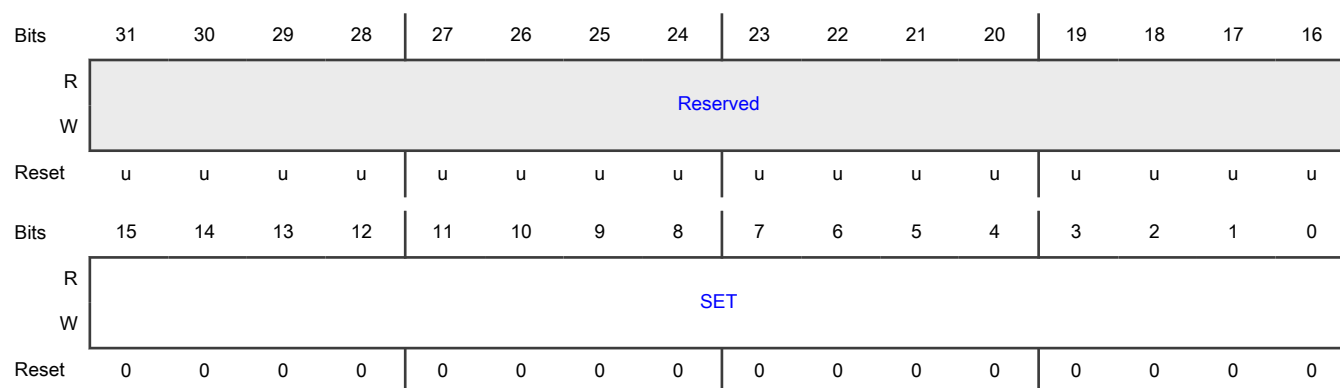
Based on a selected event, each SCT output can be set. There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

NOTE

If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 SET	Set A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, etc. The number of bits = number of events in this SCT. When the counter is used in bidirectional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTDIRCTRL register.

32.7.3.28 Output n Clear (OUT0_CLR - OUT9_CLR)

Offset

For n = 0 to 9:

Register	Offset
OUTn_CLR	504h + (n × 8h)

Function

Based on a selected event, each SCT output can be cleared.

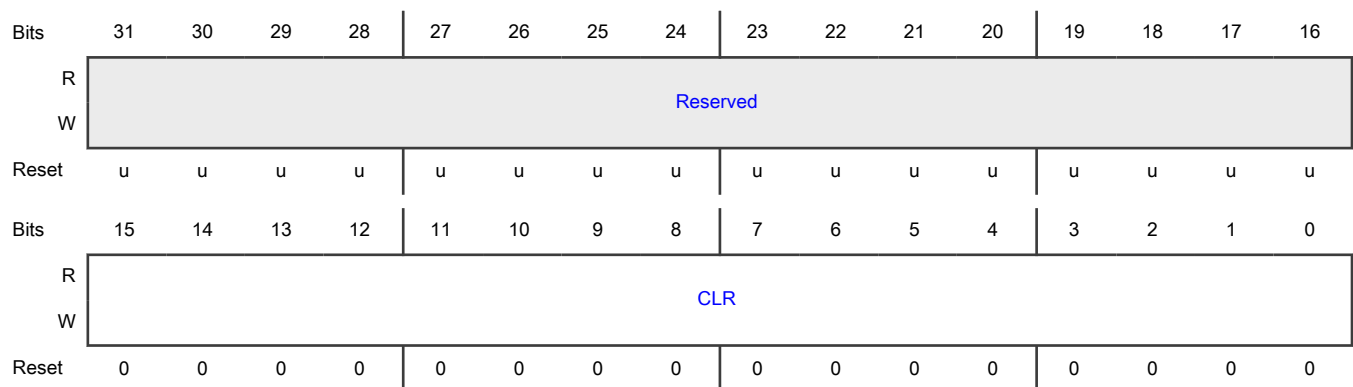
There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Only word-writes are permitted to this register. Attempting to write a halfword value results in a bus error.

NOTE

If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CLR	Clear A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, etc. The number of bits = number of events in this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.

Chapter 33

Micro-Tick Timer (UTICK)

33.1 Chip-specific UTICK information

Table 403. Reference links to related information

Topic	Related module	Reference
Full description	UTICK	UTICK
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

33.1.1 Module instances

This device has one instance of the UTICK module, UTICK0.

33.2 Overview

UTICK is a 31-bit timer that counts down to 0 and then generates an interrupt. Thus, it provides a fixed time interval between interrupts. This is a simple, ultra-low-power timer that can run and wake up the chip in any reduced-power mode, including deep power-down modes.

33.2.1 Block diagram

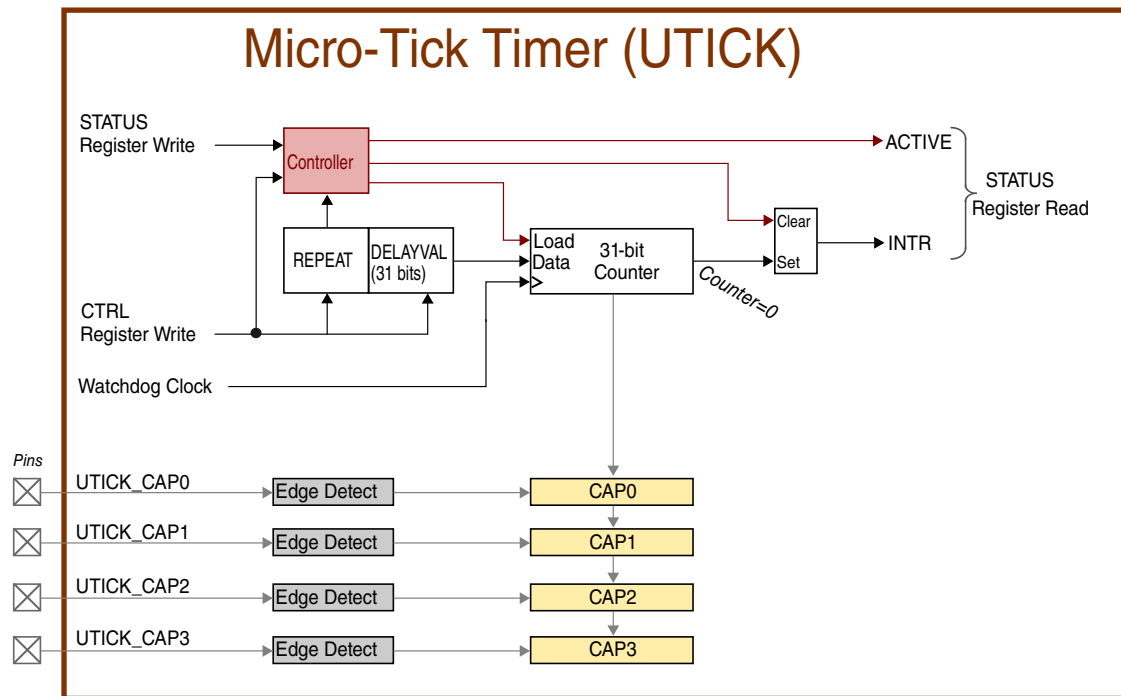


Figure 123. Block diagram

33.2.2 Features

- Wakes up the chip in any low-power mode.
- Starts with a nonzero value written to [CTRL\[DELAYVAL\]](#).
- Operates using Interrupt or Software Polling mode.
- Includes [Capture \(CAP0 - CAP3\)](#) registers that use external pin transitions as a trigger.

33.3 Functional description

33.3.1 Operations

Perform the following procedure before using UTICK:

1. Configure [CTRL\[REPEAT\]](#) to either perform a one-time delay or continuous delay.
2. Configure [CTRL\[DELAYVAL\]](#) to specify the delay time value in terms of the timer clock.
3. Configure [CAP_n\[CAP_VALUE\]](#) to decide the number of events to capture (the maximum is four).
4. Configure [CFG\[CAPPOL_n\]](#) to capture the polarity (whether to capture on a positive or negative edge of a signal).

33.4 Signals

Signal	I/O	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	I	Capture inputs. Configure the selected transition on a capture pin to load Capture (CAP0 - CAP3) with the counter value.

33.5 Initialization

To initialize UTICK:

1. Enable the clock to the UTICK register interface.
2. Enable UTICK interrupts for waking up from deep-sleep and deep power-down modes.
3. Configure the pin functions of any UTICK capture pins that are used.
4. Enable the WDOG oscillator that provides the UTICK clock.

33.6 Memory map and register definition

33.6.1 UTICK register descriptions

33.6.1.1 UTICK memory map

UTICK0 base address: 4000_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CTRL)	32	RW	0000_0000h
4h	Status (STAT)	32	RW	See section
8h	Capture Configuration (CFG)	32	RW	See section
Ch	Capture Clear (CAPCLR)	32	W	See section
10h - 1Ch	Capture (CAP0 - CAP3)	32	R	0000_0000h

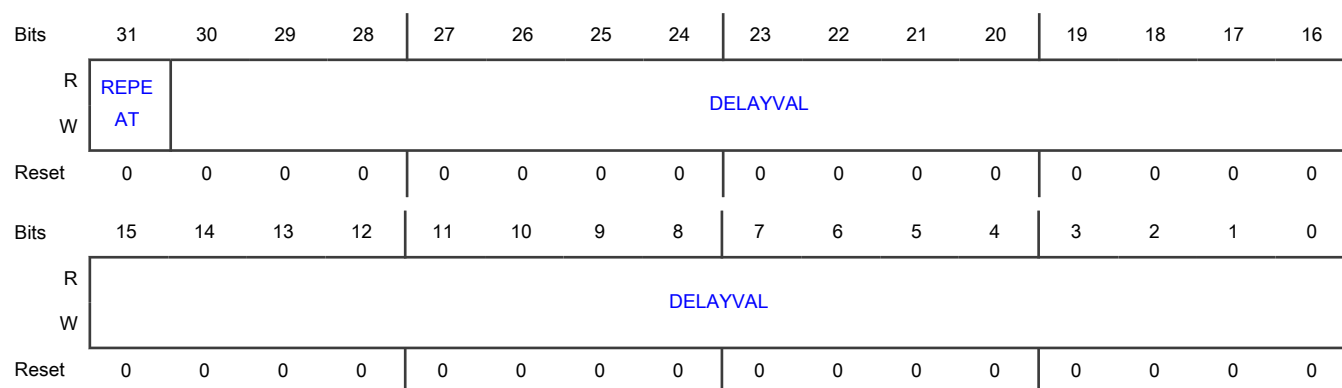
33.6.1.2 Control (CTRL)

Offset

Register	Offset
CTRL	0h

Function

Resets the counter, which means a new interval is measured if one is in progress.

Diagram**Fields**

Field	Function
31 REPEAT	Repeat Delay Specifies the frequency of repeat delay. 0b - One-time delay 1b - Delay repeats continuously
30-0 DELAYVAL	Tick Interval Specifies the tick interval. The resulting delay is (DELAYVAL + 1) periods of the timer clock. The minimum usable tick interval is one clock cycle, for a delay of two timer clocks. A tick interval = 0 stops the timer. 000_0000_0000_0000_0000_0000_0000b - Reserved All other values - Clock cycles as defined in the description

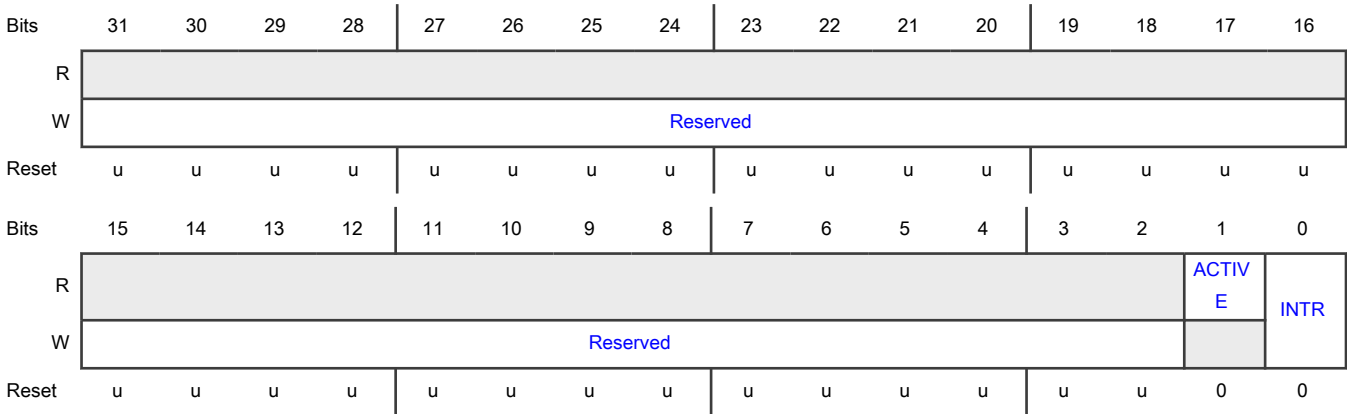
33.6.1.3 Status (STAT)**Offset**

Register	Offset
STAT	4h

Function

Specifies the UTICK and interrupt status.

Diagram



Fields

Field	Function
31-2 —	Reserved
1 ACTIVE	Timer Active Flag Specifies whether UTICK is currently active. 0b - Inactive (stopped) 1b - Active
0 INTR	Interrupt Flag Specifies whether an interrupt is pending. 0b - Not pending 1b - Pending

33.6.1.4 Capture Configuration (CFG)

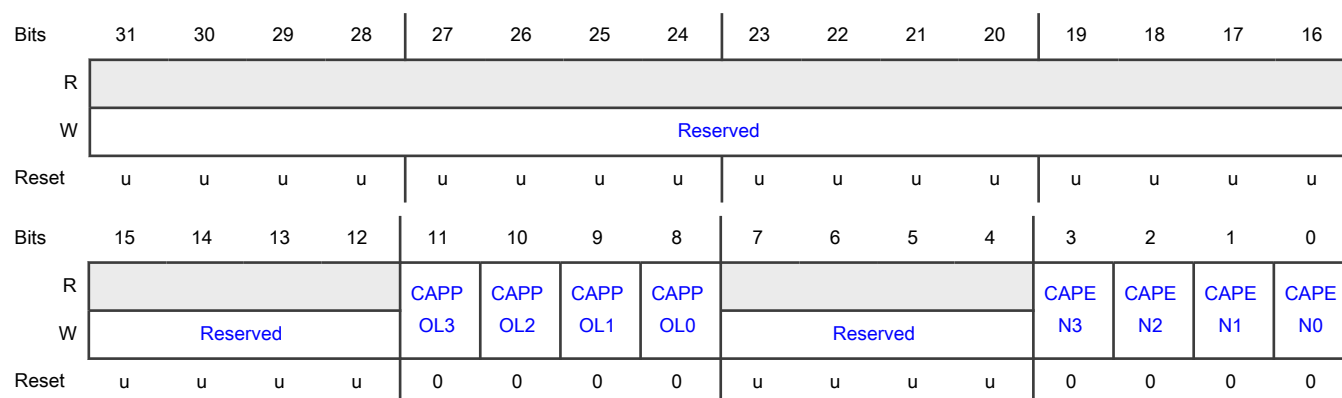
Offset

Register	Offset
CFG	8h

Function

Enables micro-tick capture functions and selects the polarity of capture triggers.

Diagram



Fields

Field	Function
31-12 —	Reserved
11 CAPPOL3	Capture Polarity 3 Specifies the capture edge. 0b - Positive 1b - Negative
10 CAPPOL2	Capture Polarity 2 Specifies the capture edge. 0b - Positive 1b - Negative
9 CAPPOL1	Capture-Polarity 1 Specifies the capture edge. 0b - Positive 1b - Negative
8 CAPPOL0	Capture Polarity 0 Specifies the capture edge. 0b - Positive 1b - Negative
7-4 —	Reserved
3	Enable Capture 3

Table continues on the next page...

Table continued from the previous page...

Field	Function
CAPEN3	0b - Disable 1b - Enable
2 CAPEN2	Enable Capture 2 0b - Disable 1b - Enable
1 CAPEN1	Enable Capture 1 0b - Disable 1b - Enable
0 CAPEN0	Enable Capture 0 0b - Disable 1b - Enable

33.6.1.5 Capture Clear (CAPCLR)

Offset

Register	Offset
CAPCLR	Ch

Function

Allows you to clear previous capture values, enabling new captures to take place.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	Reserved												CAPC LR3	CAPC LR2	CAPC LR1	CAPC LR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-4 —	Reserved
3 CAPCLR3	Clear Capture 3 0b - Does nothing 1b - Clears the CAP3 register value
2 CAPCLR2	Clear Capture 2 0b - Does nothing 1b - Clears the CAP2 register value
1 CAPCLR1	Clear Capture 1 0b - Does nothing 1b - Clears the CAP1 register value
0 CAPCLR0	Clear Capture 0 0b - Does nothing 1b - Clears the CAP0 register value

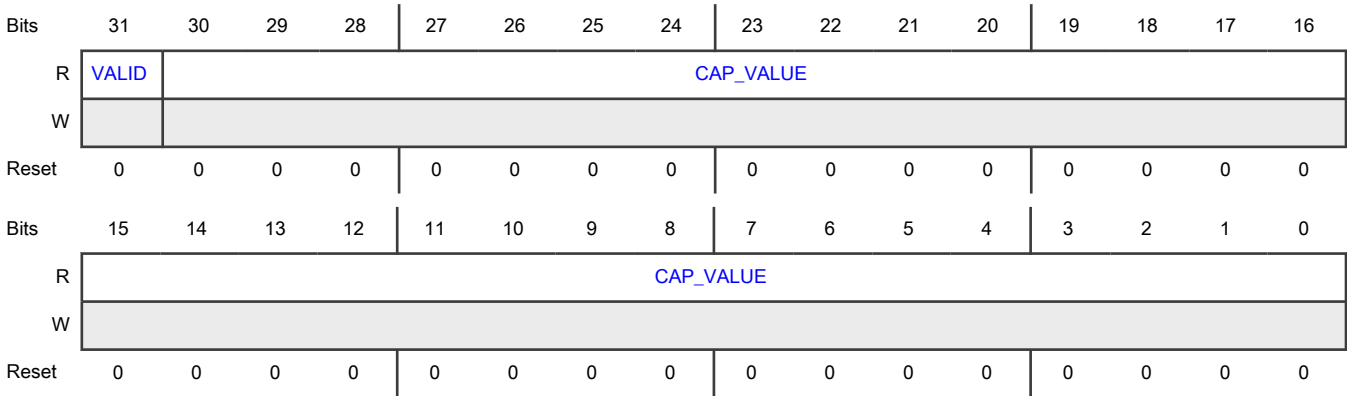
33.6.1.6 Capture (CAP0 - CAP3)**Offset**

Register	Offset
CAP0	10h
CAP1	14h
CAP2	18h
CAP3	1Ch

Function

Contains the micro-tick time value based on any previously captured events. Each capture register is associated with one of the capture trigger inputs.

Diagram



Fields

Field	Function
31 VALID	Captured Value Valid Flag Specifies whether a valid value is captured, based on a transition of the related UTICK_CAPn pin. Write 1 to the related field in Capture Clear (CAPCLR) to clear this flag. 0b - Valid value not captured 1b - Valid value captured
30-0 CAP_VALUE	Captured Value for the Related Capture Event Specifies the captured value for the related capture event. The captured value is 1 lower than the actual value of UTICK at the moment of the capture event (actual_captured_value = timer_value - 1).

Chapter 34

Multi-Rate Timer (MRT)

34.1 Chip-specific MRT information

Table 404. Reference links to related information

Topic	Related module	Reference
Full description	MRT	MRT
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

34.1.1 Module instances

This device has one instance of the MRT module, MRT0.

34.2 Overview

MRT provides a repetitive interrupt timer with 4 channels.

34.2.1 Block diagram

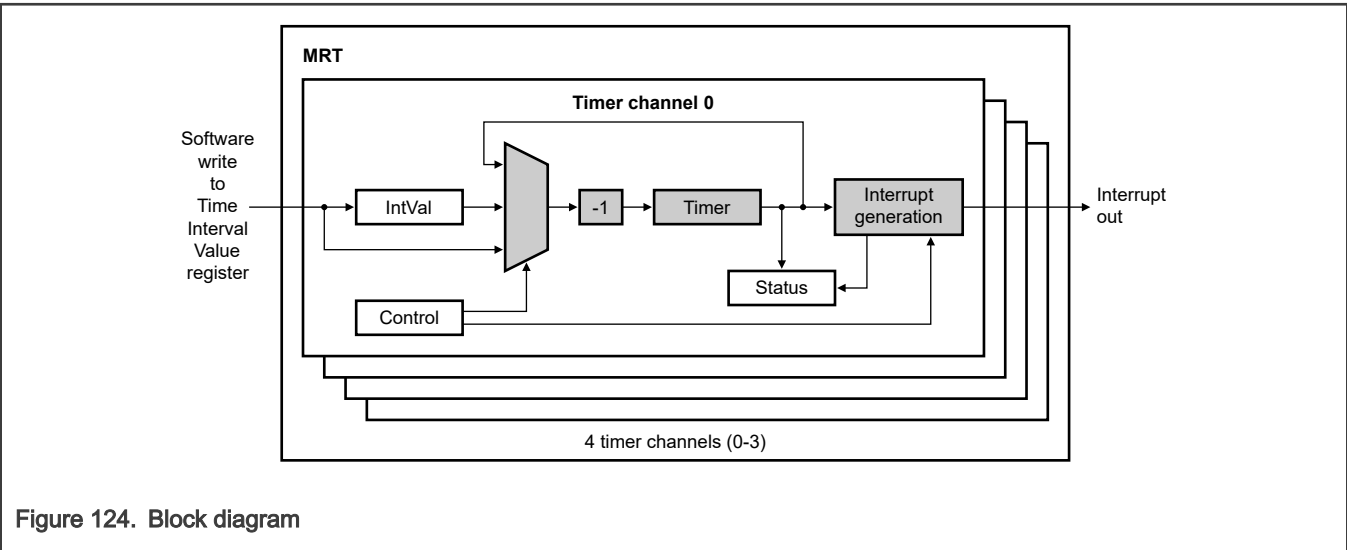


Figure 124. Block diagram

34.2.2 Features

- 24-bit interrupt timer.
- There are 4 channels independently counting down from individually set values.
- Repeat Interrupt, One-Shot Interrupt, and One-Shot Stall modes.
- You can program individual time intervals for each channel.
- Each channel operates independently of the other channels.

34.3 Functional description

The following sections discuss various modes that affect the functionality of MRT.

34.3.1 Operating modes

MRT has the following three operating modes:

- [Repeat Interrupt mode](#)
- [One-Shot Interrupt mode](#)
- [One-Shot Stall mode](#)

34.3.1.1 Repeat Interrupt mode

- Generates repeated interrupts after a selected time interval.
- Gets used for application-based Pulse Width Modulation (PWM) or Pulse Position Modulation (PPM).

When $TIMER_n$ is in idle state, writing a nonzero value (IVALUE) to $INTVAL_n$ immediately loads the time interval value (IVALUE -1). Then the timer begins to count down from this value. After the timer reaches 0,

- an interrupt is generated.
- the value in $INTVAL_n$ (IVALUE -1) is automatically reloaded, and
- the timer starts to count down again.

When the timer is running in Repeat Interrupt mode, you can perform various interval timer actions (change or stop):

Table 405. Possible actions in Repeat Interrupt mode

Action	To perform action	Timer does this	Interrupt?
Change the interval value on the next interval timer cycle.	• Write new value (>0) to $INTVAL_n$. • Write 0 to LOAD.	On the next cycle, the timer counts down from the new value.	An interrupt is generated when the timer reaches 0.
Change the interval value immediately (also called "change on-the-fly") in the interval timer.	• Write new value (>0) to $INTVAL_n$. • Write 1 to LOAD.	The timer immediately starts to count down from the new timer interval value.	An interrupt is generated when the timer reaches 0.
Stop the interval timer at the end of the time interval.	• Write 0 to $INTVAL_n$. • Write 0 to LOAD.	The timer stops at the end of the time interval.	An interrupt is generated when the timer reaches 0.
Stop the interval timer immediately.	• Write 0 to $INTVAL_n$. • Write 1 to LOAD.	The timer stops immediately.	No interrupt is generated when $INTVAL_n$ is written.

34.3.1.2 One-Shot Interrupt mode

- In this mode, the timer generates one interrupt after a one-time count.
- This mode allows you to generate an interrupt at any time.
- Using this mode, you can introduce a specific delay in a software task.

When the timer is in an idle state, writing a nonzero value (IVALUE) to $INTVAL_n$ immediately loads the time interval value (IVALUE -1), and the timer starts to count down. As soon as the timer reaches 0, the timer stops and an interrupt is generated (and then enters the idle state).

When the timer is running in One-Shot Interrupt mode, you can perform the following interval timer actions (load or stop):

Table 406. Possible actions in One-Shot Interrupt mode

Action	To perform action	Timer does this	Interrupt
Load a timer interval value.	- Write a new time interval value (>0) to INTVALn. - Write 1 to LOAD.	The timer immediately reloads the new time interval, and starts counting down from the new value.	An interrupt is generated when the timer reaches 0.
Stop the interval timer.	- Write a 0 to INTVALn. - Write 1 to LOAD.	The timer immediately stops counting and moves to the idle state.	No interrupt is generated when INTVAL n is updated.

34.3.1.3 One-Shot Stall mode

- Used for very short delays (for example, when the delay needed is less than the time it takes to arrive at an interrupt service routine).
- Designed for very low software overhead that requires only a single write to INTVAL n (if the channel is already configured for One-Shot stall mode). MRT counts the requested delay when stalling the bus write operation, and then concluding the write operation when the delay has finished. No interrupt or status polling is needed.

Use this mode:

- when a short delay is needed between two software-controlled events.
- when a delay is expected before your application can continue.

The CPU consumes a minimum amount of power during One-Shot Stall mode, because no bus transactions occur during that period.

One-Shot Stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL n and the time that your software resumes execution. Other system events, such as interrupts or other bus masters accessing the APB bus where MRT resides, can cause the delay to be longer.

34.4 Signals

There are no external MRT signals.

34.5 Initialization

To initialize:

1. Enable the clock to MRT.
2. Deassert MRT reset.
3. Ensure that MRT can issue an interrupt in the chip.

See the Chip-specific section for more details.

34.6 Memory map and register descriptions

34.6.1 Multi-Rate Timer (MRT) register descriptions

34.6.1.1 MRT memory map

MRT0 base address: 4002_D000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Time Interval Value (INTVAL0)	32	RW	See section
4h	Timer (TIMER0)	32	R	See section
8h	Control (CTRL0)	32	RW	See section
Ch	Status (STAT0)	32	RW	See section
10h	Time Interval Value (INTVAL1)	32	RW	See section
14h	Timer (TIMER1)	32	R	See section
18h	Control (CTRL1)	32	RW	See section
1Ch	Status (STAT1)	32	RW	See section
20h	Time Interval Value (INTVAL2)	32	RW	See section
24h	Timer (TIMER2)	32	R	See section
28h	Control (CTRL2)	32	RW	See section
2Ch	Status (STAT2)	32	RW	See section
30h	Time Interval Value (INTVAL3)	32	RW	See section
34h	Timer (TIMER3)	32	R	See section
38h	Control (CTRL3)	32	RW	See section
3Ch	Status (STAT3)	32	RW	See section
F0h	Module Configuration (MODCFG)	32	RW	See section
F4h	Idle Channel (IDLE_CH)	32	R	See section
F8h	Global Interrupt Flag (IRQ_FLAG)	32	RW	See section

34.6.1.2 Time Interval Value (INTVAL0 - INTVAL3)

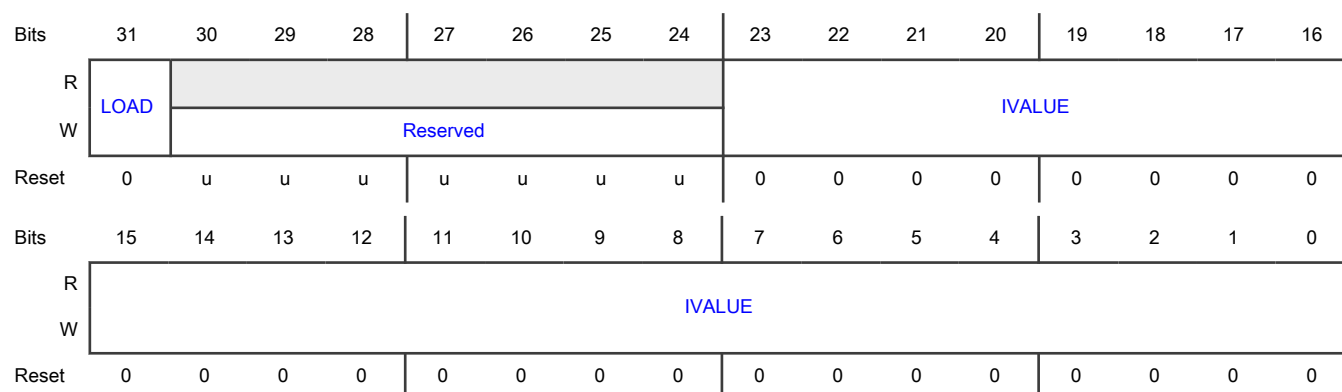
Offset

Register	Offset
INTVAL0	0h
INTVAL1	10h
INTVAL2	20h
INTVAL3	30h

Function

Contains MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Diagram



Fields

Field	Function
31 LOAD	Force Load Enable Determines how the timer interval value (IVALUE - 1) is loaded into the $TIMER_n$. LOAD is write only. Reading LOAD always returns 0. If the repeat mode is selected, the load from the $INTVAL_n$ to the $TIMER_n$ is processed at the end of the time interval. The $INTVAL_n$ interval value (IVALUE - 1) is immediately loaded into the $TIMER_n$ when it is running. 0b - No force load 1b - Force load
30-24 —	Reserved Reserved. Read value is undefined; only 0 should be written.
23-0 IVALUE	Time Interval Load Value. $INTVAL_n[IVALUE]$ is loaded into $TIMER_n$, and MRT channel n starts counting down from IVALUE - 1. If the timer is idle, then writing a nonzero value to $INTVAL_n[IVALUE]$ starts the timer immediately. If the timer is running, then writing a 0 to $INTVAL_n[IVALUE]$ does the following: if $LOAD = 1$, then the timer stops immediately; if $LOAD = 0$, then the timer stops at the end of the time interval.

34.6.1.3 Timer (TIMER0 - TIMER3)

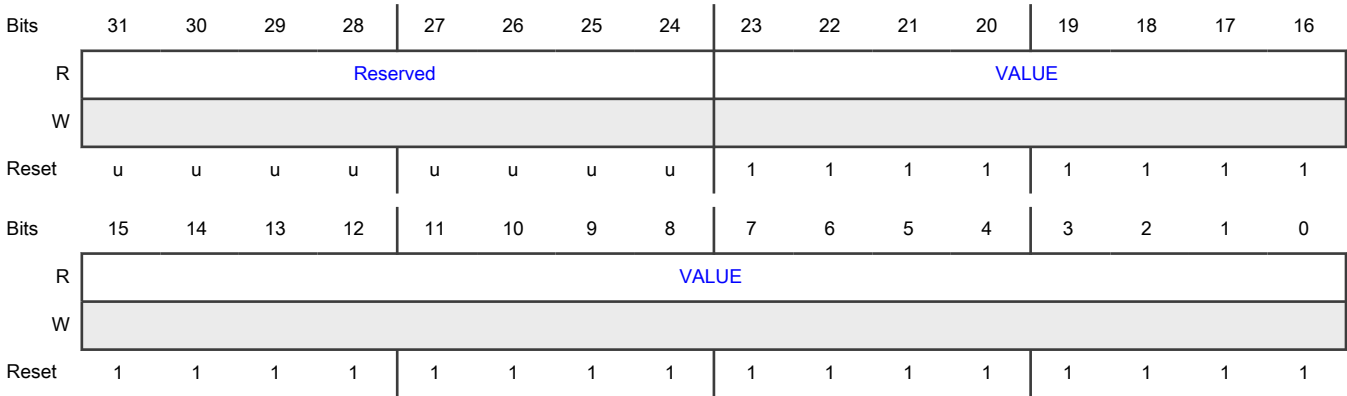
Offset

Register	Offset
TIMER0	4h
TIMER1	14h
TIMER2	24h
TIMER3	34h

Function

Holds the current timer value (the $TIMER_n$ reads the value of the down counter). $TIMER_n$ registers are read only.

Diagram



Fields

Field	Function
31-24	Reserved
—	Reserved. Read value is undefined; only 0 should be written.
23-0	Current Timer Value
VALUE	Holds the current timer value of the down counter. The initial value of $TIMER_n$ is loaded as $(IVALUE - 1)$ from $INTVAL_n$, either at the end of the time interval or immediately in the following cases: $INTVAL_n$ is updated in the idle state or $INTVAL_n$ is updated with $LOAD = 1$. When the timer is in an idle state, reading $TIMER_n[VALUE]$ fields returns -1 (00FF FFFFh).

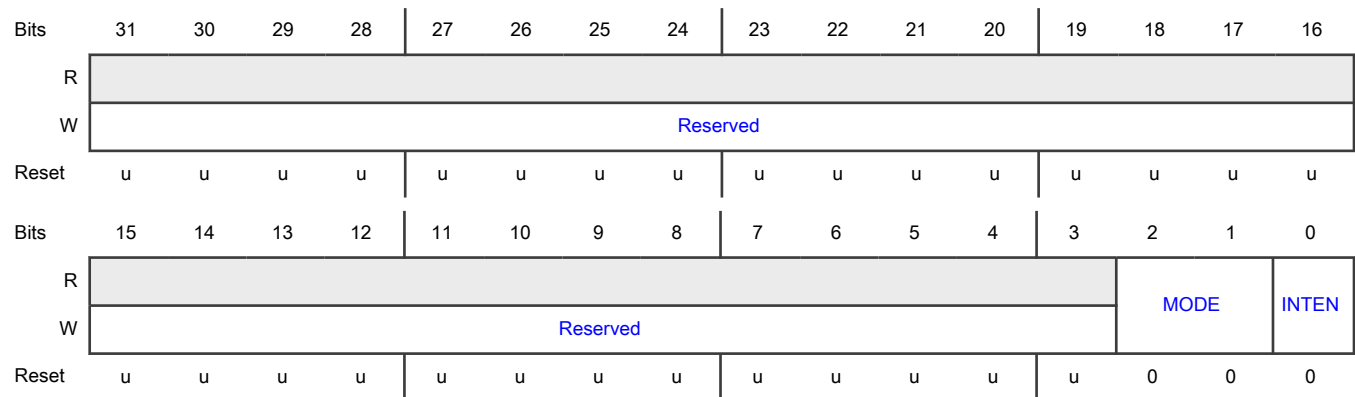
34.6.1.4 Control (CTRL0 - CTRL3)

Offset

Register	Offset
CTRL0	8h
CTRL1	18h
CTRL2	28h
CTRL3	38h

Function

Controls the modes and enables the interrupt for each MRT.

Diagram**Fields**

Field	Function
31-3 —	Reserved
2-1 MODE	MRT Operating mode Selects the timer mode. 00b - Repeat Interrupt mode 01b - One-Shot Interrupt mode 10b - One-Shot Stall mode 11b - Reserved
0 INTEN	Interrupt request Enables the $TIMER_n$ interrupt. 0b - Disabled 1b - Enabled

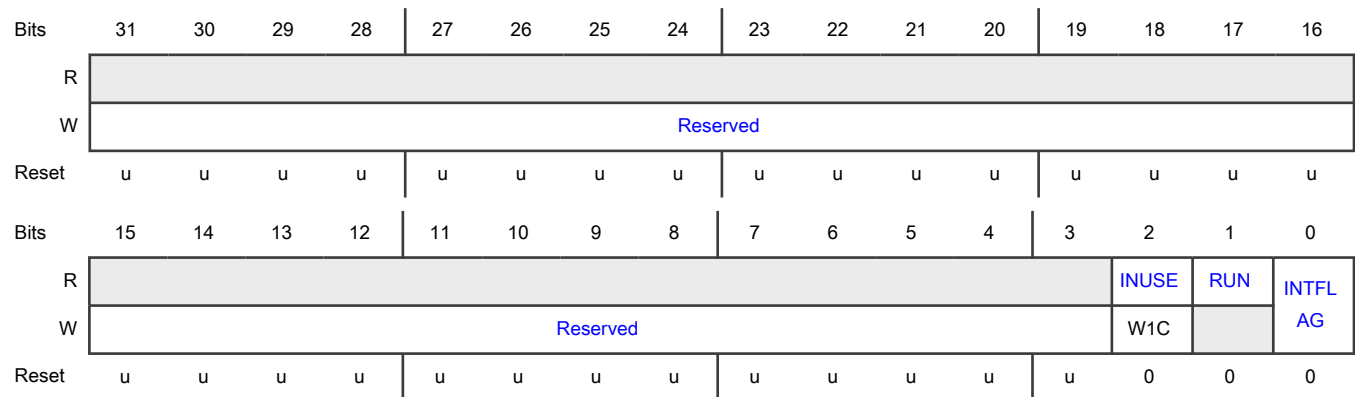
34.6.1.5 Status (STAT0 - STAT3)**Offset**

Register	Offset
STAT0	Ch
STAT1	1Ch
STAT2	2Ch
STAT3	3Ch

Function

Indicates the status of each MRT timer channel.

Diagram



Fields

Field	Function
31-3 —	Reserved
2 INUSE	Channel-In-Use flag Affects the use of Idle Channel (IDLE_CH); the operating details depend on the operating mode bit in MODCFG[MULTITASK]. Writing 1 to this bit clears the status. 0b - This timer channel is not in use. 1b - This timer channel is in use.
1 RUN	Timer n State Indicates the state of TIMERn. STATn[RUN] is read only. TIMERn stops in idle state and runs in running state. 0b - Idle state. 1b - Running.
0 INTFLAG	Monitors the interrupt flag. Writing 0 indicates no pending interrupt or no operation. Writing 1 indicates pending interrupt, because TIMERn has reached the end of the time interval. If CTRLn[INTEN] in the CONTROLn also gets 1, then the interrupt for timer channel n and the global interrupt are generated. Writing 1 to this field bit clears the interrupt request. 0b - No pending interrupt. 1b - Pending interrupt.

34.6.1.6 Module Configuration (MODCFG)

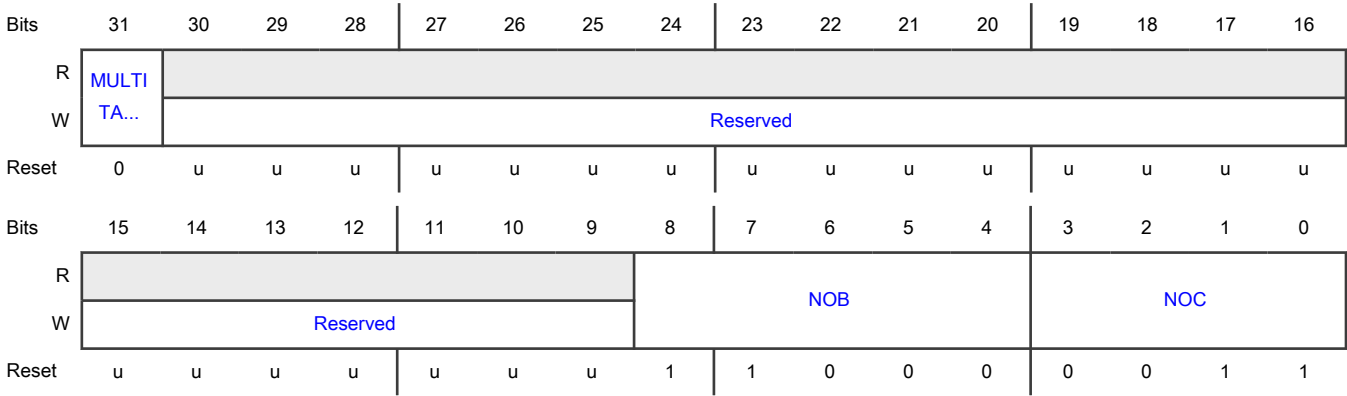
Offset

Register	Offset
MODCFG	F0h

Function

Provides information about this particular MRT instance. It also allows selecting an overall mode for the idle channel feature. [MODCFG\[MULTITASK\]](#) provides the configuration (number of channels and timer width) for this MRT. A reset of the INUSE(n) flags for all channels happens in Hardware Status mode.

Diagram



Fields

Field	Function
31 MULTITASK	MULTITASK Selects the operating mode for STAT_n[INUSE] and Idle Channel (IDLE_CH) . 0b - Hardware status mode. 1b - Multitask mode
30-9 —	Reserved Read value is undefined; only 0 should be written.
8-4 NOB	Number of Bits Identifies the number of timer bits in this MRT (24 bits on this chip).
3-0 NOC	Number of Channels Identifies the number of channels in this MRT (Minus 1 encoded).

34.6.1.7 Idle Channel (IDLE_CH)

Offset

Register	Offset
IDLE_CH	F4h

Function

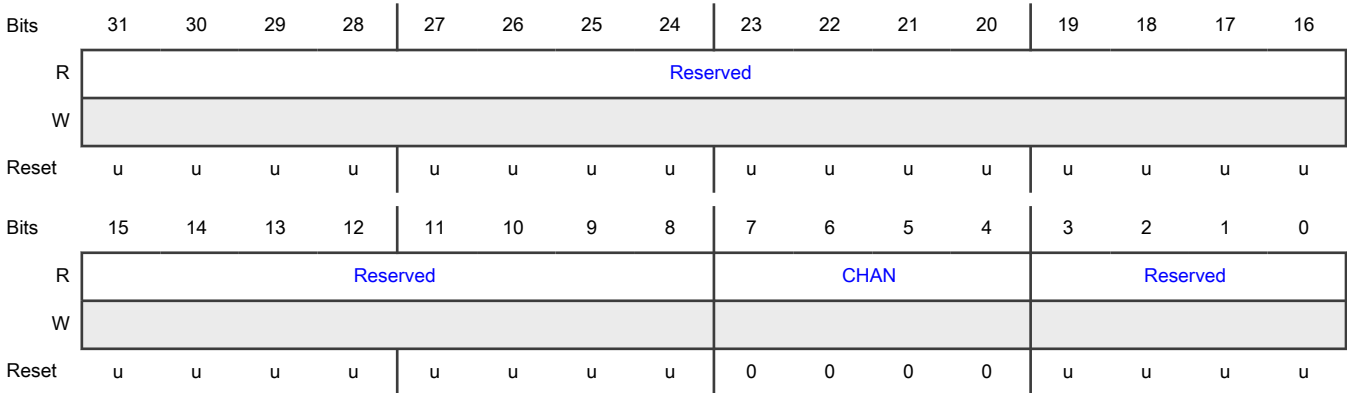
Returns the number of the first Idle Channel.

Use [Idle Channel \(IDLE_CH\)](#) for finding available channels in MRT. This allows you more flexibility when using MRT—no fixed channel assignments are required, and you do not need to search for an available channel. Generally, [Idle Channel \(IDLE_CH\)](#) returns the lowest available channel number.

[MODCFG\[MULTITASK\]](#) allows you to use [Idle Channel \(IDLE_CH\)](#) in two ways. [MODCFG\[MULTITASK\]](#) affects both the functions of [Idle Channel \(IDLE_CH\)](#), and the function of [STATn\[INUSE\]](#) for each MRT channel:

- MULTITASK = 0: Hardware status mode. The [STATn\[INUSE\]](#) for all MRT channels are reset. [Idle Channel \(IDLE_CH\)](#) returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: Multitask mode. In Multitask mode, the [STATn\[INUSE\]](#) allow more control over when MRT channels are released for further use. When [Idle Channel \(IDLE_CH\)](#) is read, returning a channel number of an idle channel, the [STATn\[INUSE\]](#) for that channel is set by hardware. That channel is not considered idle until its RUN flag = 0, and there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and there is no need to start the channel before that channel is no longer considered idle by [Idle Channel \(IDLE_CH\)](#). It also allows you to identify a specific MRT channel. Then you can use that channel more than once without releasing it, removing the need to ask for an available channel for every use.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-4 CHAN	Idle Channel Helps in reading IDLE_CH[CHAN] and returns the lowest idle timer channel. The number is positioned so that you can use it as an offset from MRT base address. This provides access to the registers for the allocated channel. If all timer channels are running, then CHAN = Fh.
3-0 —	Reserved

34.6.1.8 Global Interrupt Flag (IRQ_FLAG)

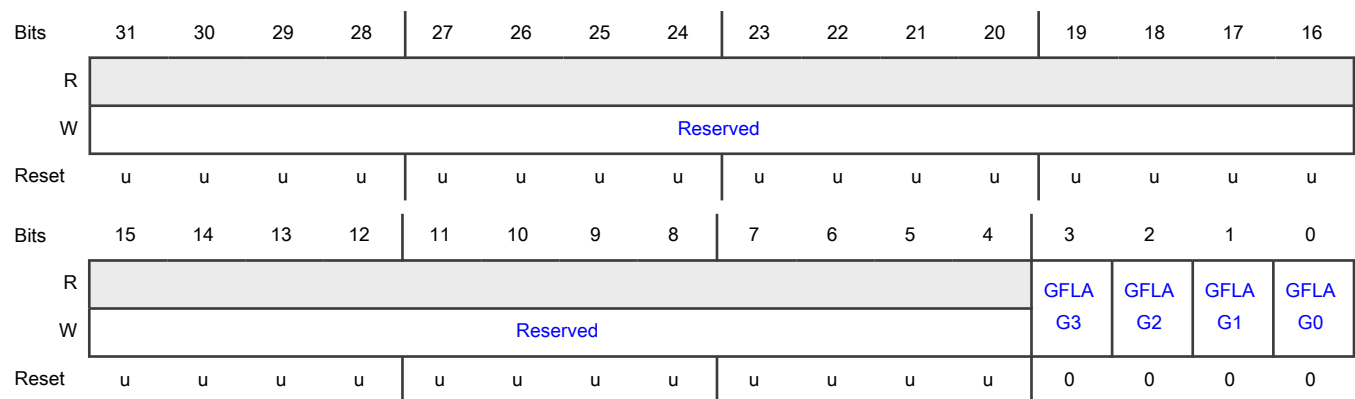
Offset

Register	Offset
IRQ_FLAG	F8h

Function

Combines the interrupt flags from the individual timer channels into one register. Setting and clearing each flag behaves in the same way as setting and clearing [STAT \$n\$ \[INTFLAG\]](#) in each of the STATUS n registers.

Diagram



Fields

Field	Function
31-4	Interrupt Flag
—	Reserved. Read value is undefined; only 0 should be written.
3 GFLAG3	Interrupt Flag Monitors the interrupt flag of TIMER3 and acts similar to channel 0.
2 GFLAG2	Interrupt Flag Monitors the interrupt flag of TIMER2 and acts similar to channel 0.
1 GFLAG1	Interrupt Flag Monitors the interrupt flag of TIMER1 and acts similar to channel 0.
0 GFLAG0	Interrupt Flag Monitors the interrupt flag of TIMER0. Writing 0 is equivalent to no operation. In the pending interrupt state, Timer (TIMER0 - TIMER3) has reached the end of the time interval. If CTRLn[INTEN] in CONTROL n also gets 1, then there is a timer channel n interrupt and a global interrupt. Writing 1 to IRQ_FLAG[GFLAG0] clears the interrupt request.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No pending interrupt. 1b - Pending interrupt

Chapter 35

Windowed Watchdog Timer (WWDT)

35.1 Chip-specific WWDT information

Table 407. Reference links to related information

Topic	Related module	Reference
Full description	WWDT	WWDT
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

The WWDT clock source is external to the block and depends on the system configuration. See the System Configuration for more information, PSCCTL2[WWDT0] in CLKCTL0 registers.

In all power modes, except Deep Power-down mode, a Watchdog reset or interrupt can occur when the WWDT is running and has an operating clock source. The Watchdog oscillator can be configured to keep running in Sleep and Deep-sleep modes. If a Watchdog interrupt occurs in Sleep or Deep-sleep mode, and the WWDT interrupt is enabled in the Nested Vectored Interrupt Controller (NVIC), the device will wake up. Note that in Deep-sleep mode, both the WWDT interrupts in the STARTEN0 register and in the NVIC must be enabled.

NOTE

The Watchdog function is clocked by a 1 MHz oscillator.

35.1.1 Module instance

The chip includes two watchdog timers. WWDT0 is for the Cortex-M33, WWDT1 is for the DSP. Functionally, both watchdog timers are identical.

Both watchdog timers are able to generate an interrupt to the Cortex-M33. This allows the M33 to monitor the DSP watchdog.

The M33 watchdog timer is able to reset the entire device if set up to do so. The DSP watchdog timer can only reset the DSP system.

35.1.2 Initialization

Initialize the WWDT as follows:

- Turn on and configure the LPOSC, PDRUNCFG0[LPOSC_PD] in SYSCTL0 register section, and select LPOSC as clock source for WWDT in CLKCTL registers.
- Enable the register interface (WDT bus clock), PSCCTL2[WWDT] CLKCTL0 register section.
- Enable the Watchdog interrupt for wake-up in the STARTEN0 register.

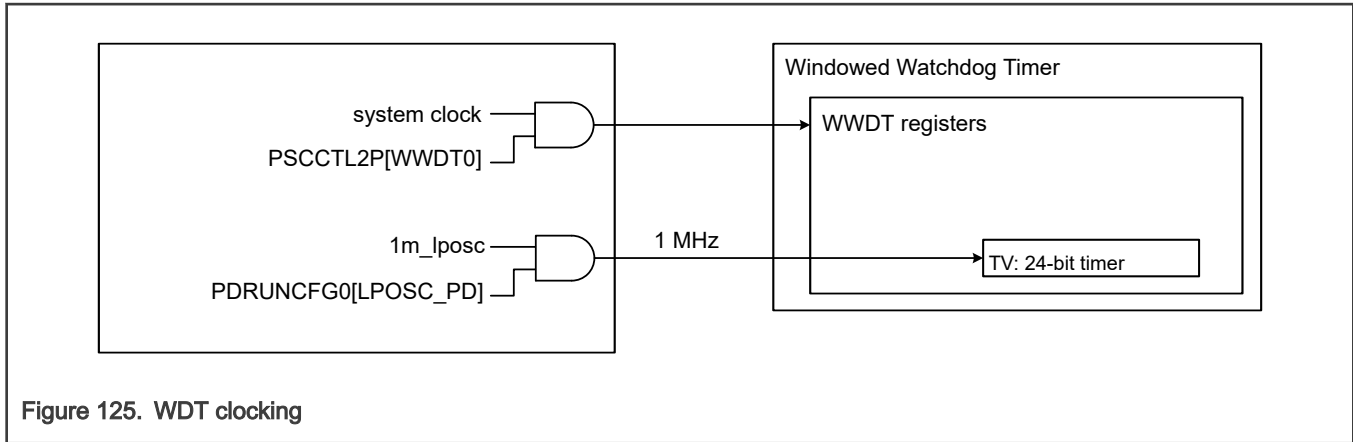


Figure 125. WDT clocking

35.1.3 Debug mode

Clocks to the watchdog timers are automatically stopped when the related processor is halted during chip debug mode:

- When the Cortex-M33 is in debug halt mode, the clock to WWDT0 will be gated, and the WWDT0 TV counter will be stopped. This clock and the counter will resume when the Cortex-M33 leave the debug halt mode.
- When the DSP is in the debug halt mode, the clock to WWDT1 will be gated, and the WWDT1 TV counter will be stopped. This clock and counter will resume when the DSP leaves the debug halt mode.

35.2 Overview

If a microcontroller (or core) enters an erroneous state, a Watchdog timer can reset or interrupt the microcontroller within a programmable time. If an application fails to reload (also called feed) a Watchdog timer within a prespecified amount of time, a Watchdog reset (if enabled) is generated.

35.2.1 Block diagram

The Windowed Watchdog Timer block diagram is shown below. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram. (See the System configuration chapter.)

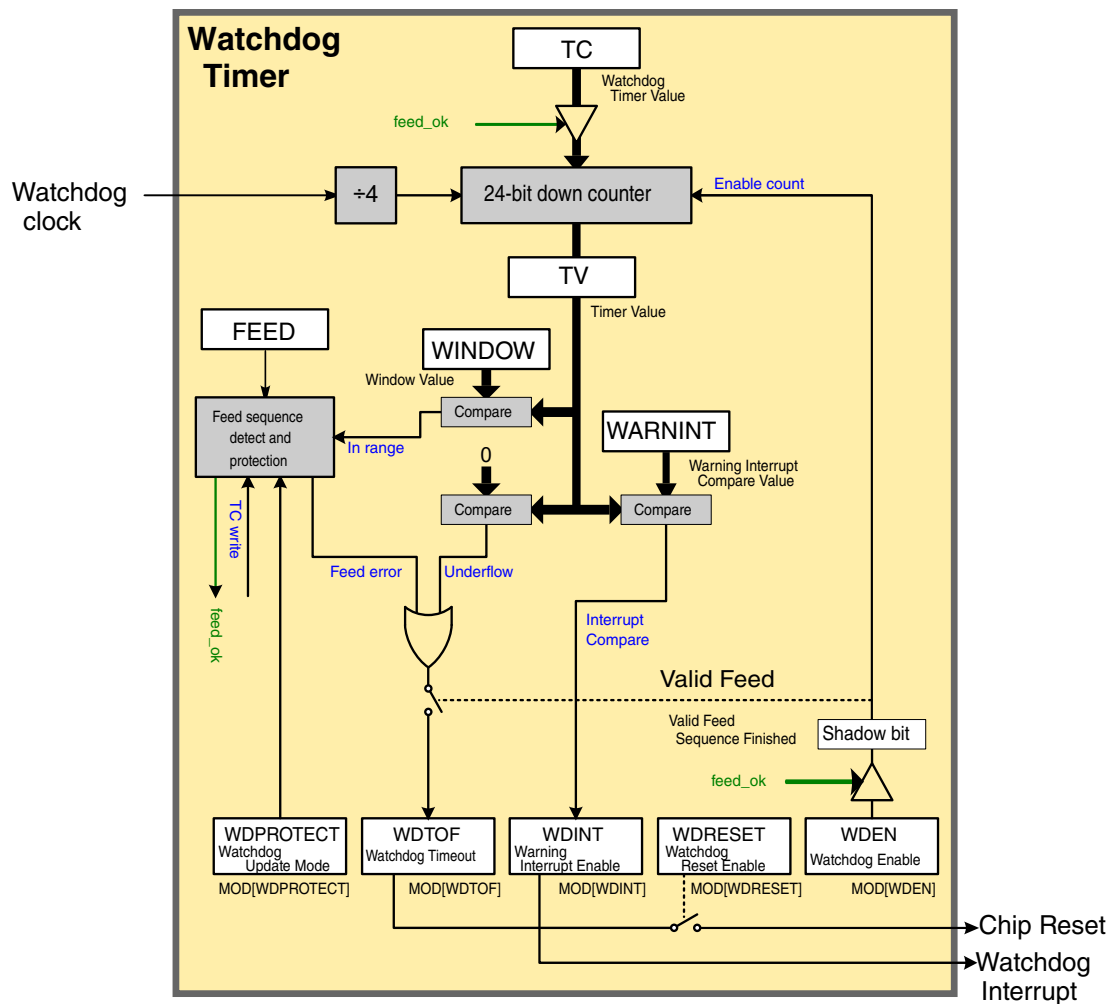


Figure 126. Windowed Watchdog Timer block diagram

35.2.2 Features

- Incorrect feed sequence causes immediate Watchdog event if enabled.
- Watchdog reset indication flag
- Some WWDT are configurable to reset the chip if not reloaded during the programmable timeout period. Other WWDT are configurable to reset a core and generate interrupts. See Chip-specific information.
- Clocking
 - Programmable 24-bit timer with internal fixed pre-scaler
 - Selectable time period from 1,024 Watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million Watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 Watchdog clocks
 - Watchdog clock (WDCLK) source is a selectable frequency in the range of 6 kHz to 1.5 MHz
- WWDT can be configured to run in Deep-sleep mode
- Debug mode
- Optional features

- Optional windowed operation requires reload to occur between programmable minimum and maximum timeout periods.
- Optional warning interrupt can be generated at a programmable time prior to Watchdog timeout.
- Watchdog reload value can optionally be protected so it can only be changed after the warning interrupt time is reached.
- Optional safe operation, which requires a hardware reset or a Watchdog reset to be disabled.

35.3 Functional description

The following sections describe functional details of the WWDT module.

When a Watchdog window is programmed (a timing window), an early Watchdog reload is also treated as a Watchdog event, which prevents situations where a system failure may still reload the Watchdog timer. For example, application code could be stuck in an interrupt service that contains a Watchdog reload (feed). Setting the timing window such that this situation would cause an early Watchdog timer reload, which then would generate a Watchdog event, and thereby enable the system to recover from this situation.

The WWDT consists of a fixed (divide by 4) pre-scaler and a 24-bit counter (which decrements when clocked). The minimum value is 0xFF; setting a value lower than 0xFF causes 0xFF to be loaded in the counter.

- Minimum Watchdog interval is $(T_{WDCLK} \times 256 \times 4)$
- Maximum Watchdog interval is $(T_{WDCLK} \times 2^{24} \times 4)$ in multiples of $(T_{WDCLK} \times 4)$.

Use WWDT to:

- Enable and configure the Watchdog oscillator. (See [Initialization](#))
- Set the Watchdog timer constant reload value in the [Timer Constant \(TC\)](#) register.
- Set the Watchdog timer operating mode in the [Mode \(MOD\)](#) register.
- Set a value for the Watchdog window time in the [Window Compare Value \(WINDOW\)](#) register if windowed operation is desired.
- Set a value for the Watchdog warning interrupt in the [Warning Interrupt Compare Value \(WARNINT\)](#) register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the [Feed Sequence \(FEED\)](#) register.

To prevent a Watchdog event, the WWDT must be fed (reloaded) again before the Watchdog counter reaches 0. If a WINDOW value is programmed, then the feed (reload) must also occur after the Watchdog counter passes that programmed WINDOW value.

When the Watchdog timer is configured (MOD[WDRESET]) so that a Watchdog event causes a reset, or when the counter reaches 0 (it causes a reset), then the microcontroller (or core) is also reset (loading the stack pointer and program counter from the vector table, just like what happens for an external reset).

35.3.1 Operating modes

This section describes all functional operation modes of the WDT module: Disabled, Watchdog Interrupt, Watchdog Reset.

Table 408. Watchdog operating modes selection

MOD[WDEN]	MOD[WDRESET]	Mode	Mode description
0	X (0 or 1)	Disabled	Debug/Operate without the Watchdog running.
1	0	Watchdog Interrupt	The Watchdog warning interrupt generates but the Watchdog reset does not.

Table continues on the next page...

Table 408. Watchdog operating modes selection (continued)

MOD[WDEN]	MOD[WDRESET]	Mode	Mode description
			A Watchdog counter equal to WARNINT sets the WDINT flag and generates a Watchdog interrupt request.
1	1	Watchdog Reset	Both the Watchdog Interrupt and Watchdog Reset are enabled. • A Watchdog counter equal to WARNINT sets the WDINT flag and generates a Watchdog interrupt request. • A Watchdog counter equal to zero resets the microcontroller. • A Watchdog feed prior to reaching the value of WINDOW also causes a Watchdog reset.

35.3.2 Example timing diagrams

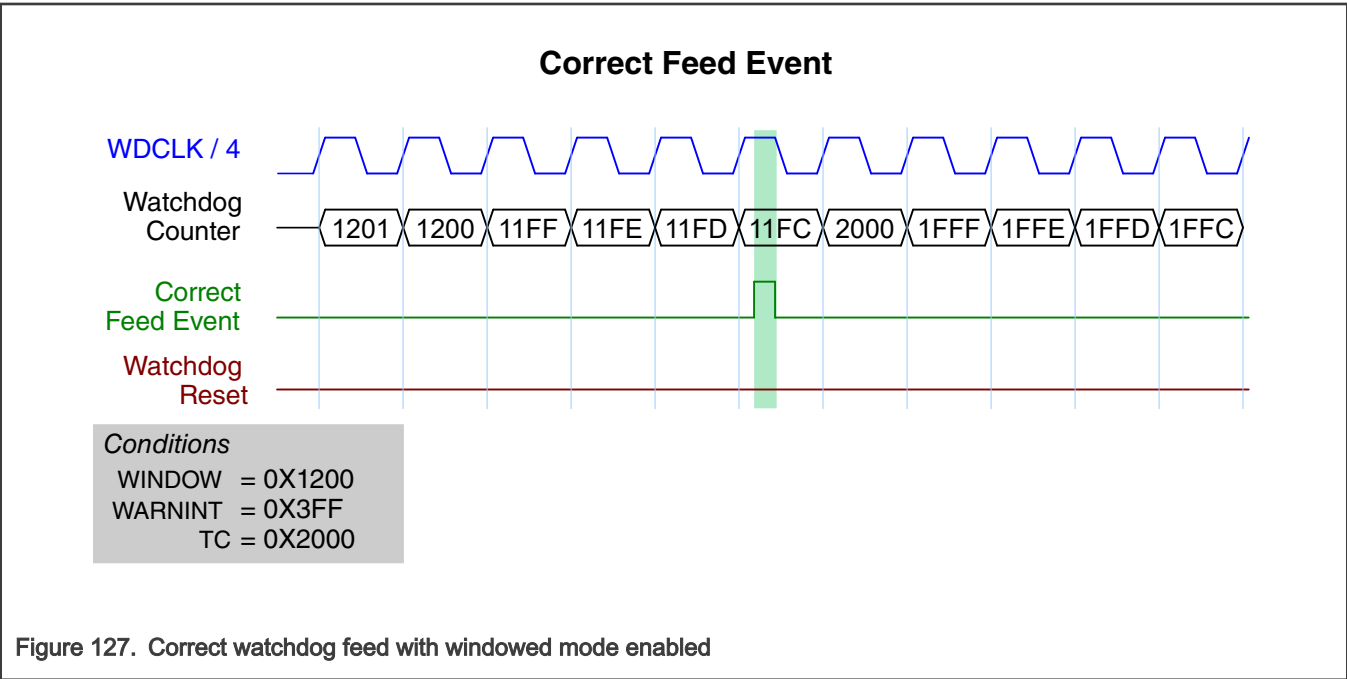
The following example timing diagrams show the WWDT in operation.

A feed is correct when both of the following conditions are true:

- A valid feed sequence completes, writing 0xAA followed by 0x55 to the FEED register.
- The TV register value is not greater than the WINDOW register value.

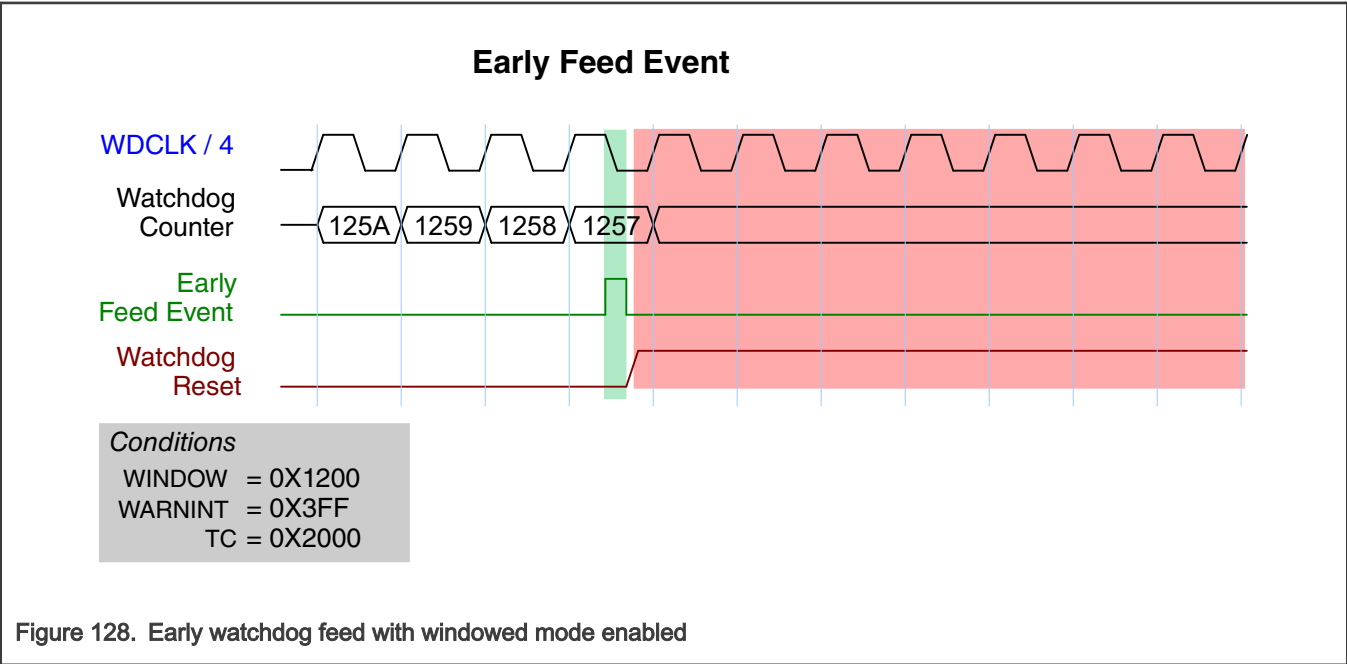
If either of these conditions is not true, a feed error occurs.

In the Correct watchdog feed with windowed mode enabled figure, the sequence of writing 0xAA followed by 0x55 occurs when the counter value (0x11FC) is not greater than WINDOW register value (0x1200). It is a correct feed (reload) event. After the correct feed event occurs, the watchdog counter is reloaded with the TC register value (0x2000). In addition to the above two conditions, when MOD[WDPROTECT] = 1, the watchdog counter must not be greater than the WARNINT register value for a correct feed to occur.



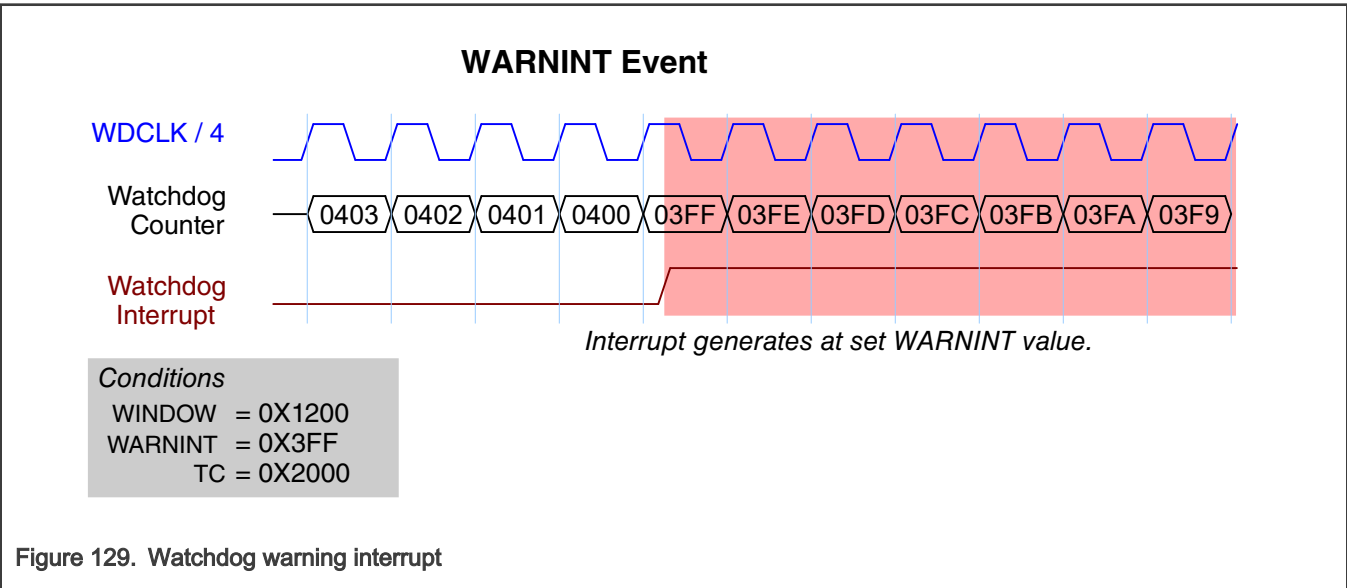
The Early watchdog feed with windowed mode enabled figure shows a reset being triggered because software fed the watchdog too early.

An early feed event occurs if the feed sequence happens when the watchdog counter value (0x1257) is greater than the WINDOW register value (0x1200). This is a feed error and can generate a reset if MOD[WDRESET] = 1.



The Watchdog warning interrupt figure below shows the WWDT generating a warning.

A correct feed sequence (done by software) can only happen when the watchdog counter value (TV register) is not greater than the WINDOW value (0x1200). Otherwise, a feed error occurs.



35.3.3 Clocks

The WWDT uses two clocks: APB bus clock and WDCLK.

- The APB bus clock is used for the APB accesses to the Watchdog registers and is derived from the system clock.

- The WDCLK is used for the Watchdog timer counting and is derived from the Watchdog oscillator.

The synchronization logic between the two clock domains works as follows:

- When the MOD and TC registers are updated by APB operations, the new value takes effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.
- When the Watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

NOTE

Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time software enables MOD[WDPROTECT]. The length of the delay depends on the selected Watchdog clock, WDCLK.

35.3.4 Using WWDT lock

The WWDT supports lock features that can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

35.3.4.1 Disabling the WWDT clock source

If MOD[LOCK] = 1, the WWDT clock source is locked and cannot be disabled either by software or by hardware when entering Sleep or Deep-sleep modes. Therefore, enable the Watchdog oscillator for each Power mode before setting MOD[LOCK] = 1.

35.3.4.2 Changing the WWDT reload value

If the Watchdog Update Mode MOD[WDPROTECT] = 1 (Threshold), then any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets MOD[WDTOF] = 1.

Any type of reset disables the reload overwrite lock mechanism.

35.4 Initialization

Initialize the WWDT as follows:

- Turn on and configure the Watchdog oscillator and the Watchdog oscillator control register.
- Enable the register interface
- Enable the Watchdog interrupt for wake-up.

35.5 Memory map and register definition

This section includes the WWDT module memory map and detailed descriptions of all registers.

35.5.1 WWDT register descriptions

35.5.1.1 WWDT memory map

WWDT0 base address: 4000_E000h

WWDT1 base address: 4002_E000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Mode (MOD)	32	RW	0000_0000h
4h	Timer Constant (TC)	32	RW	0000_00FFh
8h	Feed Sequence (FEED)	32	W	See section
Ch	Timer Value (TV)	32	R	0000_00FFh
14h	Warning Interrupt Compare Value (WARNINT)	32	RW	0000_0000h
18h	Window Compare Value (WINDOW)	32	RW	00FF_FFFFh

35.5.1.2 Mode (MOD)

Offset

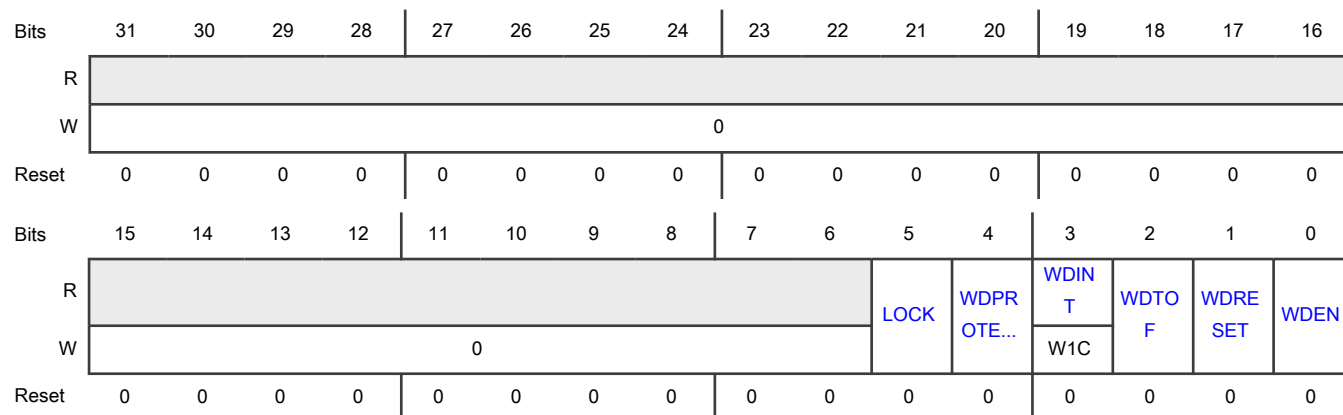
Register	Offset
MOD	0h

Function

The MOD register controls the operation of the WWDT. Note that you must perform a Watchdog feed before any changes to the MOD register takes effect.

After WDEN, WDPROTECT, or WDRESET are set, they cannot be cleared by software, but are cleared by an external reset or a Watchdog timer reset.

Diagram



Fields

Field	Function
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined, only zero should be written.
5 LOCK	Lock After LOCK is set to one and a Watchdog feed is performed, hardware prevents disabling or powering down the Watchdog oscillator. LOCK can be set once by software and is only cleared by any reset. 0b - No Lock 1b - Lock
4 WDPROTECT	Watchdog Update Mode WDPROTECT can be set once by software and is only cleared by a reset. 0b - Flexible. The Timer Constant (TC) can be changed at any time. 1b - Threshold. The Timer Constant (TC) can be changed only after the counter is less than the value of WARNINT and WINDOW. Any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets MOD[WDTOF] = 1.
3 WDINT	Warning Interrupt Flag Set when the timer is at or below the value in WARNINT. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT and WDRESET are both 0 when TV decrements to 0. When the Watchdog timer is configured to generate WDINT, the interrupt occurs when the counter is no longer greater than the value defined by the WARNINT register. Any reset clears this flag. Software clears WDINT by writing a 1. 0b - No flag. 1b - Flag. The Watchdog interrupt flag is set when the Watchdog counter is no longer greater than the value specified by WARNINT.
2 WDTOF	Watchdog Timeout Flag Flag set for these events: • The Watchdog timer times out. • There is a feed error. • When WDPROTECT = 1 and an attempt is made to write to TC. To determine if a Watchdog event has caused the reset condition, examine WDTOF. The flag must be cleared by software by software writing a 0 to WDTOF. NOTE WDTOF resets to 0x0 only by an external or a power-on reset. 0b - Clear. 1b - Reset. Causes a chip reset if WDRESET = 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 WDRESET	Watchdog Reset Enable After WDRESET has been written with a 1, WDRESET cannot be re-written with a 0. 0b - Interrupt. A Watchdog timeout will not cause a chip reset. 1b - Reset. A Watchdog timeout will cause a chip reset.
0 WDEN	Watchdog Enable After setting WDEN to one, perform a Watchdog feed (reload) to enable the Watchdog timer. 0b - Stop. The Watchdog timer is stopped. 1b - Run. The Watchdog timer is running.

35.5.1.3 Timer Constant (TC)

Offset

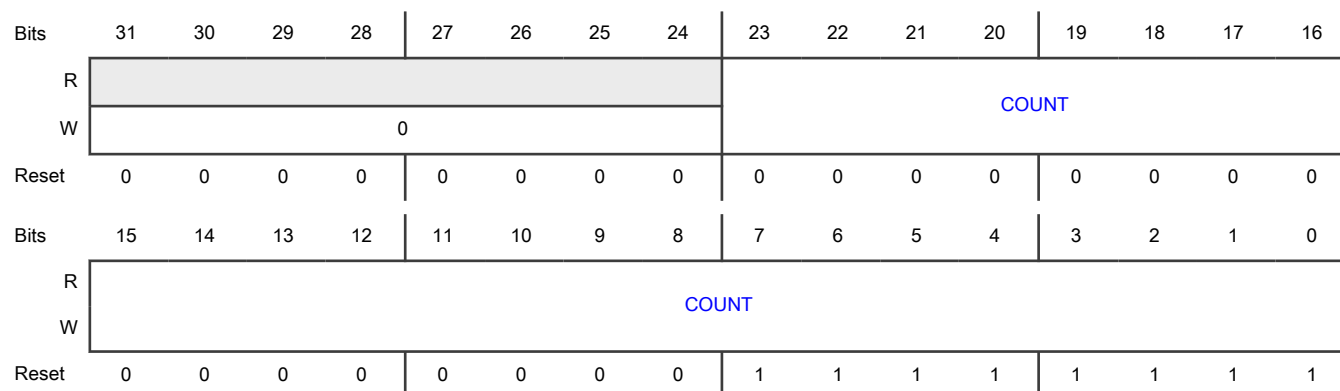
Register	Offset
TC	4h

Function

The TC register specifies the desired timeout value. Transferring the TC value into the Watchdog counter requires a feed sequence. The TC resets to 0x00_00FF. Writing a value below 0xFF causes 0x00_00FF to load into the TC. Therefore the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the Watchdog Update Mode ([MOD\[WDPROTECT\]](#)) = 1 (Threshold), then any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets [MOD\[WDTOF\]](#) = 1.

Diagram



Fields

Field	Function
31-24	Reserved
—	Read value is undefined, only zero should be written.
23-0 COUNT	Watchdog Timeout Value

35.5.1.4 Feed Sequence (FEED)

Offset

Register	Offset
FEED	8h

Function

Writing 0xAA followed by 0x55 to FEED reloads the Watchdog timer with the Timer Constant (TC) register value. This operation also starts WWDT, if the Watchdog timer is enabled using MOD[WDEN] = 1.

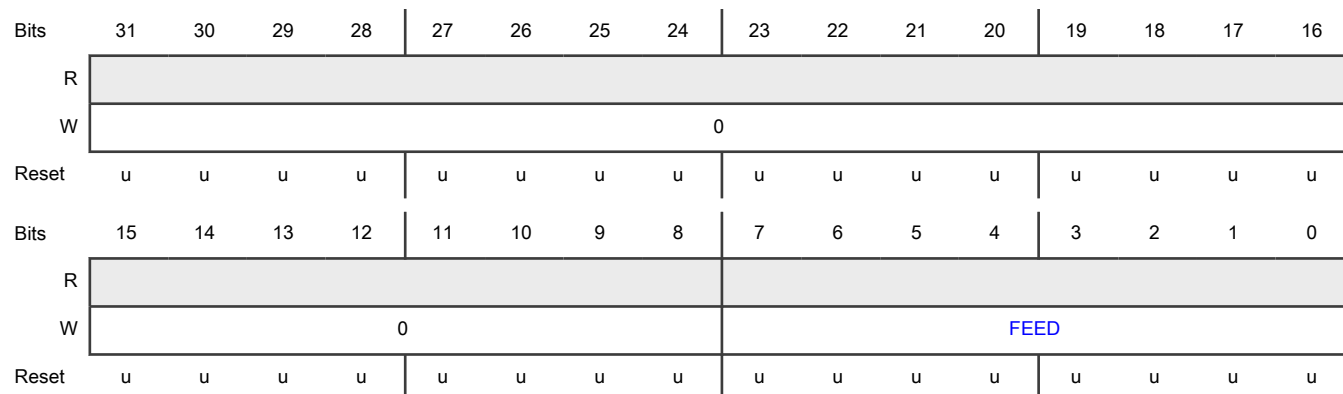
However setting MOD[WDEN] is not sufficient to enable WWDT. Before the watchdog can actually generate a reset, MOD[WDEN] must be set, followed by a valid feed sequence that finishes. Until then, the Watchdog ignores feed errors.

After writing 0xAA to FEED, access to any Watchdog register other than writing 0x55 to the FEED register causes an immediate reset/interrupt when WWDT is enabled, and sets the WDTOF flag. The reset is generated during the second APB bus clock after an incorrect access to a Watchdog register during a feed sequence.

To avoid an unintended interrupt, it is good practice to disable interrupts around a feed sequence. Disable interrupts if they might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WWDT before control is returned to the interrupted task.

If a value in the WINDOW register is smaller than the default, it may limit the time when a Watchdog feed is allowed.

Diagram



Fields

Field	Function
31-8	Reserved
—	Read value is undefined, only zero should be written.
7-0	Feed Value
FEED	Feed value should be 0xAA followed by 0x55.

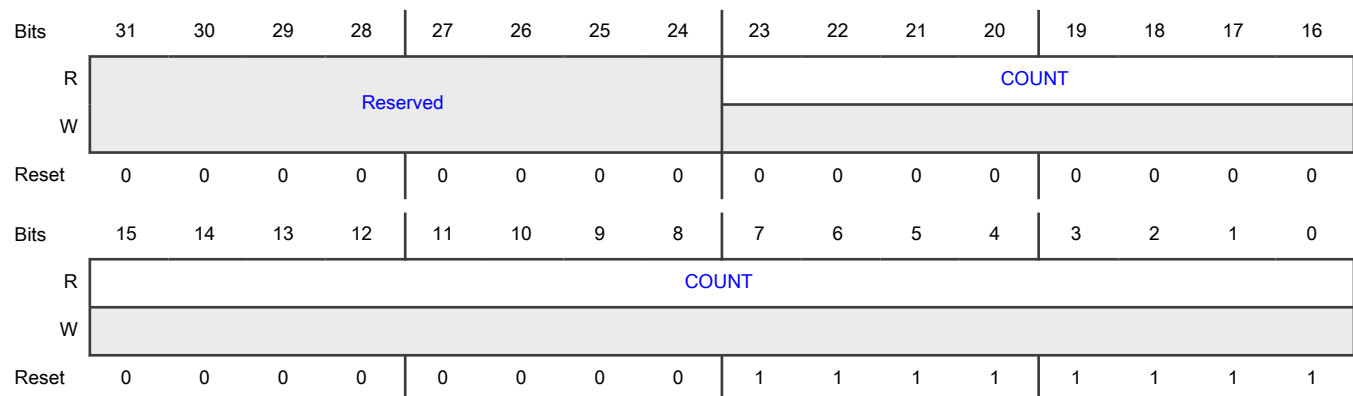
35.5.1.5 Timer Value (TV)**Offset**

Register	Offset
TV	Ch

Function

The Timer Value (TV) register reads the current value of the Watchdog timer counter.

When the TV value is read by the core (or CPU), the TV value is older than the actual value of the timer. Reading the timer using the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles.

Diagram**Fields**

Field	Function
31-24	Reserved
—	Read value is undefined, only zero should be written.
23-0	Counter Timer Value
COUNT	

35.5.1.6 Warning Interrupt Compare Value (WARNINT)

Offset

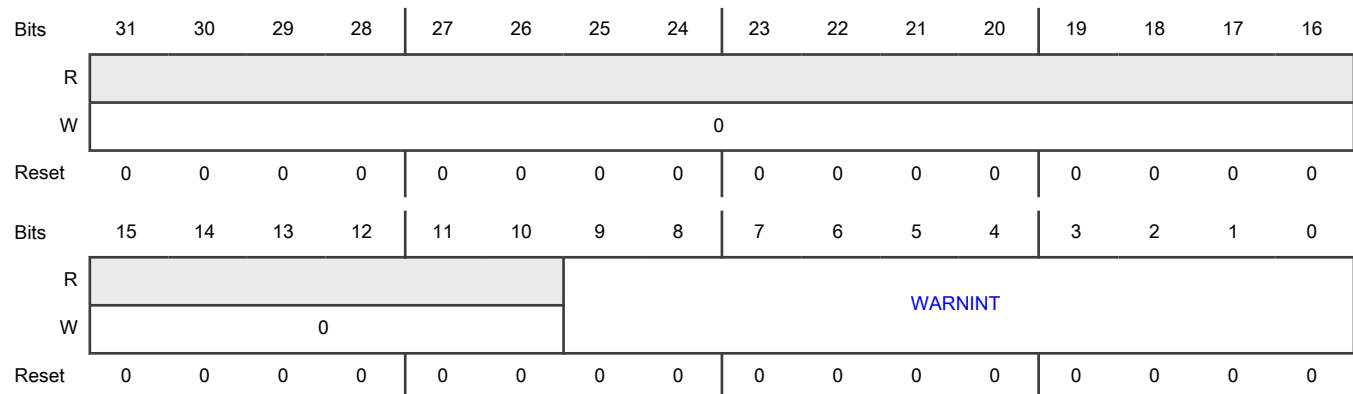
Register	Offset
WARNINT	14h

Function

The WARNINT register determines the Timer Value (TV) that generates a Watchdog interrupt. When TV[COUNT] is less than or equal to the value defined by WARNINT, an interrupt generates after the subsequent WDCLK.

A match of the Watchdog timer counter to WARNINT occurs when the bottom 10 bits of the timer counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 Watchdog timer counts (4,096 Watchdog clocks) for the interrupt to occur prior to a Watchdog event. If WARNINT is 0, then the interrupt occurs at the same time as the Watchdog event.

Diagram



Fields

Field	Function
31-10	Reserved
—	Only zero should be written.
9-0 WARNINT	Watchdog Warning Interrupt Compare Value

35.5.1.7 Window Compare Value (WINDOW)

Offset

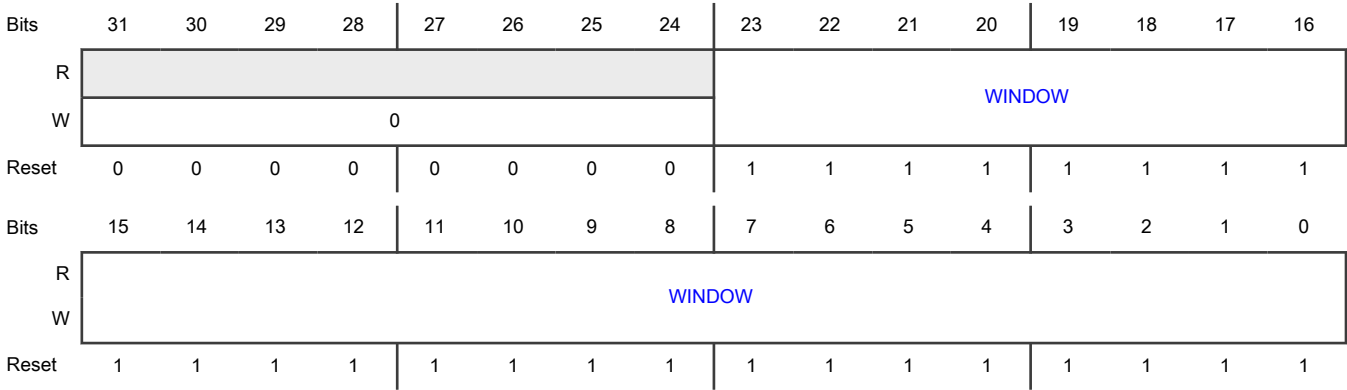
Register	Offset
WINDOW	18h

Function

The WINDOW register determines the highest timer value (TV[COUNT]) allowed during a Watchdog feed. If a feed sequence occurs when TV is greater than the value in WINDOW, then a Watchdog event occurs.

WINDOW resets to the maximum possible TV value, where windowing is not in effect.

Diagram



Fields

Field	Function
31-24	Reserved
—	Only zero should be written.
23-0 WINDOW	Watchdog Window Value.

Chapter 36

Frequency Measurement (FREQMEASURE)

36.1 Chip-specific Frequency Measurement information

Table 409. Reference links to related information

Topic	Related module	Reference
Full description	FREQMEASURE	Frequency Measurement
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

These Frequency Reference clock select and target clock select register choices are available in FMEASURE_CH0_SEL[FMEASURE_SEL] and FMEASURE_CH1_SEL[FMEASURE_SEL] (See INPUTMUX register descriptions):

- OSC_CLK
- Low Power Oscillator Clock (LPOSC)
- RTC 32 kHz OSC
- Main SYSCLK
- FREQME_GPIO_CLK
- FRO_DIV16
- FRO_DIV1
- Clock Out

36.1.1 Module instances

This device has one instance of the Frequency Measurement module.

36.1.2 Chip-specific initialization

To initialize the Frequency Measurement module:

- Enable the clock to the Frequency Measurement function (PSCCTL1_SET[FREQME] = 1) in CLKCTL1 registers. This enables the register interface and the peripheral function clock.
- Clear the Frequency Measurement function peripheral reset (PRSTCTL1_CLR[FREQME] = 1) in RSTCTL1 registers.
- Use the IOPCTL registers to connect the FREQME_GPIO_CLK input (if it is used) to an external pin.
- Make sure the clocks used as the reference and target are enabled.
- Select reference and target clocks using input mux block.
 - Reference clock selection: FMEASURE_CH0_SEL[FMEASURE_SEL]
 - Target clock selection: FMEASURE_CH1_SEL[FMEASURE_SEL]

36.2 Overview

Find the frequency of any on-chip or off-chip clock signal with the Frequency Measurement module. Select a clock frequency to measure against a selectable reference clock to get an accurate frequency. For example, use the Frequency Measurement function to accurately determine the frequency of the low-power oscillator which varies depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block.

36.2.1 Features

- High-accuracy Frequency Measurement function for on-chip and off-chip clocks
- Reference and target inputs selectable from among various chip-specific channels
- Pulse Width Measurement (PWM) mode

36.3 Functional description

The following sections describe functional details of the Frequency Measurement module.

36.3.1 Frequency measure function

The Frequency Measure circuit is based on two 31-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

Software initiates a measurement cycle by setting the FREQMECTRL[MEASURE_IN_PROGRESS] equal to one ([Frequency Measurement \(in Read mode\) \(FREQMECTRL_R\)](#)). Software can then poll this same bit which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over. At that point the state of the target counter is loaded into a result register, and the FREQMECTRL[MEASURE_IN_PROGRESS] bit equals zero. Software can read this capture value and apply to it a specific calculation which returns the precise frequency of the target clock in MHz.

Also see:

- Frequency reference clock select register
- Frequency target clock select register

This FREQMECTRL register starts the Frequency Measurement function and returns the result in the FREQMECTRL[RESULT] field. The target frequency can be calculated as follows with the frequencies given in MHz:

$$F_{\text{target}} = (\text{RESULT} - 2) \times F_{\text{reference}} / 2^{\text{AREF\_SCALE}}$$

Select the reference and target frequencies. Then start a Frequency Measurement by setting FREQMECTRL[PROG] = 1.

36.3.1.1 Accuracy

The Frequency Measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) with a high degree of accuracy by using another on-chip clock of known frequency as a reference.

Uncertainty in the reference clock (for example the $\pm 1\%$ accuracy of either FRO) adds to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

36.3.2 Pulse measurement function

When FREQMECTRL[PULSE_MODE] = 1, the reference clock is counted while the target input is in a specific state (high or low), selected by the FREQMECTRL[PULSE_POL] bit. See the description of these bits in [Frequency Measurement \(in Write mode\) \(FREQMECTRL_W\)](#).

36.4 Signals

The Frequency Measurement module is not associated with any signals. Although some clock sources may be taken from device signals.

36.5 Memory map and register definition

This section includes the Frequency Measurement module memory map and detailed descriptions of all registers.

36.5.1 Frequency Measurement register descriptions

36.5.1.1 FREQMEASURE memory map

FREQME base address: 4002_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Frequency Measurement (in Read mode) (FREQMCTRL_R)	32	R	0000_0000h
0h	Frequency Measurement (in Write mode) (FREQMCTRL_W)	32	W	0000_0000h

36.5.1.2 Frequency Measurement (in Read mode) (FREQMCTRL_R)

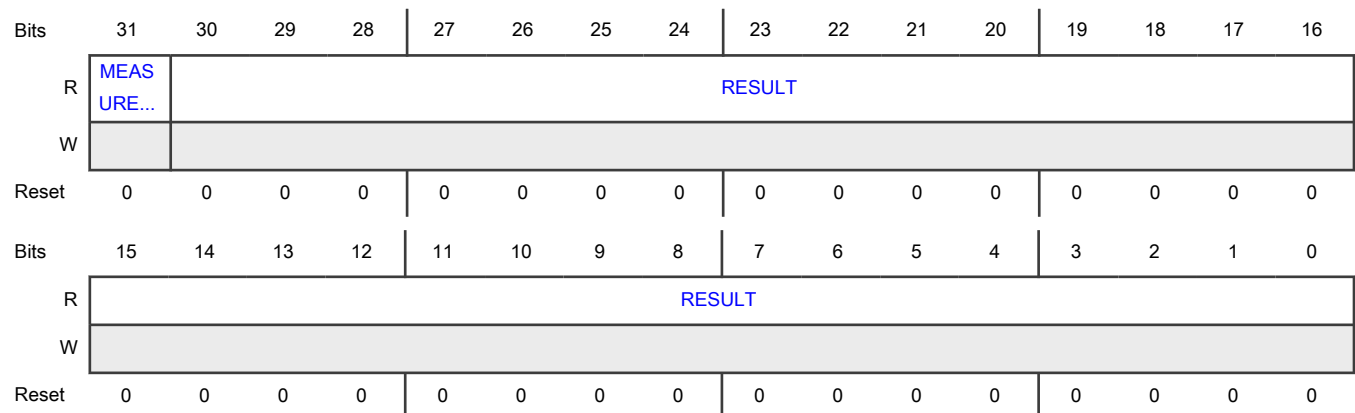
Offset

Register	Offset
FREQMCTRL_R	0h

Function

The FREQMCTRL register has different fields depending on whether it is in Read or Write mode. In Read mode, the FREQMCTRL register provides the status and measurement results.

Diagram



Fields

Field	Function
31	Measure in Progress

Table continues on the next page...

Table continued from the previous page...

Field	Function
MEASURE_IN_PROGRESS	The MEASURE_IN_PROGRESS bit indicates the run status of the frequency or pulse width measurement. 0b - Process complete. Measurement cycle is complete. The results are ready in the RESULT field. 1b - In Progress. Measurement cycle is in progress.
30-0 RESULT	Result Indicates measurement result (target counter value) of frequency or pulse width measurement. This field is valid only when MEASURE_IN_PROGRESS = 0, indicating the measurement process is complete.

36.5.1.3 Frequency Measurement (in Write mode) (FREQMECTRL_W)

Offset

Register	Offset
FREQMECTRL_W	0h

Function

In Write mode the FREQMECTRL register controls and configures the measurement cycle.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	MEAS URE...															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								PULSE _P...		PULSE _M...						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 MEASURE_IN_PROGRESS	Measure in Progress Initiates a frequency measurement cycle

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Force Terminate. Forces the termination of any measurement cycle currently in progress and resets RESULT or just resets RESULT if in idle. 1b - Initiates Measurement Cycle. Initiates frequency or pulse width measurement process. Hardware clears the MEASURE_IN_PROGRESS bit when the measurement cycle completes. A new measurement starts if there is an active measurement in progress.
30-10 —	Reserved
9 PULSE_POL	Pulse Polarity High or low period of reference clock is measured in Pulse Width Measurement mode. PULSE_POL is valid only in Pulse Width Measurement mode. 0b - High Period. High period of reference clock is measured in Pulse Width Measurement mode triggered by the rising edge on the reference clock input. 1b - Low Period. Low period of reference clock is measured in Pulse Width Measurement mode triggered by the falling edge on the reference clock input.
8 PULSE_MODE	Pulse Width Measurement mode select The PULSE_MODE bit selects the operating mode frequency or pulse width measurement. 0b - Frequency Measurement Mode. FREQMECTRL works in a Frequency Measurement mode. Once the measurement starts (real count start is aligned at rising edge arrival on reference clock), the target counter increments by the target clock until the reference counter running by the reference clock reaches the count end point selected by REF_SCALE. 1b - Pulse Width Measurement mode. FREQMECTRL works in a Pulse Width Measurement mode, measuring the high or low period of reference clock input selected by PULSE_POL. The target counter starts incrementing by the target clock once a corresponding trigger edge (rising edge for high period measurement and falling edge for low period) occurs.
7-5 —	Reserved
4-0 REF_SCALE	Reference Clock Scaling Factor Selects the reference clock counter scaling factor in Frequency Measurement mode. The reference count cycle is $2^{\text{ref\_scale}}$. A higher number provides better accuracy but consumes more processing time. This field is valid only in Frequency Measurement mode. 0_0000b - Count cycle = $2^0 = 1$ 0_0001b - Count cycle = $2^1 = 2$ 0_0010b - Count cycle = $2^2 = 4$ 1_1111b - Count cycle = $2^{31} = 2,147,483,648$

Chapter 37

Cyclic Redundancy Check (CRC)

37.1 Chip-specific CRC information

Table 410. Reference links to related information

Topic	Related module	Reference
Full description	CRC	CRC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

37.1.1 Module instances

This device has one instance of the CRC module, CRC_ENGINE.

37.2 Overview

The CRC generator module supports various CRC standards commonly used in the market. Each of these CRC standards uses a solid specific polynomial, but there are some variations in applications, such as using different seed numbers, bit order reverse and 1's complement, and others. The CRC module provides programmability options to support these variations as desired. To save system power and bus bandwidth, DMA transfers are supported, which is more efficient than software programmed I/O operations (PIO) performed by the main core. The CRC generator module is simpler by design, uses less power and has lower device costs.

37.2.1 Block diagram

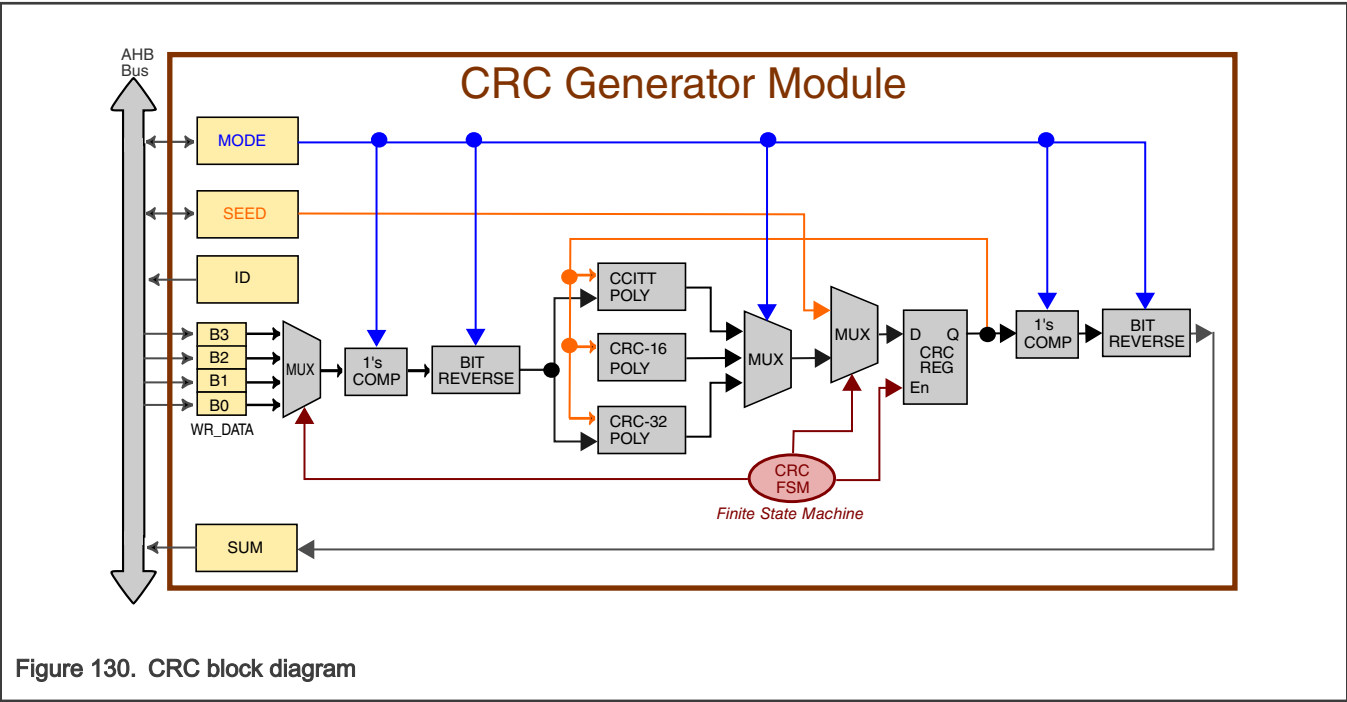


Figure 130. CRC block diagram

37.2.2 Features

CRC module:

- Supports using 3 common polynomials: CRC-CCITT, CRC-16, CRC-32
 - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
 - CRC-16: $X^{16} + X^{15} + X^2 + 1$
 - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$
- Bit-order reverse and 1's complement programmable setting for input data and CRC sum
- Programmable seed number setting
- CPU PIO or DMA back-to-back transfers
- Accepts any size of data wide per write: 8, 16 or 32 bits
 - 8-bit write: 1-cycle operation
 - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
 - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

37.3 Functional description

The following sections describe functional details of the CRC module.

37.3.1 Operations

Before using the CRC module, decisions must be made for which CRC polynomial must be used, and whether to use 1's complement or bit-order reverse for CRC sum calculations and write data operations.

- For CRC polynomial, use one of the following choices:
 - CRC-CCITT polynomial
 - CRC-16 polynomial
 - CRC-32 polynomial
- For CRC sum calculations, use one of the following choices:
 - 1's Complement
 - Bit-order reverse
- For write data operations, use one of the following choices:
 - 1's Complement
 - Bit-order reverse

A CRC seed value must also be provided.

37.4 Signals

The CRC module communicates with other SoC modules using the AHB bus (an internal bus), and has no external signals.

37.5 Application information

This section provides additional application information for the CRC module.

Standard CRC configurations

Sometimes in different CRC applications, there are variations from the common CRC standards, like using a different seed number. The next table shows 3 common standard CRC configurations.

Table 411. Common CRC standard configurations

Parameter	CRC-CCITT	CRC-16	CRC-32
Polynomial	$X^{16} + X^{12} + X^5 + 1$	$X^{16} + X^{15} + X^2 + 1$	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$
Seed Value	0xFFFF	0x0000	0xFFFFFFFF
Bit order reverse for data input	No	Yes	Yes
1's complement for data input	No	No	No
Bit order reverse for CRC sum	No	Yes	Yes
1's complement for CRC sum	No	No	Yes
MODE	0x00000000	0x00000015	0x00000036
SEED	0x0000FFFF	0x00000000	0xFFFFFFFF

Timing considerations

Because the CRC engine uses some time to process data, which depends on how the data is accessed, wait states are added.

- A write followed by another write:
 - For a 16-bit write to the CRC data register, followed by another write to the same register, 1 wait state is added.
 - For a 32-bit write to the CRC data register, followed by another write to the same register, 3 wait states are added.
- A write followed by a read:
 - For an 8-bit write to the CRC data register, followed by a read of the CRC checksum register, 1 wait state is added.
 - For a 16-bit write to the CRC data register, followed by a read of the CRC checksum register, 2 wait states are added.
 - For a 32-bit write to the CRC data register, followed by a read of the CRC checksum register, 4 wait states are added.

37.6 Memory Map and register definition

This section includes the CRC module memory map and detailed descriptions of all registers.

37.6.1 CRC register descriptions

37.6.1.1 CRC memory map

CRC_ENGINE base address: 4012_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	MODE Register (MODE)	32	RW	0100_1000h
4h	CRC Seed Register (SEED)	32	RW	0000_FFFFh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
8h	CRC Sum (SUM)	32	R	0000_FFFFh
8h	CRC Write Data (WR_DATA)	32	RW	See section

37.6.1.2 MODE Register (MODE)

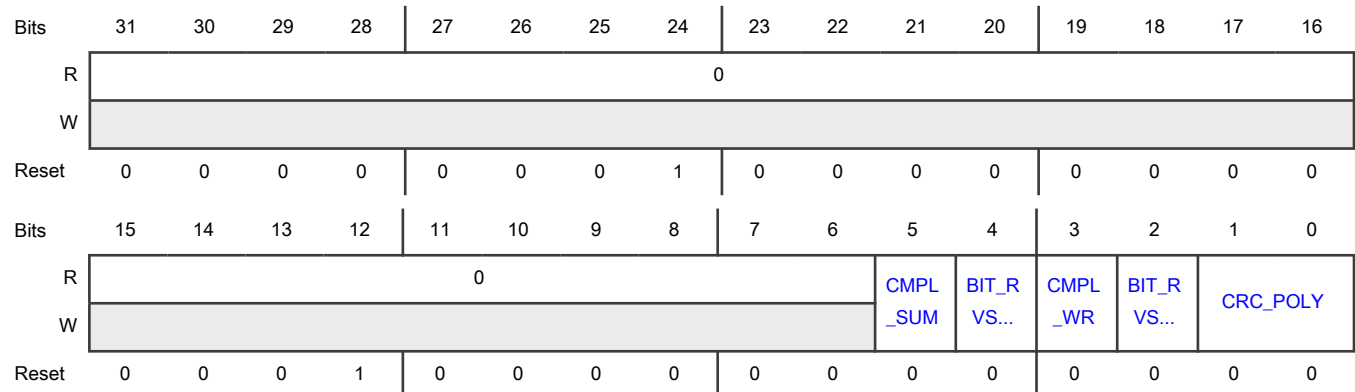
Offset

Register	Offset
MODE	0h

Function

Controls the CRC modes of operation.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 CMPL_SUM	1's Complement for CRC Sum Selects whether 1's complement is used for CRC Sum. 0b - Do not use 1's complement for CRC Sum 1b - Use 1's complement for CRC Sum
4	Bit-order Reverse for CRC Sum

Table continues on the next page...

Table continued from the previous page...

Field	Function
BIT_RVS_SUM	Selects where bit-order reverse is used for CRC Sum. 0b - Do not use bit-order reverse for CRC Sum 1b - Use bit-order reverse for CRC Sum
3 CMPL_WR	1's Complement for Write Data Selects whether 1's complement is used for WR_DATA. 0b - Do not use 1's complement for WR_DATA 1b - Use 1's complement for WR_DATA
2 BIT_RVS_WR	Bit-order Reverse for Write Data Selects where bit-order reverse is used for Write Data. 0b - Do not use bit-order reverse for WR_DATA (per byte) 1b - Use bit-order reverse for WR_DATA (per byte)
1-0 CRC_POLY	CRC Polynomial Selects which polynomial is used for the CRC algorithm. 00b - Use CRC-CCITT polynomial 01b - Use CRC-16 polynomial 10b - Use CRC-32 polynomial 11b - Reserved

37.6.1.3 CRC Seed Register (SEED)

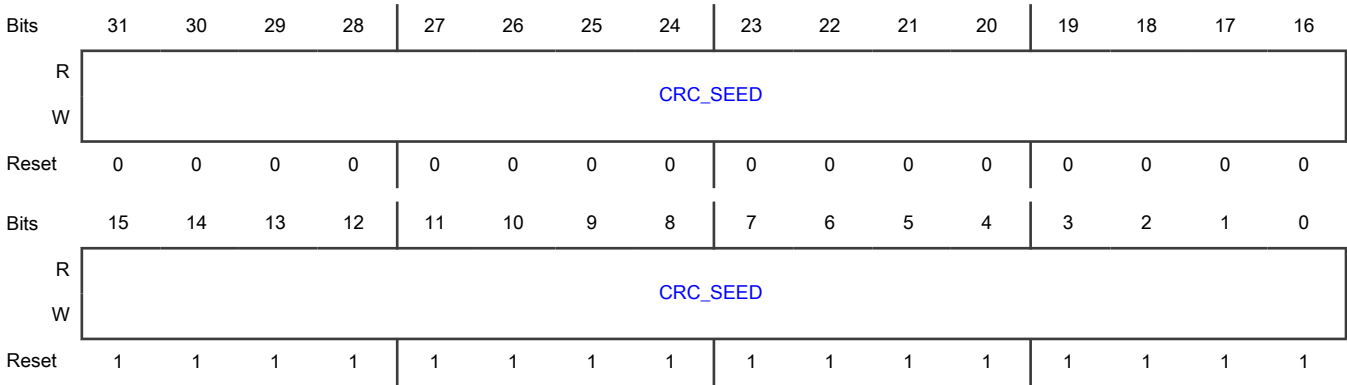
Offset

Register	Offset
SEED	4h

Function

A write access to SEED register will load the CRC seed value to the SUM register with the selected bit order and 1's complement pre-processes. A write access to SEED register will overrule the CRC calculation in progress.

Diagram



Fields

Field	Function
31-0	CRC Seed
CRC_SEED	The CRC seed value to be used in the CRC algorithms.

37.6.1.4 CRC Sum (SUM)

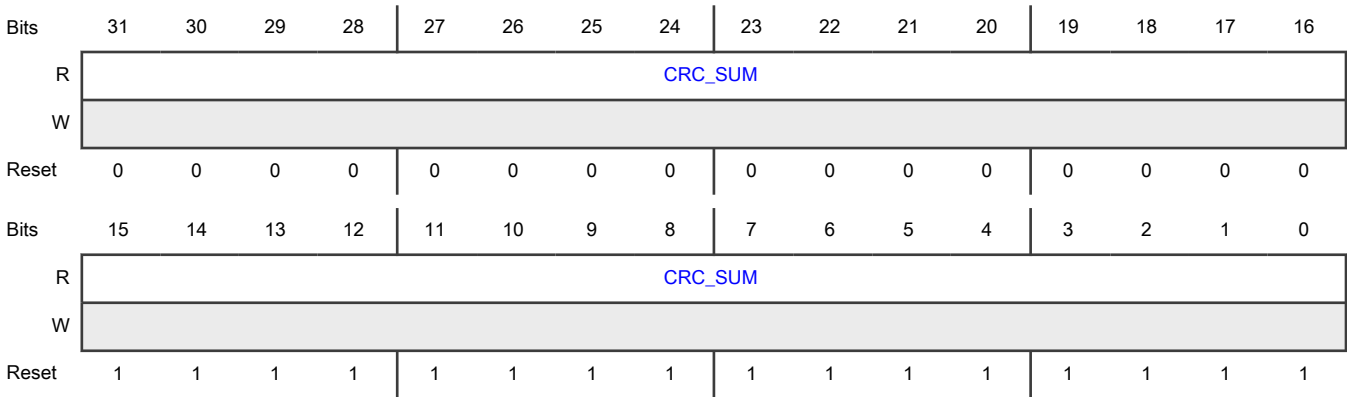
Offset

Register	Offset
SUM	8h

Function

Contains the most recent CRC sum with the selected bit order and 1’s complement post-processes.

Diagram



Fields

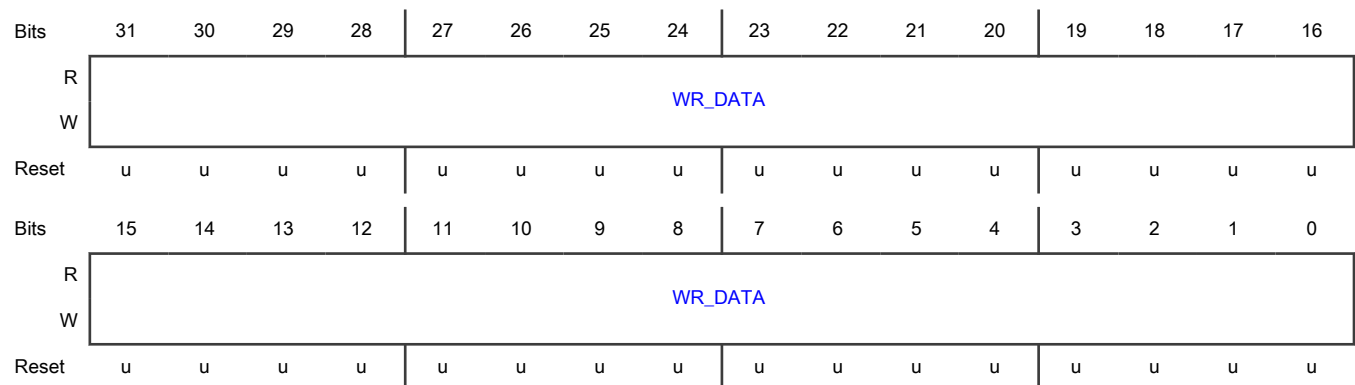
Field	Function
31-0 CRC_SUM	CRC Sum

37.6.1.5 CRC Write Data (WR_DATA)**Offset**

Register	Offset
WR_DATA	8h

Function

Data written to WR_DATA register will be used to perform the CRC calculation with the selected bit order and 1's complement pre-process. Any write size (8, 16 or 32-bit) is allowed, and back-to-back transactions are accepted.

Diagram**Fields**

Field	Function
31-0 WR_DATA	CRC Write Data

Chapter 38

On-Chip OTP Controller (OCOTP_CTRL)

38.1 Chip-specific OCOTP_CTRL information

Table 412. Reference links to related information

Topic	Related module	Reference
Full description	OCOTP_CTRL	OCOTOP_CTRL
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

38.1.1 Module instances

This device has one instance of the OCOTP_CTRL module, OCOTP0.

38.2 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller along with details about the block functionality and implementation.

The OCOTP_CTRL module provides an interface for reading, programming and/or overriding identification and control information stored in on-chip fuse elements. The module supports electrically-programmable poly fuses (e-Fuses). For security purposes, the fuses protect the confidentiality or integrity of critical security data against both software attacks and board-level hardware attacks.

NOTE

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

38.2.1 OTP Controller Sub System Block Diagram

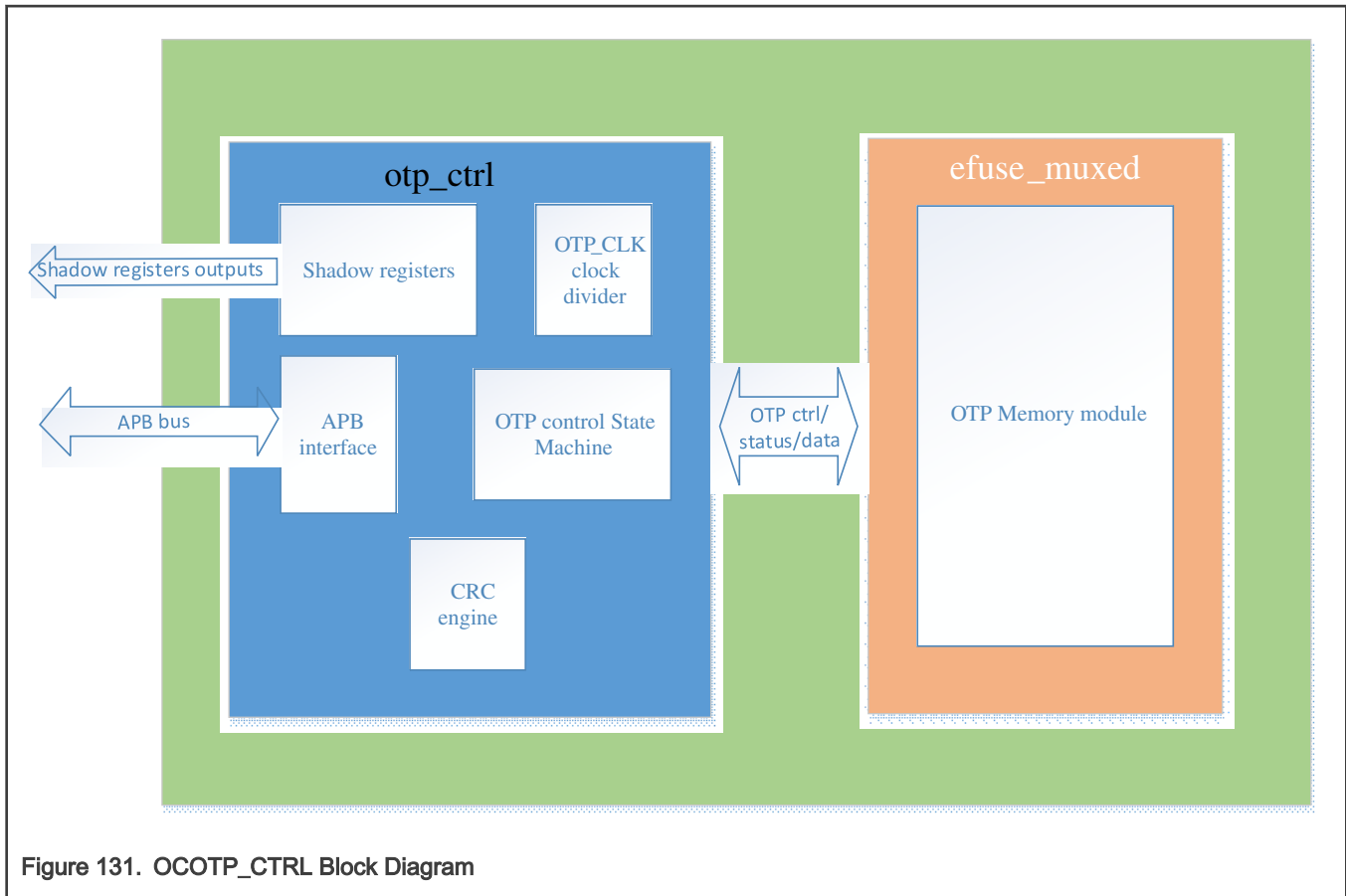


Figure 131. OCOTP_CTRL Block Diagram

38.2.2 Features

The OCOTP provides the following features:

- 32-bit word restricted program and read to 16 kbit of eFuse OTP.
- Shadow registers are automatically loaded with OTP fuse value on power-on-reset (POR) and can be read/written/reloaded by software after reset. Shadow registers can be used to temporarily change boot options, or for development, without permanently blowing an OTP fuse.
- The outputs of all shadow registers are visible to other modules internally
- Some shadow register can be configured with read/write/reload protection by using other fuse bits
- The APB bus can run at full system speed (e.g. 200 Mhz) while the OTP memory access clock is limited to the maximum 120Mhz with the internal clock divider
- Support CRC calculation for fuse block
- Maximum functional clock frequency is 120Mhz
- Mean programming time is less than 10us per bit
- Programming lock word-by-word
- Unprogrammed fuse bit value is 0
- Word size: 32 useful data bits for ECC mode and 16 useful data bits for redundancy mode

38.3 Functional Description

38.3.1 APB Interface

APB Interface is implemented to access the registers in OTP_CTRL. No wait states for general registers and shadow registers access.

38.3.1.1 Error Conditions

For other operations, the following conditions will set the OTP_STATUS[ERROR] register bit:

- Read non-existent or read-locked shadow registers
- Read read-locked fuse words
- Program write-locked fuse words
- Write to write-locked shadow registers
- Write to registers while reloading is in progress

38.3.2 Reset

After Power-On-Reset is de-asserted, OTP controller starts reading the fuse words and loading the value to the corresponding shadow registers. OTP_STATUS[BUSY] is set while the loading is in progress. Once all shadow registers are loaded, OTP_STATUS[FUSE_LATCHED] is set and the internal signal fuse_latched is asserted.

38.3.3 Operations

OTP controller supports the following operations:

- Automatically load shadow registers with the corresponding fuse words after power-on-reset.
- Read fuse word
- Program fuse word
- Reload shadow registers
- Read and write shadow registers
- Perform CRC on fuse words in specified address range

38.3.3.1 Fuse Read Operation

Here is the sequence to read a fuse word:

1. Wait for previous operation to end by checking OTP_STATUS[BUSY] bit.
2. Handle error conditions from previous operation and clear all error bits including OTP_STATUS[ERROR] bit.
3. Set fuse word address OTP_CTRL[ADDR].
4. Write 1 to OTP_READ_CTRL[READ] to start read operation.
5. Wait until read operation is done by checking OTP_STATUS[BUSY] bit.
6. Read fuse data from OTP_READ_DATA register. Check status bits including OTP_STATUS[ERROR] and OTP_STATUS[DED] and take actions accordingly.

38.3.3.2 Fuse Programming Operation

Here is the sequence to program a fuse word:

1. Wait for previous operation to end by checking OTP_STATUS[BUSY] bit.
2. Handle error conditions from previous operation and clear all error bits including OTP_STATUS[ERROR] bit.

3. Write 3E77h to OTP_CTRL[WR_UNLOCK] and set fuse word address OTP_CTRL[ADDR].
4. Write data to program data register OTP_WRITE_DATA to start programming fuse
5. Wait until programming operation is done by checking OTP_STATUS[BUSY] bit.
6. Check OTP_STATUS[ERROR], OTP_STATUS[PROGFAIL] and OTP_STATUS[LOCKED] and take actions as needed.

38.3.3.3 Shadow Register Reload Operation

Here is the sequence to reload shadow registers:

1. Wait for previous operation to end by checking OTP_STATUS[BUSY] bit.
2. Handle error conditions from previous operation and clear all error bits including OTP_STATUS[ERROR] bit.
3. Set OTP_CTRL[RELOAD_SHADOWS] to start reloading shadow registers
4. Wait until read operation is done by checking OTP_STATUS[BUSY] bit.
5. Check status bits including OTP_STATUS[ERROR] and OTP_STATUS[DED] and take actions as needed.

38.3.3.4 Read and write shadow registers

Software can perform APB read operation to read the shadow register if it's not read-locked. It returns 0 if the shadow register is read-locked.

Software can perform APB write operation to write the shadow register if it's not write-locked. It has no effect if the shadow register is write-locked.

38.3.3.5 CRC Operation

Here is the sequence to perform CRC operation:

1. Wait for previous operation to end by checking OTP_STATUS[BUSY] bit.
2. Handle error conditions from previous operation and clear all error bits including OTP_STATUS[ERROR] bit.
3. Set OTP_CRC_ADDR[CRC_START_ADDR], OTP_CRC_ADDR[CRC_END_ADDR] and OTP_CRC_ADDR[CRC_REF_ADDR]
4. Set OTP_CTRL[CRC_TEST] to start CRC calculation
5. Check status bits including OTP_STATUS[CRC_FAIL], OTP_STATUS[ERROR] and OTP_STATUS[DED] and take actions as needed.
6. Optionally read OTP_CRC_VALUE register for the CRC result if it's not read-locked. If it's read-locked, it reads back BADA_BADAh no matter the CRC passes or not (indicating by OTP_STATUS[CRC_FAIL]). If it's not read-locked, it reads back 0 when CRC passes and non-zero value when fails.

38.4 Initialization

1. Wait for previous operation to end by checking OTP_STATUS[BUSY] bit.
2. Handle error conditions from previous operation and clear all error bits including OTP_STATUS[ERROR] bit.

38.5 Memory map and register definition

38.5.1 OCOTP Controller register descriptions

38.5.1.1 OTP memory map

OCOTP0 base address: 4013_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 7BCh	One Time Programmable Controller shadow (OTP_SHADOW0 - OTP_SHADOW495)	32	RW	See section
800h	Control/Address (OTP_CTRL)	32	RW	See section
804h	Power-down (OTP_PDN)	32	RW	See section
808h	OTP programming data (OTP_WRITE_DATA)	32	RW	See section
80Ch	OTP read start control (OTP_READ_CTRL)	32	RW	See section
810h	OTP read data (OTP_READ_DATA)	32	R	0000_0000h
814h	OTP clock divider (OTP_CLK_DIV)	32	RW	See section
81Ch	CRC address range (OTP_CRC_ADDR)	32	RW	See section
820h	CRC result (OTP_CRC_VALUE)	32	R	0000_0000h
824h	OTP Status (OTP_STATUS)	32	RW	See section
82Ch	VERSION ID (OTP_VERSION)	32	R	0800_0000h

38.5.1.2 One Time Programmable Controller shadow (OTP_SHADOW0 - OTP_SHADOW495)

Offset

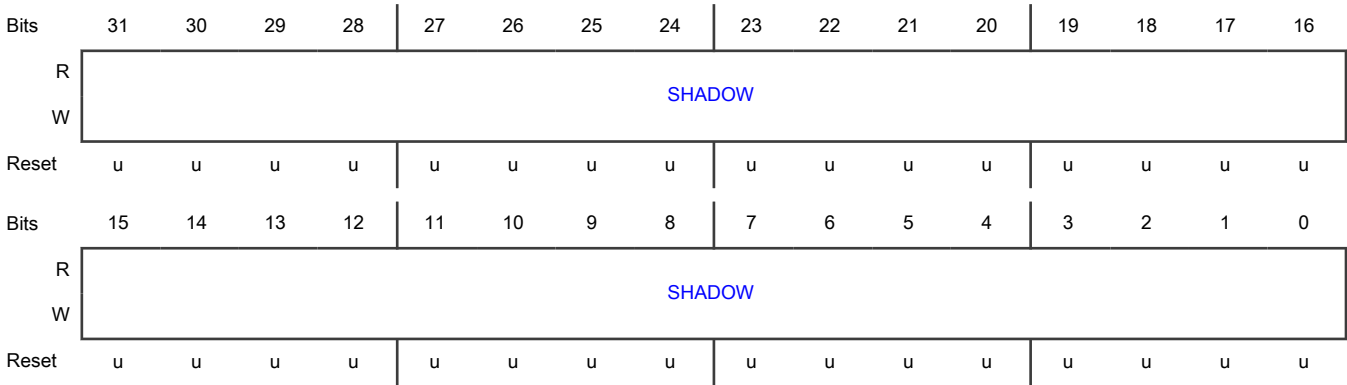
For a = 0 to 495:

Register	Offset
OTP_SHADOWa	0h + (a × 4h)

Function

Software can perform APB read operation to read the shadow register if it's not read-locked. It returns 0 if the shadow register is read-locked. Software can perform APB write operation to write the shadow register if it's not write-locked. It has no effect if the shadow register is write-locked.

Diagram



Fields

Field	Function
31-0 SHADOW	OTP shadow It returns 0 if the shadow register is read-locked. It has no effect if the shadow register is write-locked.

38.5.1.3 Control/Address (OTP_CTRL)

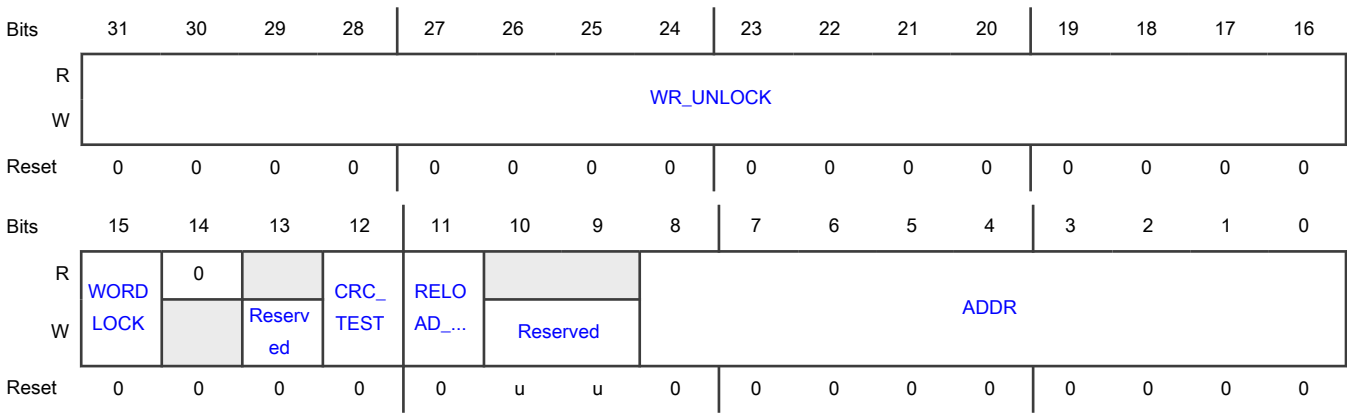
Offset

Register	Offset
OTP_CTRL	800h

Function

This register enables reloading shadow registers, CRC calculation, writ locking and unlocking fuse word.

Diagram



Fields

Field	Function
31-16 WR_UNLOCK	Write unlock Write 0x3E77 to enable OTP write accesses. NOTE: The write operation must be unlocked for each word by writing 0x3E77 to WR_UNLOCK field. Then writing to OTP_WRITE_DATA register will automatically start the programming procedure.
15 WORDLOCK	Wordlock Set to write-lock the fuse word when it's being programming. When programming with ECC mode, it recommends to set this bit.
14 —	Reserved
13 —	Reserved
12 CRC_TEST	Set to start CRC calculation. This operation will automatically set OTP_STATUS.BUSY. Once CRC is calculation done, OTP_STATUS.BUSY and CRC_TEST are automatically cleared by the controller.
11 RELOAD_SHADOWS	Reload Shadow registers Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set OTP_STATUS.BUSY. Once the shadow registers have been re-loaded, OTP_STATUS.BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
10-9 —	Reserved
8-0 ADDR	OTP word address These bits include word address for read/programming.

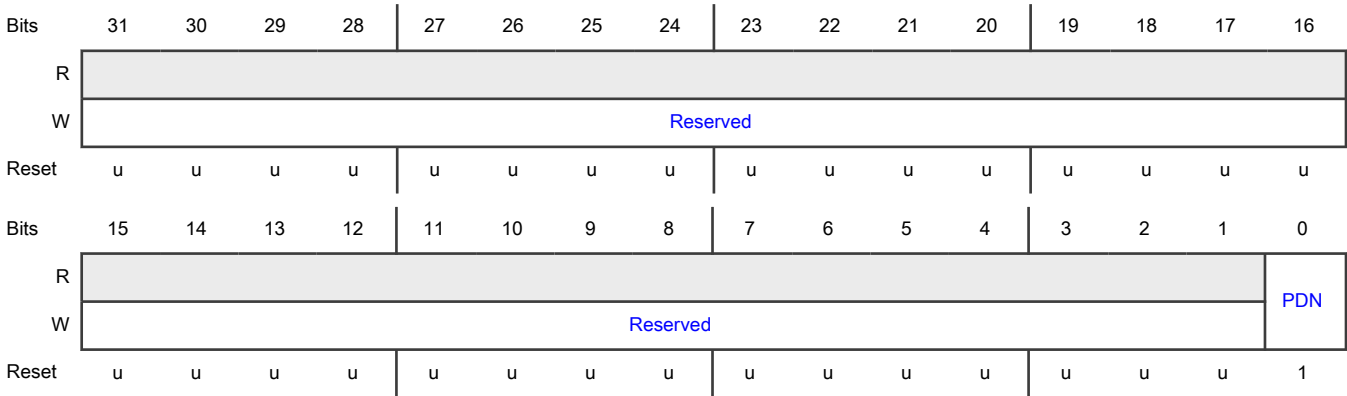
38.5.1.4 Power-down (OTP_PDN)**Offset**

Register	Offset
OTP_PDN	804h

Function

This register indicates the PDN value of OTP memory.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 PDN	Power-down This bit indicates the PDN value of OTP memory. Writing 1 to the bit to clear PDN. Writing 0 has no effect. Note: Software need to write 1 to this bit to shut off power of OTP memory after system power up. At the beginning of every fuse operation, the controller will automatically turn-on power to the OPT memory. After every fuse operation, software also need to write 1 to this bit to shut off power to the OTP memory to reduce power consumption.

38.5.1.5 OTP programming data (OTP_WRITE_DATA)

Offset

Register	Offset
OTP_WRITE_DATA	808h

Function

Fuse word programming data.

Diagram



Fields

Field	Function
31-0 WRITE_DATA	Write data After the write operation is unlocked in OTP_CTRL register, writing data to this register automatically start the programming procedure.

38.5.1.6 OTP read start control (OTP_READ_CTRL)

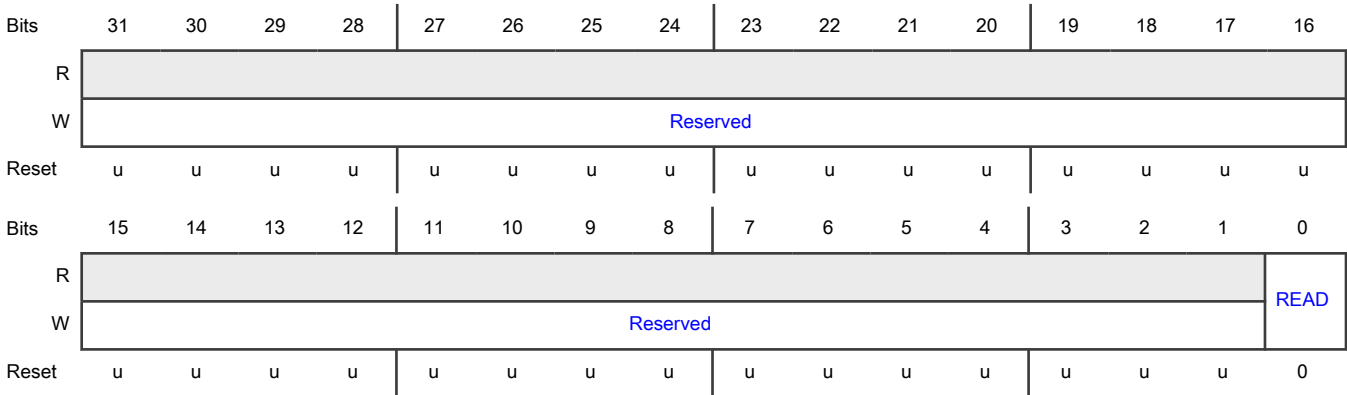
Offset

Register	Offset
OTP_READ_CTRL	80Ch

Function

This register starts fuse word read operation.

Diagram



Fields

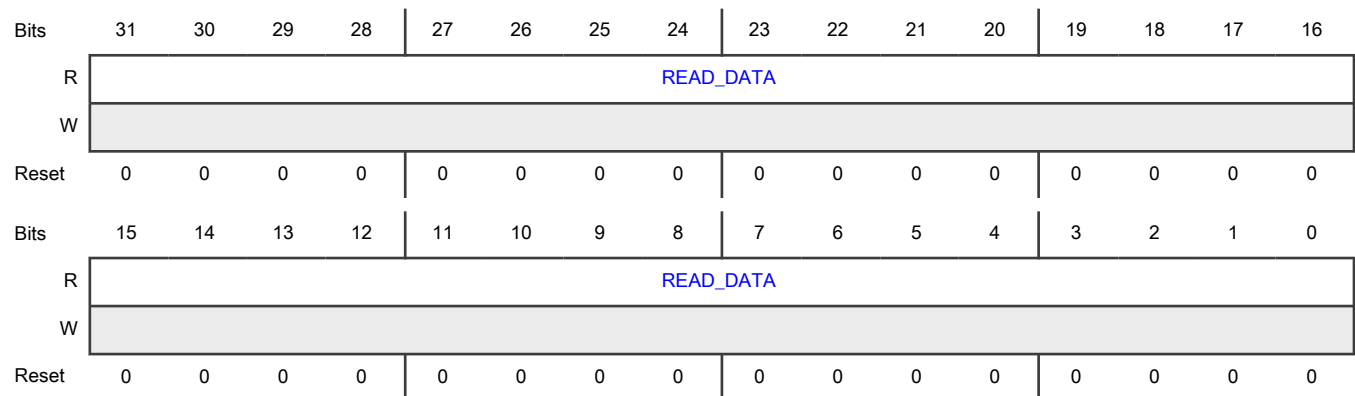
Field	Function
31-1 —	Reserved
0 READ	Read operation Write 1 to start read operation. This operation will automatically set OTP_STATUS[BUSY]. Once the read operation is complete, OTP_STATUS[BUSY] and READ bits are automatically cleared by the controller.

38.5.1.7 OTP read data (OTP_READ_DATA)**Offset**

Register	Offset
OTP_READ_DATA	810h

Function

This register reads fuse word data from read operation.

Diagram**Fields**

Field	Function
31-0 READ_DATA	Fuse word read data from read operation

38.5.1.8 OTP clock divider (OTP_CLK_DIV)

Offset

Register	Offset
OTP_CLK_DIV	814h

Function

This register divides the OTP clock, halts and resets the OTP counter.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQF LAG				Reserved											
W		HALT	RESET													
Reset	0	0	0	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												DIV			
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0

Fields

Field	Function
31 REQFLAG	Divider status flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts the divider counter The intent is to allow the divider's clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets the divider counter. Can be used to make sure a new divider value is used right away rather than completing the previous count.
28-4 —	Reserved for future use
3-0 DIV	Clock divider value by -1 encoding It is used to generate the clock to OTP memory (otp_clk) with apb_clk. The maximum otp_clk frequency is 120Mhz. 0000b - Divide by 1 0001b - Divide by 2

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0010b - Divide by 3
	0011b - Divide by 4
	0100b - Divide by 5
	0101b - Divide by 6
	0110b - Divide by 7
	0111b - Divide by 8
	1000b - Divide by 9
	1001b - Divide by 10
	1010b - Divide by 11
	1011b - Divide by 12
	1100b - Divide by 13
	1101b - Divide by 14
	1110b - Divide by 15
	1111b - Divide by 16

38.5.1.9 CRC address range (OTP_CRC_ADDR)

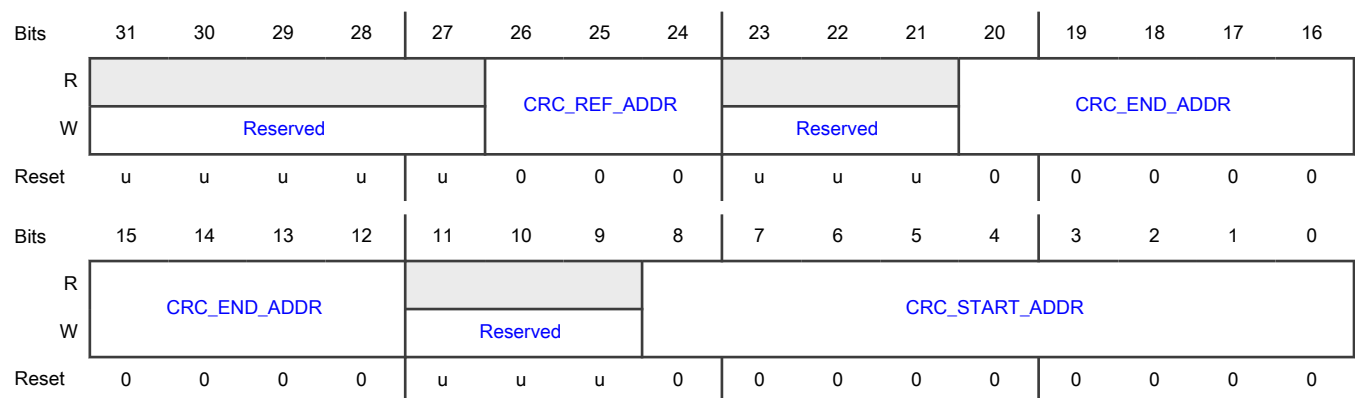
Offset

Register	Offset
OTP_CRC_ADDR	81Ch

Function

This register specifies start, end and reference for CRC address.

Diagram



Fields

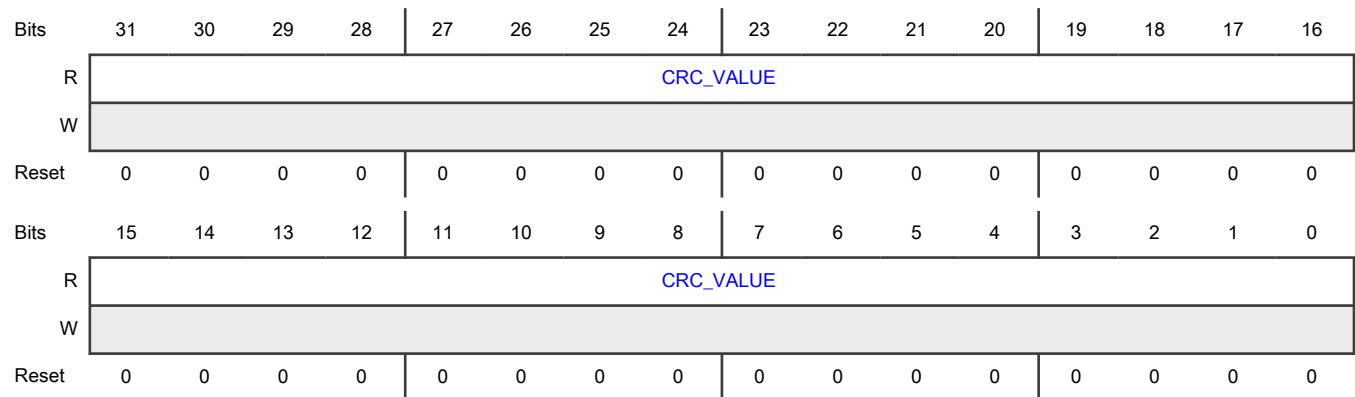
Field	Function
31-27 —	Reserved
26-24 CRC_REF_AD DR	CRC reference address Specify which of the 8 CRC reference value to use for CRC calculation. When the CRC result for the fuse data from CRC_START_ADDR to CRC_END_ADDR and this CRC reference value is 0, the CRC check passes.
23-21 —	Reserved
20-12 CRC_END_AD DR	CRC ending fuse word address
11-9 —	Reserved
8-0 CRC_START_A DDR	CRC starting fuse word address

38.5.1.10 CRC result (OTP_CRC_VALUE)**Offset**

Register	Offset
OTP_CRC_VALUE	820h

Function

This register provides result of CRC.

Diagram**Fields**

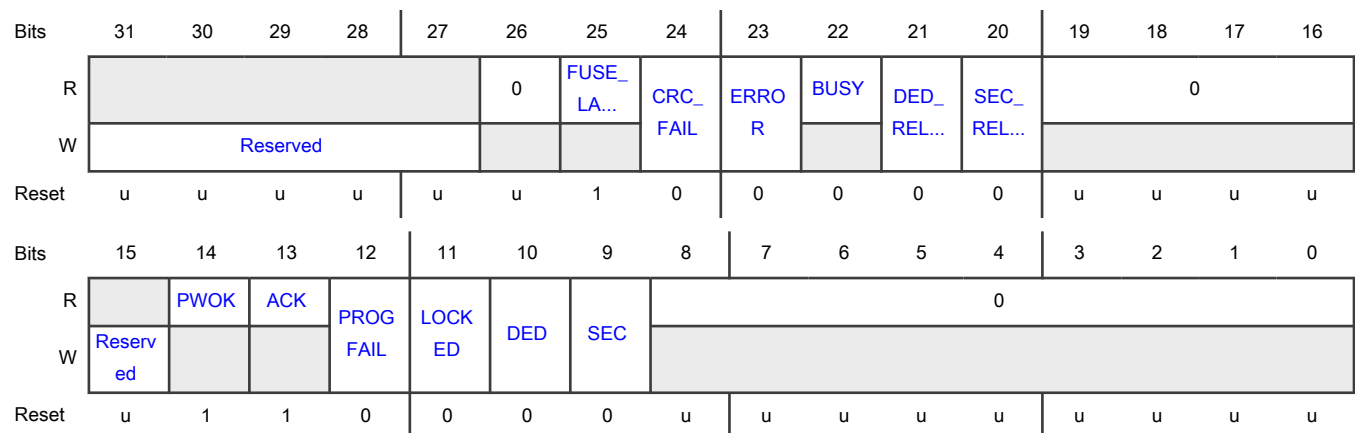
Field	Function
31-0 CRC_VALUE	CRC result value. When it is locked, reading from it returns value 0xBADA_BADA.

38.5.1.11 OTP Status (OTP_STATUS)**Offset**

Register	Offset
OTP_STATUS	824h

Function

This register provides information on OTP status.

Diagram

Fields

Field	Function
31-27 —	Reserved
26 —	Reserved
25 FUSE_LATCHED	Indicate all shadows registers have been loaded with their corresponding fuse words when set by the controller after reset.
24 CRC_FAIL	CRC failed when set by hardware for CRC operation. Write 1 to clear.
23 ERROR	Error Set by the controller when a read/write access to a locked region (OTP or shadow register) is requested. Writing 1 to clear it before any further access can be performed. This bit can only be set by the controller.
22 BUSY	OTP controller status When active, no new write or read access to OTP (including RELOAD_SHADOWS) can be performed. Cleared by the controller when the access completes. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller and cleared after all the shadow registers are successfully loaded.
21 DED_RELOAD	OTP Double Error Detect status of ECC during reload process. Write 1 to clear.
20 SEC_RELOAD	OTP Single Error Corrected status of ECC during reload process. Write 1 to clear.
19-16 —	Reserved
15 —	Reserved
14 PWOK	OTP Power OK status. Indicate that power VDD are in the operating range.
13 ACK	OTP ACK value Operation acknowledgment from OTP.
12 PROGFAIL	OTP PROGFAIL status. Write 1 to clear. Informs if programming failed.
11	OTP LOCKED status during read/write operation. Write 1 to clear.

Table continues on the next page...

Table continued from the previous page...

Field	Function
LOCKED	
10 DED	OTP Double Error Detection status of ECC during read operation. Write 1 to clear.
9 SEC	OTP Single Error Corrected status of ECC during read operation. Write 1 to clear.
8-0 —	Reserved

38.5.1.12 VERSION ID (OTP_VERSION)

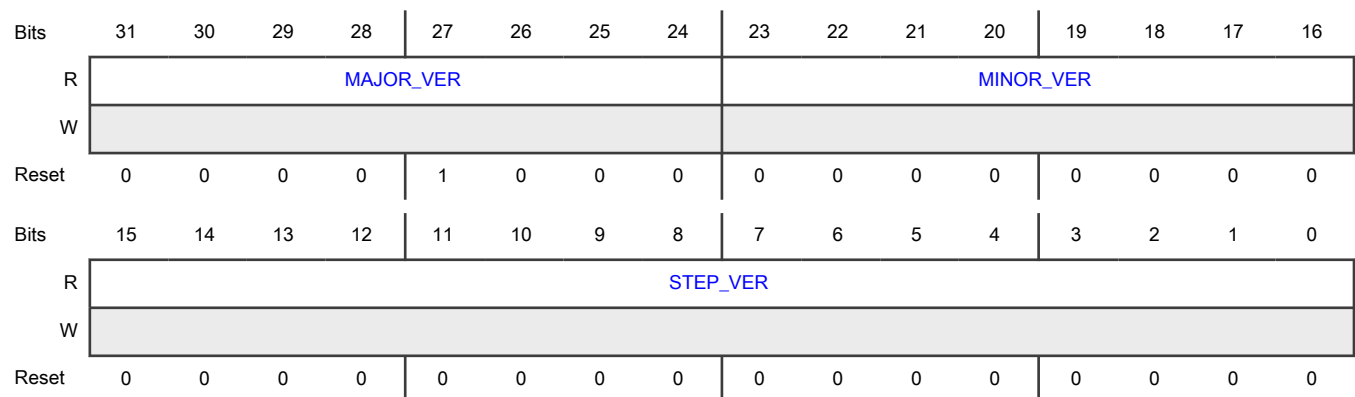
Offset

Register	Offset
OTP_VERSION	82Ch

Function

This register includes information for the OTP controller major, minor and step version.

Diagram



Fields

Field	Function
31-24 MAJOR_VER	Major version OTP controller major version
23-16	Minor version

Table continues on the next page...

Table continued from the previous page...

Field	Function
MINOR_VER	OTP controller minor version
15-0 STEP_VER	Step version OTP controller step version

Chapter 39

FlexSPI

39.1 Chip-specific FLEXSPI information

Table 413. Reference links to related information

Topic	Related module	Reference
Full description	FLEXSPI	FLEXSPI
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

There are two FlexSPI Flash Interface modules, supporting Octal and Quad SPI memory devices. The first instance is primarily intended for code execution from off-chip SPI flash memory. The second instance is primarily intended to access graphics data. The second instance will be accessible by the DSP processor as well as the M33. For data rates supported by the interfaces, please refer to the datasheet.

The FlexSPI interfaces supports HyperFlash, HyperRAM and Xccela memory types, among others. The first FlexSPI interface will support execute-in-place and on-the-fly decryption using the latest OTFAD module. (This module is accessible only by the M33 CPU, the HASH-CRYPT block and both DMA controllers)

Both the FlexSPI interfaces include a 32 KB cache with an LPMEM AHB-cache controller.

NOTE

In DLL Control Register 0(DLLACR - DLLBCR), bit SLVDLYTARGET, if serial root clock is ≥ 100 MHz, DLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0x0 is recommended.

	FlexSPI0 (OTFAD_EN=1)	FlexSPI1 (OTFAD_EN=0)
Address range is mapped for AHB read access to IP RX FIFO: 0x(ARDF_BASE.toHEX()) - 0x(ARDF_TOP.toHEX())	0x1000_0000 – 0x1000_03FF	0x1000_0000 – 0x1000_03FF
Address range is mapped for AHB write access to IP TX FIFO: 0x(ATDF_BASE.toHEX()) - 0x(ATDF_TOP.toHEX())	0x1100_0000 – 0x1100_03FF	0x1100_0000 – 0x1100_03FF
AHB Memory Map for Serial Flash memory access		
Total AHB RX Buffer size: (2**AHB_RX_BUF_ADDR_WIDTH * 8) KBytes	(2**7) * 8 buffers = 1 KByte	(2**8) * 8 buffers = 2 kBytes
AHB TX Buffer size: (2**AHB_TX_BUF_ADDR_WIDTH * 8) Bytes	(2**3) * 8 = 64 Bytes	(2**3) * 8 = 64 Bytes

Table continues on the next page...

Table continued from the previous page...

IP Bus memory space access - 0x(ARDF_BASE.toHEX()) - 0x(ARDF_TOP.toHEX()) (by AHB Bus)	0x1000_0000 – 0x1000_03FF	0x1000_0000 – 0x1000_03FF
AHB bus memory space access- (ASFM_BASE.toHEX())~0x(ASFM_TOP .toHEX())	0x0800_0000 – 0x0FFF_FFFF	0x2800_0000 – 0x2FFF_FFFF

39.1.1 Module instances

This device has two instances of the FLEXSPI module, FLEXSPI0 and FLEXSPI1. FlexSPI0 has an AHB RX Buffer size of 1KB, and FlexSPI1 buffer size is 2KB. FlexSPI0 and FlexSPI1 supports Port A only; Port B is not included.

NOTE

Only FLEXSPI0 supports OTFAD

39.1.2 Master ID Assignments

Table 414. FlexSPI0 Master ID Assignments

FlexSPI0		
Master Port	Master Name	Master ID
M0	M33 Code	0000
M3	DSP	0011
M4	DMA0	0100
M5	DMA1	0101
M7	Smart DMA Data	0111
M10	HASH_CRYPT	1010
M11	AXI:AHB Bridge	1011

Table 415. FlexSPI1 Master ID Assignments

FlexSPI1		
Master Port	Master Name	Master ID
M1	M33 System	0001
M2	Power Quad	0010
M3	DSP	0011
M4	DMA0	0100
M5	DMA1	0101
M7	Smart DMA Data	0111
M10	HASH_CRYPT	1010
M11	AXI:AHB Bridge	1011

39.1.3 Initialization

In [Initialization](#), steps 2 and 3 are as below-

- Power the FlexSPI SRAM using PDRUNCFG registers.
- Release FlexSPI from reset using PRSTCRL registers.

39.2 Overview

Flexible Serial Peripheral Interface (FlexSPI) host controller supports one SPI channel and up to 2 external devices. Each channel supports Single/Dual/Quad/Octal mode data transfer (1/2/4/8 bidirectional data lines).

NOTE

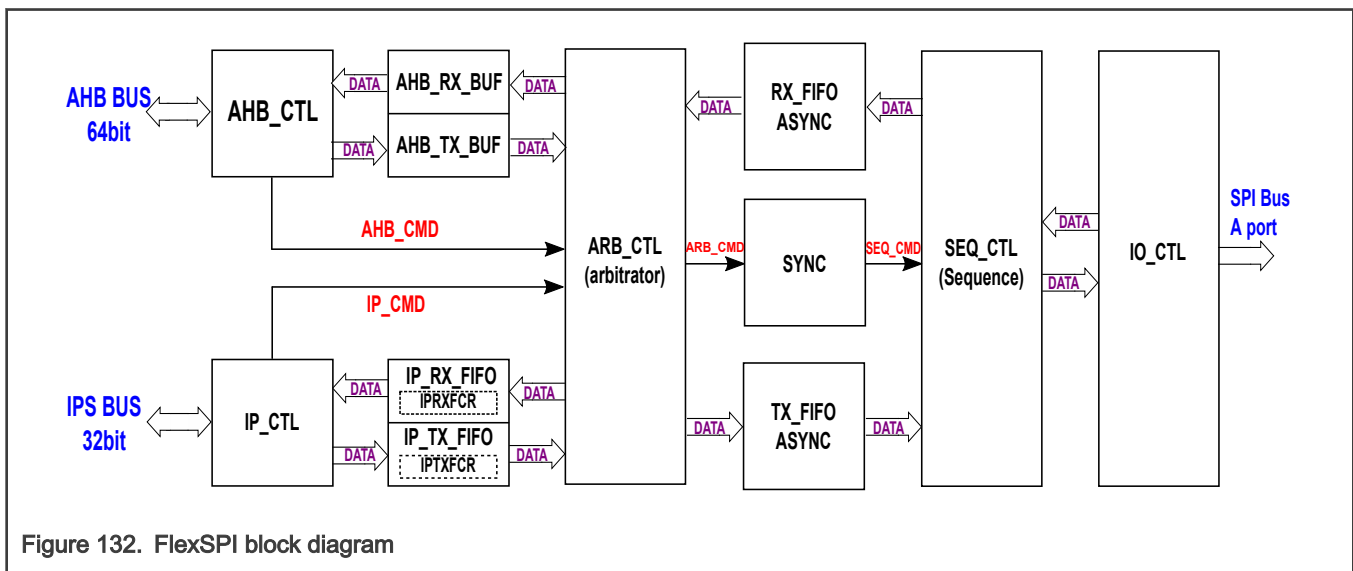
FlexSPI configuration is dependent on the chip configuration. See the system-level section for boot and pinmux for chip specific information regarding the modes and number of devices supported.

NOTE

The FlexSPI controller supports communication with both serial flash and serial RAM devices. While many of the descriptions, registers, and field specifically reference flash almost all information can also be applied to serial RAM. Flash is used as an example in the tables and figures in this chapter.

39.2.1 Block diagram

The block diagram of FlexSPI is indicated in the following figure:



39.2.2 Features

FlexSPI supports the following features:

- Flexible sequence engine (LUT table) to support various vendor devices
 - Serial NOR Flash or other device with similar SPI protocol as Serial NOR Flash
 - Serial NAND Flash
 - HyperBus device (HyperFlash/HyperRAM)
 - FPGA device
- Flash access mode

- Single/Dual/Quad/Octal mode
- SDR/DDR mode
- Individual/Parallel mode
- Support for sampling clock mode:
 - Internal dummy read strobe loopbacked internally
 - Internal dummy read strobe loopbacked from pad
 - Flash provided read strobe
- Automatic data learning to select the correct sample clock phase
- Memory mapped read/write access by AHB Bus
 - AHB RX Buffer implemented to reduce read latency. See the chip-specific section for AHB buffer size.
 - 16 AHB masters supported with programmable priority for read access from each
 - 8 flexible and configurable buffers in AHB RX Buffer
 - AHB TX Buffer implemented to buffer all write data from one AHB burst. See the chip-specific section for AHB buffer size.

NOTE

All AHB masters share this AHB TX Buffer. No AHB master number limitation for write access.

- Software triggered Flash read/write access by IP Bus
 - IP RX FIFO implemented to buffer all read data from external devices. FIFO size: 1024 Bytes
 - IP TX FIFO implemented to buffer all write data to external devices. FIFO size: 1024 Bytes
 - DMA support to fill IP TX FIFO
 - DMA support to read IP RX FIFO
 - SCLK would stop when IP RX FIFO is full during reading flash data
 - SCLK would stop when IP TX FIFO is empty during writing flash data

39.3 Glossary for FlexSPI module

Table 416. Glossary

Term	Definition
AHB Command	AHB Command is one or several Serial Flash Memory access command sequences triggered by AHB read/write access to AHB address ranged mapped for Serial Flash Memory.
ASFM_BASE	Base address of AHB address space mapped to Serial Flash Memory.
Set	Write 1 to register bit to establish logic level one on the bit.
Clear	Write 0 to register bit to establish logic level zero on the bit.
Command Sequence	A command sequence is 4*32 bits sequence code consisting of up to 8 instructions. Command sequence should be programmed in LUT according to Flash command. When a command sequence executed, the chip selection signal becomes asserted.

Table continues on the next page...

Table 416. Glossary (continued)

Term	Definition
	When a command sequence execution finished, the chip selection signal becomes negated.
DDR	Dual data transfer rate for flash access mode Flash receive data on both SCLK rise and fall edge and transmit data on both SCLK rise and fall edge.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Instruction Code	16 bits code defining the type of command to be executed on FlexSPI interface.
IP Command	IP Command is one or several Serial Flash Memory access command sequences triggered by set register bit IPCMD.TRG.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
SDR	Single data transfer rate for flash access mode Flash receive data on SCLK rise edge and transmit data on SCLK fall edge.
Set	To set a bit or bits means to establish logic level one on the bit or bits.
SFM	Serial Flash Memory
Individual Mode	Access to a single, individual serial flash device at a time.
Parallel Mode	Read/Program Access to two serial flash devices in parallel (Port A and Port B). FlexSPI will split flash program data before transmitting to Flash or merge flash read data before putting into IP RX FIFO or AHB RX Buffer automatically.
Memory Command	Serial access Command for reading/programming/configuring. A memory command may consist one or several command sequences, for each command sequence the Chip selection signal will be asserted and deasserted once.
LUT	Look-up table to preserve command sequence. LUT is programmed by software with command sequences which is used to issue memory commands.

39.4 External signals

This section provides the external signal information of the FlexSPI module.

Table 417. External Signal List

Signal Name	Function	Direction	Description
A_SS0_B	Peripheral Chip Select Flash A1	O	This signal is the chip select for the serial flash device A1.
A_SS1_B	Peripheral Chip Select Flash A2	O	This signal is the chip select for the serial flash device A2.
A_SCLK	Serial Clock Flash A	O	This signal is the serial clock output to the serial flash device A. Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence.
A_DATA _n	Serial I/O Flash A	I/O	These signals are the data I/O lines to/from the serial flash device A.
A_DQS	Data Strobe Signal Flash A	I/O	Data strobe signal for port A. There are three functions for this signal: • Driven with read strobe by external device: Some flash devices provide the read strobe signal together with read data. In this case, this pad may need a pull down resistor if external device drives this pad only when reading flash data. • Driven with latency information by external device: Certain devices use this pin to indicate the dummy cycles needed (before program/read data transfer) such as HyperRAM or HyperFlash. • Loopback dummy read strobe: The FlexSPI controller provides internal dummy read strobe for flash read data. Higher read frequency can be achieved by looping back this dummy read strobe from pad. This pin can be floated or put some cap loads on board level to compensate DATA or SCLK pins load.

39.5 Functional description

The following sections describe functional details of the FlexSPI module.

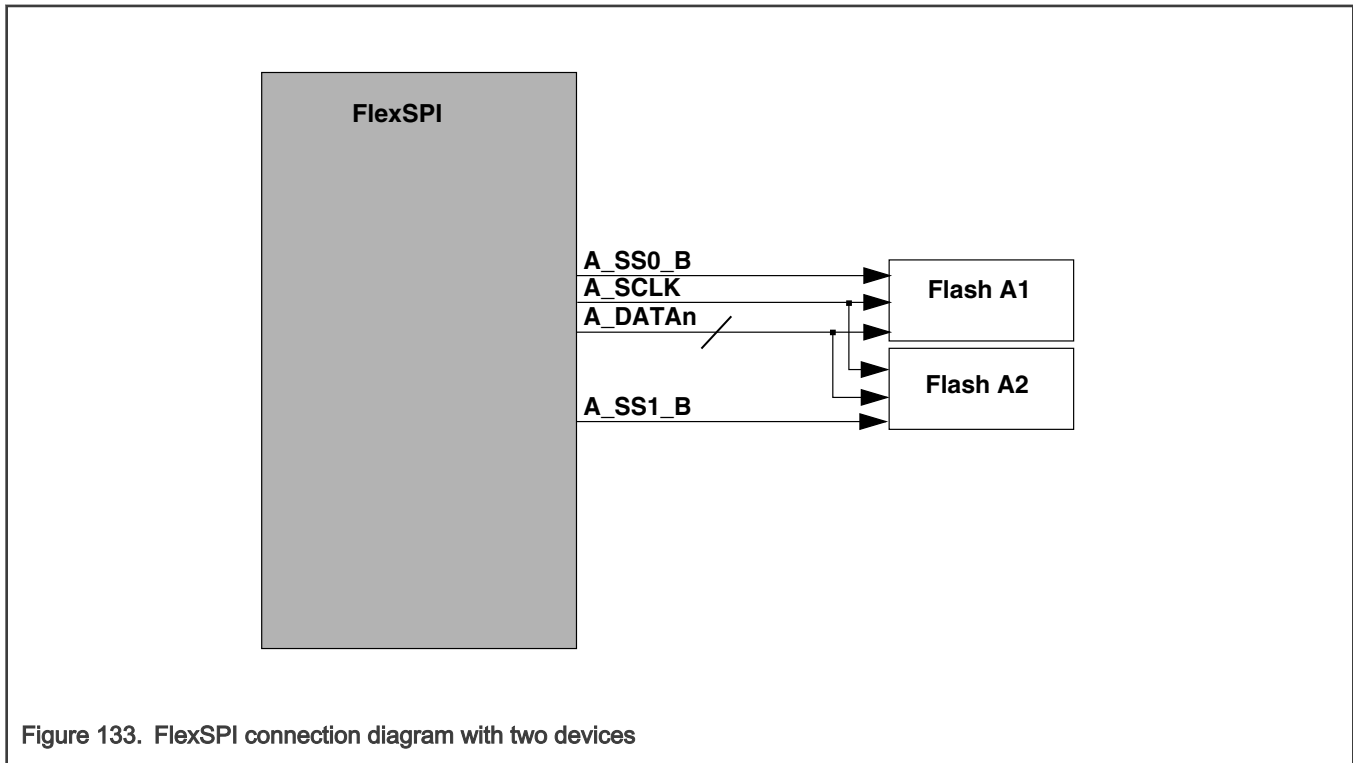
39.5.1 Flash connection

There is one FlexSPI interface port (port A) that supports up to 2 flash devices by providing two chip select outputs.

NOTE

FlexSPI configuration is dependent on the chip configuration. See the chip-specific FlexSPI information regarding the number of devices supported.

The connection diagram with two devices is shown as follows:

**NOTE**

- Flash A1 and Flash A2 can be two Flash chip packages or two Flash die in the same package.
- In parallel mode, Flash A1 and Flash A2 cannot be accessed at the same time.
- In individual mode, Flash A1, Flash A2 cannot be accessed at the same time. But these two devices can be accessed separately.

39.5.2 Flash access mode

This section describes the flash access mode.

39.5.2.1 SPI clock mode

FlexSPI only supports SPI clock mode 0: Clock polarity (CPOL)=0 and Clock Phase (CPHA)=0. SCLK stays at logic low state when SPI bus is idle.

39.5.2.2 Individual mode and Parallel mode

In the individual mode, read/write data is received/transmitted on port A or port B.

In the parallel mode, read/write data is received/transmitted on port A and port B in parallel. FlexSPI merges or splits the read/program data automatically (READ/WRITE instruction). Note that only read/program data is merged/split (READ/WRITE instruction). For other instructions (such as Command/Address/Mode/Data size), same command code/address/mode bits/data size information are transmitted to the devices connected to port A and port B. For more details, see [Instruction execution on SPI interface](#).

Individual and Parallel mode is determined statically by the register field IPCR1[IPAREN] (for IP command) or AHBCR[APAREN] (for AHB command).

NOTE

FlexSPI does not support 16-bit read/write for the SPI interface.

39.5.2.3 SDR mode and DDR mode

In SDR (Single Data Transfer Rate) mode, Flash receives data on the SCLK rising edge and transmits data on the SCLK falling edge.

In DDR (Dual Data Transfer Rate) mode, Flash receives and transmits data on both SCLK rising and falling edges.

SDR and DDR modes are determined dynamically by the instruction (opcode) in the LUT sequence. There is no static configuration register field setting for SDR and DDR modes. For more details about input and output timing, see [FlexSPI input timing](#) and [FlexSPI output timing](#).

39.5.2.4 Single, Dual, Quad, and Octal mode

In Single mode, the FlexSPI transmits/receives data on 1 data pin (DATA0 for transmitting, DATA1 for receiving).

In Dual mode, the FlexSPI transmits/receives data on 2 data pins (DATA0 - DATA1 for both transmitting and receiving).

In Quad mode, the FlexSPI transmits/receives data on 4 data pins (DATA0 - DATA3 for both transmitting and receiving).

In Octal mode, the FlexSPI transmits/receives data on 8 data pins (DATA0 - DATA7 for both transmitting and receiving).

Single, Dual, Quad and Octal modes are determined by instruction (num_pads) in LUT sequence dynamically. There is no static configuration register field setting for Single, Dual, Quad and Octal modes.

39.5.3 Operation modes

This section introduces the FlexSPI modes of operation.

- Module disable mode

This mode is low power mode in the FlexSPI.

In module disable mode, the AHB clock and serial clock domains are gated off internally, but the IPS bus clock domain is not gated off. Control and status register read/write access is available, but LUTn, IPRXFSTS, IPTXFSTS registers cannot be accessed. Serial memory access is also not available.

This mode is entered by setting MCR0[MDIS], and exited by clearing MCR0[MDIS].

- Doze mode

This mode is low power mode in SOC system.

When the SOC system requires the FlexSPI to enter doze mode and MCR0[DOZEEN] is set, the FlexSPI enters the doze mode after all transactions are completed (STS0[ARBIDLE]=0x1). In doze mode, the AHB clock and serial clock domains are gated off internally, but IPS Bus clock is not gated off. Control and status register read/write access is available, but LUTn, IPRXFSTS, IPTXFSTS registers cannot be accessed. Serial memory access is also not available.

This mode is entered by the system request, and exited by deasserting this system request.

- Stop mode

This mode is low power mode in SOC system.

When the SOC system requires the FlexSPI to enter stop mode, the FlexSPI waits for all transactions to complete (STS0[ARBIDLE]=0x1) and return ACK handshake to system. After the ACK handshake is returned, the FlexSPI gates off the AHB clock and serial clock domains internally, and the system can gate off the AHB bus clock, IPS bus clock and serial clock at the system level.

This mode is entered by system request. This mode is exited by deasserting this system request, the ACK handshake message is also deasserted immediately.

- Normal mode

In normal mode, none of the clocks are gated off internally. Normal register access and serial memory access are available.

39.5.4 AHB access memory map

The FlexSPI allocates AHB memory space for each memory device starting from the FlexSPI region base address of the system memory map. The `FLSHxCR0[FLSHSZ]` field determines the amount of memory that is allocated for each memory device in Kbytes, `Fx_SIZE`. The memory is then allocated sequentially and contiguously.

NOTE

Refer to the chip-specific section for the maximum flash size supported.

When `MCR2[SAMEDEVICEEN]` is set to 0x1, $FA1\_SIZE/FA2\_SIZE/FB1\_SIZE/FB2\_SIZE = FLSHA1CR0[FLSHSZ] * 1\text{KByte}$.

When `MCR2[SAMEDEVICEEN]` is set to 0x0, $FA1\_SIZE = FLSHA1CR0[FLSHSZ] * 1\text{KByte}$; $FA2\_SIZE = FLSHA2CR0[FLSHSZ] * 1\text{KByte}$; $FB1\_SIZE = FLSHB1CR0[FLSHSZ] * 1\text{KByte}$; $FB2\_SIZE = FLSHB2CR0[FLSHSZ] * 1\text{KByte}$.

Flash B1 and Flash B2 size settings are ignored in parallel mode (`FLSHB1CR0[FLSHSZ]`, `FLSHB2CR0[FLSHSZ]`). To execute in parallel mode, Flash B1 should be the same device as Flash A1, and Flash B2 should be the same device as Flash A2.

AHB access memory map in individual and parallel mode is shown as follows:

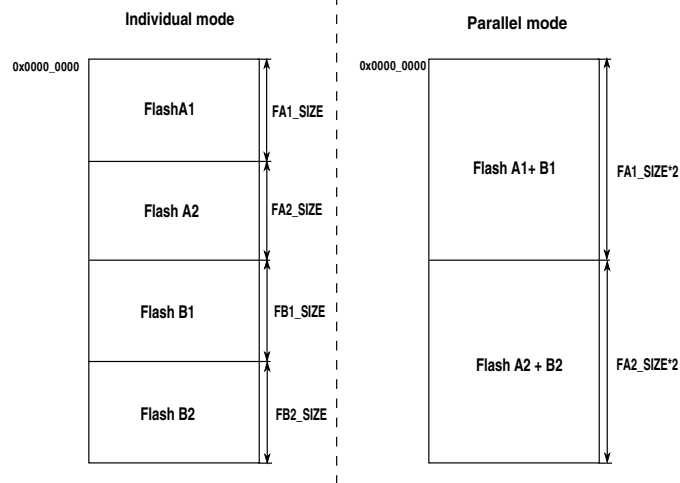


Figure 134. AHB access memory map in individual and parallel mode

AHB access memory map in individual mode:

- Flash A1 address range: 0x0000_0000 to $FA1\_SIZE$
- Flash A2 address range: $FA1\_SIZE$ to $(FA1\_SIZE + FA2\_SIZE)$
- Flash B1 address range: $(FA1\_SIZE + FA2\_SIZE)$ to $(FA1\_SIZE + FA2\_SIZE + FB1\_SIZE)$
- Flash B2 address range: $FA1\_SIZE + FA2\_SIZE + FB1\_SIZE$ to $(FA1\_SIZE + FA2\_SIZE + FB1\_SIZE + FB2\_SIZE)$

AHB access memory map in parallel mode:

- Flash A1+B1 address range: 0x0000_0000 to $FA1\_SIZE*2$

- Flash A2+B2 address range: $FA1\_SIZE*2$ to $(FA1\_SIZE*2 + FA2\_SIZE*2)$

NOTE

The address used in [Figure 134](#) and equations in this topic is the address presented to the memory. At this point, the base address for the FlexSPI in the device's system memory map has already been removed so it not used/shown here.

39.5.4.1 Dual image use-case using HADDRSTART, HADDREND, and HADDROFFSET registers

The FlexSPI controller supports a remap feature that helps to support storing dual-images in the memory. The HADDRSTART and HADDREND register can be used to configure an address range to which an offset (configured by HADDROFFSET) is applied. This allows for a single system bus address range to be mapped to two different memory ranges (the unremapped range or the remapped range), so that you can switch between accessing two images at the same system memory address by enabling and disabling the remap. It is used for booting.

NOTE

- Remap in this table includes REMAP register (HADDRSTART[0]) and SWAP register (HADDRSTART[2]).
- Swap function impacts both AHB and IPS address remap (with same mechanism), but remap function only impacts AHB address.

Table 418. FlexSPI address translation

Enable remap	Write to ¹	Encryption ²	Read from ¹	Decryption ²	Description
AHB WR	ADDR	ADDR+OFFSET			The remap function is available if $ADDR \geq HADDRSTART$ & $ADDR < HADDREND$, else no change in address.
AHB RD			ADDR	ADDR + OFFSET	The remap function is available if $ADDR \geq HADDRSTART$ & $ADDR < HADDREND$, else no change in address.
Disable remap	Write to	Encryption	Read from	Decryption	Description
AHB WR	ADDR	ADDR			
AHB RD			ADDR	ADDR	
Enable swap	Write to	Encryption	Read from	Decryption	
IP WR	ADDR	ADDR + OFFSET			The swap function will be available if $ADDR \geq HADDRSTART$ & $ADDR < HADDREND$, else no change in address.

Table continues on the next page...

Table 418. FlexSPI address translation (continued)

Enable remap	Write to ¹	Encryption ²	Read from ¹	Decryption ²	Description
	ADDR	ADDR + OFFSET			The swap function will be available if the ADDR $\geq$ HADDRSTART + OFFSET & ADDR < HADDREND + OFFSET, else no change in address.
AHB WR	ADDR	ADDR+OFFSET			The swap function will be available if the ADDR $\geq$ HADDRSTART & ADDR <HADDREND, else no change in address.
	ADDR	ADDR+OFFSET			The swap function will be available if the ADDR $\geq$ HADDRSTART + OFFSET & ADDR <HADDREND + OFFSET, else no change in address.
IP RD			ADDR	ADDR + OFFSET	The swap function will be available if the ADDR $\geq$ HADDRSTART & ADDR <HADDREND, else no change in address.
			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR $\geq$ HADDRSTART + OFFSET & ADDR < HADDREND + OFFSET, else no change in address.
AHB RD			ADDR	ADDR + OFFSET	The swap function will be available if the ADDR $\geq$ HADDRSTART & ADDR <HADDREND,

Table continues on the next page...

Table 418. FlexSPI address translation (continued)

Enable remap	Write to ¹	Encryption ²	Read from ¹	Decryption ²	Description
					else no change in address.
			ADDR	ADDR+OFFSET	The swap function will be available if the ADDR>=HADDRS TART + OFFSET & ADDR< HADDREND + OFFSET, else no change in address.
Disable swap	Write to	Encryption	Read from	Decryption	
IP WR	ADDR	ADDR			
IP RD			ADDR	ADDR	
AHB WR	ADDR	ADDR			
AHB RD			ADDR	ADDR	

1. Address from AHB or IPS interface.

2. Actual address to the flash.

NOTE

1. HADDRSTART[0] is used to enable the remap feature and HADDRSTART[2] is used to enable the swap feature.
2. SWAP function impacts both AHB and IPS address, but REMAP function only impacts AHB address.
3. SWAP and REMAP must not both be enabled at the same time.
4. The ADDR above is the address from IPS or AHB interface, the OFFSET is same as the OFFSET register (HADDROFFSET).

39.5.4.2 Flash address sent to flash devices

The Flash access start address is determined by AHB address (AHB command) or IPCR0[SFAR] register (IP command). See [Flash access by AHB Command](#) and [Flash access by IP command](#) for more details.

For AHB commands, the FlexSPI controller removes the flash base address automatically when sending flash address to flash devices. The Flash address is sent to devices in two parts: Row address and Column address. For flash devices not supporting Column address, set FLSHxCR1[CAS] to 0, then all flash address bits are sent to flash devices as Row address.

For flash devices, supporting word-addressable feature, the last bit of the address is not needed because flash is read/programmed in terms of 2 bytes. For parallel mode, Flash A1 and Flash B1 (or Flash A2 and Flash B2) are accessed parallelly. As a result, the flash address sent to flash devices should be divided by 2. The following table indicates the relationship of Row address, Column address and flash address (FA).

Table 419. Flash address, row address and column address

Parallel mode	Word-addressable	Row address	Column address	Comment

Table continues on the next page...

Table 419. Flash address, row address and column address (continued)

0	0	FA[31:CAS]	FA[CAS-1:0]	There is no limitation on FA and data size alignment.
0	1	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	0	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	1	FA[31:CAS+2]	FA[CAS+1:2]	FA and data size should be 4 byte aligned.

NOTE

- FA is the flash address without base address.
- If the bit number of the Row or Column flash address to be sent to flash devices defined in instructions is more than the valid value, then the high position bits are supplemented with zero. See [Programmable sequence engine](#) for more details about row or column address instructions.

When parallel mode is enabled or the word-addressable flash is used, there are limitations on the flash start address and data size. This requirement can be addressed by aligning AHB bus access address (for AHB command) or IP command address IPCR0[SFAR] (for IP command) in software. There are two ways to avoid these limitations in specified case.

1. For **AHB Read Command** only:

When AHBCR[READADDROPT] and AHBCR[PREFETCHEN] are both set to 1, or OTFAD is enabled, FlexSPI guarantees the flash access start address and data size are 8 bytes aligned by hardware.

When AHBCR[READADDROPT] is set to 1, or OTFAD is enabled, FlexSPI fetches redundant data to guarantee the flash start address is 8 bytes aligned. When prefetch is enabled (AHBCR[PREFETCHEN] is set to 1), the flash read data size is determined by the size of AHB RX Buffer which is 8 bytes aligned.

2. For **AHB Write Command and Individual Mode** only:

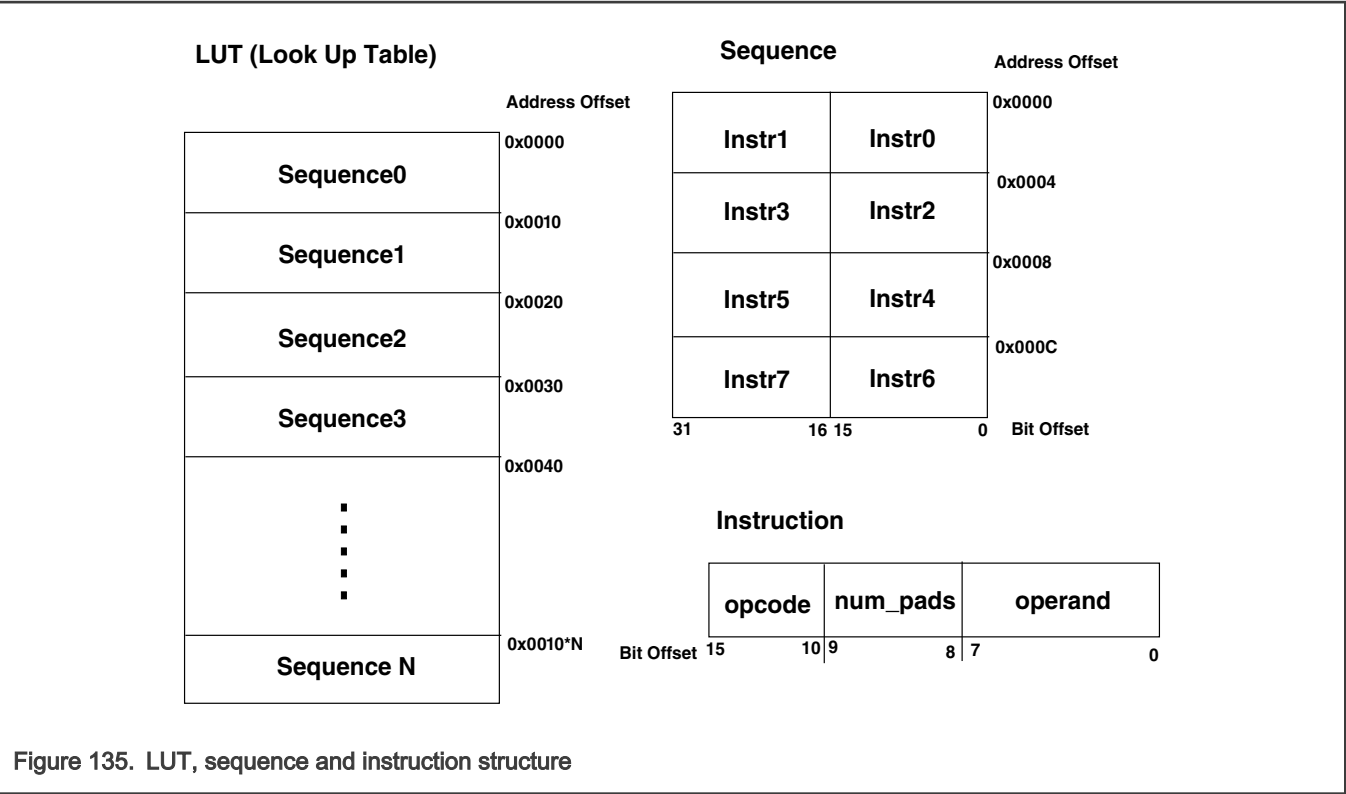
By default, FlexSPI guarantees the flash write access start address and data size are 2 bytes aligned when using DQS as write mask (FLSHCR4[WMENB or WMENA] = 1).

This feature is not applied in parallel mode and must be used if the external device supports write mask feature.

39.5.5 Look up table (LUT)

The LUT is an internal memory to preserve a number of pre-programmed sequences. Each sequence consists of up to 8 instructions which are executed sequentially. When a flash access is triggered by an IP command or an AHB command, the FlexSPI controller fetches the sequence from LUT (sequence index or number) and execute the flash access to generate a valid flash transaction on SPI interface.

The following figure indicates the structure of LUT, sequence and instruction.



NOTE

If the number of needed instructions is less than 8 for a flash transaction, the STOP instruction must be programmed for unneeded instructions (instruction code 8'h00).

For IP and AHB write commands, the FlexSPI controller always executes from instruction pointer 0. For AHB read commands, the FlexSPI controller executes from a saved instruction start pointer. The FlexSPI controller saves the instruction start pointers separately for each flash device. All these saved instruction pointers are zero before the JMP_ON_CS instruction is executed. When the JMP_ON_CS instruction is executed, the operand in the JMP_ON_CS instruction will be saved as instruction start pointer. See [XIP enhanced mode](#) for more details.

The reset value of LUT is unknown because it is implemented as internal memory. LUT must be programmed according to the device connected on board. To protect its contents during a code runover, the LUT can be locked or unlocked to prevent unwanted changes after the LUT has been configured. The key for locking or unlocking the LUT is **0x5AF05AF0**.

The process for locking and unlocking the LUT is shown as follows:

Locking the LUT

1. Write the key (**0x5AF05AF0**) in to the LUT Key Register [LUT Key Register \(LUTKEY\)](#).
2. Write 1b1 to LUTCR[LOCK] and 1b0 to LUTCR[UNLOCK] fields of the LUT Control Register. LUT is not successfully locked when there is another register write access to FlexSPI between these two write accesses.

Unlocking the LUT

1. Write the key (**0x5AF05AF0**) in to the LUT Key Register [LUT Key Register \(LUTKEY\)](#).
2. Write 1b0 to LUTCR[LOCK] and 1b1 to LUTCR[UNLOCK] fields of the LUT Control Register. LUT is not successfully locked when there is another register write access to FlexSPI between these two write accesses.

The lock status of the LUT can be read from LUTCR[LOCK] and LUTCR[UNLOCK].

39.5.6 Programmable sequence engine

The FlexSPI controller implements a programmable sequence engine that executes the sequence from LUT. The FlexSPI controller executes the instructions sequentially, and generates flash transactions on the SPI interface accordingly. The following table is a complete list of the supported instructions.

Table 420. Instruction set

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
CMD_SDR/ CMD_DDR	6'h01/ 6'h21	2'h0 - one pad (Single mode)	Transmit command code to Flash	Command code: Operand[7:0]	Bit number: 8
RADDR_SDR/ RADDR_DDR	6'h02/ 6'h22	2'h1 - two pad (Dual mode) 2'h2 - four pad (Quad mode) 2'h3 - eight pad (Octal mode)	Transmit row address to Flash NOTE For details, see Flash address sent to flash devices	Row_Address[31:0] The Row_Address comes from the AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] The value of operand determines the number of bits in the Row_Address that are sent.
CADDR_SDR/ CADDR_DDR	6'h03/ 6'h23		Transmit column address to Flash NOTE For details, see Flash address sent to flash devices	Column_Address[31:0] The Column_Address comes from the AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] The value of operand determines the number of bits in the Column_Address that are sent.
MODE1_SDR/ MODE1_DDR	6'h04/ 6'h24		Transmit mode bits to Flash	Mode bits: Operand[0]	Bit number: 1
MODE2_SDR/ MODE2_DDR	6'h05/ 6'h25			Mode bits: Operand[1:0]	Bit number: 2
MODE4_SDR/ MODE4_DDR	6'h06/ 6'h26			Mode bits: Operand[3:0]	Bit number: 4
MODE8_SDR/ MODE8_DDR	6'h07/ 6'h27			Mode bits: Operand[7:0]	Bit number: 8
WRITE_SDR/ WRITE_DDR	6'h08/ 6'h28		Transmit program data to Flash device	Program data in IP_TX_FIFO or AHB_TX_BUF	Byte number (data size) is determined by AHB burst size and burst type (AHB command) or IPCR1[IDATSZ] (IP command). For more details about flash read or program data size, see Flash
READ_SDR/ READ_DDR	6'h09/ 6'h29		Receive read data from Flash device	-	

Table continues on the next page...

Table 420. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
			Read data is put into AHB_RX_BUF or IP_RX_FIFO.		access by AHB Command and Flash access by IP command .
LEARN_SDR/ LEARN_DDR	6'h0A/ 6'h2A		Receive read data or Preamble bit from Flash device The FlexSPI controller will compare the data line bits with DLPR register to determine a correct sampling clock phase.	-	Bit number: operand[7:0] Never set operand to zero for LEARN instruction. The value of operand indicates the number of bits to receive and compare to the DLPR value. For example, 8 bits pattern 0x5A on each data line needs to set operand to 8.
DATSZ_SDR/ DATSZ_DDR	6'h0B/ 6'h2B		Transmit read or program data size (byte number) to Flash device	Internal logic Read or program data size for current command sequence	Bit number: operand[7:0] Never set operand to zero or larger than 64 for DATSZ instruction.
DUMMY_SDR/ DUMMY_DDR	6'h0C/ 6'h2C		Leave data lines undriven by the FlexSPI controller. Turnaround cycles are provided from host driving to device driving. num_pads determines the number of pads in input mode.	-	Dummy cycle number (in serial root clock): Operand[7:0] Dummy cycle (N), described in the data sheet of Flash device, is in number of SCLK cycles and this number may be configurable. In SDR mode, SCLK cycle is same as serial root clock. The operand value must be set to N. In DDR mode, SCLK cycle is double the serial root clock cycle. The operand value must be set to 2N, 2N-1 or 2N+1 depending on how dummy is defined in the data sheet of Flash device. See Flash access sequence examples and dummy cycle definition in data sheet of external memory.

Table continues on the next page...

Table 420. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
DUMMY_RWDS_SDR/ DUMMY_RWDS_DDR	6'h0D/ 6'h2D		This instruction is similar to DUMMY_SDR/ DUMMY_DDR instruction. The difference lies in the dummy cycle number. DQS pin is called RWDS in HyperBus specification. See Dummy instruction for more details. Set operand as "Latency count" for HyperBus devices.	-	For read command, dummy cycle number (in serial root clock): (operand[7:0]*4-1) if RWDS (DQS pin) is high; (operand[7:0]*2-1) if RWDS (DQS pin) is low; For write command, dummy cycle number (in serial root clock): (operand[7:0]*4-2) if RWDS (DQS pin) is high; (operand[7:0]*2-2) if RWDS (DQS pin) is low;
JMP_ON_CS	6'h1F	Num_pads setting will be ignored.	Stop execution, deassert CS and save operand[7:0] as the instruction start pointer for next sequence. Normally this instruction is used to support Execute-In-Place enhanced mode. See XIP enhanced mode for more details. This instruction is only allowed for AHB read commands. There is interrupt status bit set (INTR[IPCMDERR] or INTR[AHBCMDERR]) when using this instruction in IP command or AHB write command.	-	No transaction on SPI interface.
STOP	6'h00		Stop execution and deassert CS. Then,	-	

Table continues on the next page...

Table 420. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
			the next command sequence (to the same flash device) starts from instruction pointer 0.		

The programmable sequence engine allows for configuring the LUT according to the external serial device connected on board. The flexible LUT structure is easily adaptable to new command or protocol changes from different vendors.

DDR sequence is a flash access sequence that contains DDR instruction other than DUMMY_DDR, and may optionally contain SDR instructions. The output/input timing on FlexSPI is different for SDR and DDR sequences. Especially note that SDR instructions execute differently if they are included as part of a DDR sequence than they do when used in an SDR sequence. See [FlexSPI input timing](#) and [FlexSPI output timing](#) for more details.

39.5.6.1 Instruction execution on SPI interface

This section describes the detail of instruction execution on SPI interface. For all instructions that receive bits from Flash devices or transmit bits to Flash device, the bit order in one byte is higher on DATA7 than DATA0, and higher on B port than A port.

1. Command Instruction

Command instruction (CMD_SDR/CMD_DDR) is normally used to transmit a command code to the external device. Command code is the 8-bit operand in instructions. Command code is sent to both port A and port B in parallel mode. See [Flash access sequence examples](#) for more details.

2. Address Instruction

Address Instructions (RADDR_SDR/RADDR_DDR/CADDR_SDR/CADDR_DDR) are normally used to send the Flash access start address (Row Address or Column Address) to external device. The bits of Row address or Column address are determined by FlexSPI according to AHB access address or IP command address. See [Flash address sent to flash devices](#) for more details. The number of address bits to send is the operand value in instruction code. The bits in Row Address or Column Address are sent to both port A and port B in parallel mode. For example, assuming that num_pads is configured as 8, for memories that read the flash address on 4 SCK edges, the sum of bits in CADDR and RADDR instructions must be 32. Otherwise, the FlexSPI will not generate 4 SCK edges of address phase. See [Flash access sequence examples](#) for more details.

3. Mode Instruction

Mode Instructions (MODEx_SDR/MODEx_DDR) are normally used to send mode bits to external devices. Mode bits are the (lower) bits of operand. For example, the bit number is 1 for MODE1_SDR/MODE1_DDR, 2 for MODE2_SDR/MODE2_DDR, 4 for MODE4_SDR/MODE4_DDR and 8 for MODE8_SDR/MODE8_DDR. Note that pad number should be no more than mode bit number. For example, it is not allowed to set num_pads to 2'b11 (Octal mode) for MODE4_* instructions. Mode bits will be sent to both port A and port B in parallel mode. See [Flash access sequence examples](#) for more details.

4. Data Size Instruction

Data Size Instructions (DATSZ_SDR/DATSZ_DDR) are used to send the size of program data or read data (byte number) to external device. This instruction is normally used in FPGA application when the memory space in external device acts similarly as a FIFO. The external device needs data size information to determine how much data will be popped from or pushed into internal FIFO. The bit number is the operation value in the data size instructions. Data size is sent to both port A and port B in parallel mode. See [Flash access sequence examples](#) for more details.

5. Write Instruction

Write Instructions (WRITE_SDR/WRITE_DDR) are normally used to send program data to external device. Programming data is fetched from IP_TX_FIFO (IP Command) or AHB_TX_Buffer (AHB command). For more details about flash program data size, see [Flash access by AHB Command](#) and [Flash access by IP command](#). The byte order for program data is always from low to high. Odd bytes are sent on port A and even bytes are sent on port B in parallel mode. See [Flash access sequence examples](#) for more details.

6. Read Instruction

Read Instructions (READ_SDR/READ_DDR) are normally used to receive flash data from external device. Received data will be put into IP_RX_FIFO (IP Command) or AHB_RX_Buffer (AHB command). For more detail about flash read data size, see [Flash access by AHB Command](#) and [Flash access by IP command](#). The byte order for reading data is always from low to high. Odd bytes are received from port A and Even bytes are received from port B in parallel mode. See [Flash access sequence examples](#) for details.

7. Dummy Instruction

Dummy Instructions (DUMMY_SDR/DUMMY_DDR/DUMMY_RWDS_SDR/DUMMY_RWDS_DDR) are used to provide turnaround cycles on the SPI interface. During dummy instruction, neither FlexSPI controller nor external device drives the SPI interface. See [Programmable sequence engine](#) for more details about dummy cycle number.

DUMMY_RWDS_DDR can be used for HyperBus device which uses RWDS pin to indicate if extra latency count is needed. DUMMY_RWDS_SDR is reserved. The FlexSPI controller checks DQS pin input level at the 4th cycle after SCLK output toggling is enabled. DQS pin is called RWDS in HyperBus specification. See [Flash access sequence examples](#) for details.

NOTE

The FlexSPI releases the bus after at least one cycle. Therefore, to avoid data contention, the Num_pads value for the dummy commands should be configured to match the number of data lines used for the external memory.

8. Learn Instruction

Learn Instructions (LEARN_SDR/LEARN_DDR) are used to determine the correct sample clock phase for flash read data sampling. External device drives read data (or data learning pattern) bits on the FlexSPI interface. The FlexSPI controller will compare the data line bits to the DLPR register to determine a correct sampling clock phase.

Clock phase selection is automatically updated after executing learn instructions. See [Data learning feature](#) for more details. The byte number is the operand value in learn instructions. FlexSPI checks the same data pattern on each data line.

The byte order is byte 0, byte 1, byte 2, byte 3, byte0, byte 1, ... (byte 0 is DLPR register bit 7-0, byte 1 is DLPR register bit 15-8, byte 2 is DLPR register bit 23-16, byte 3 is DLPR register bit 31-24). The bit order is from high to low in each byte. See [Flash access sequence examples](#) for details.

39.5.6.2 Flash access sequence examples

Following is an example for SDR single I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

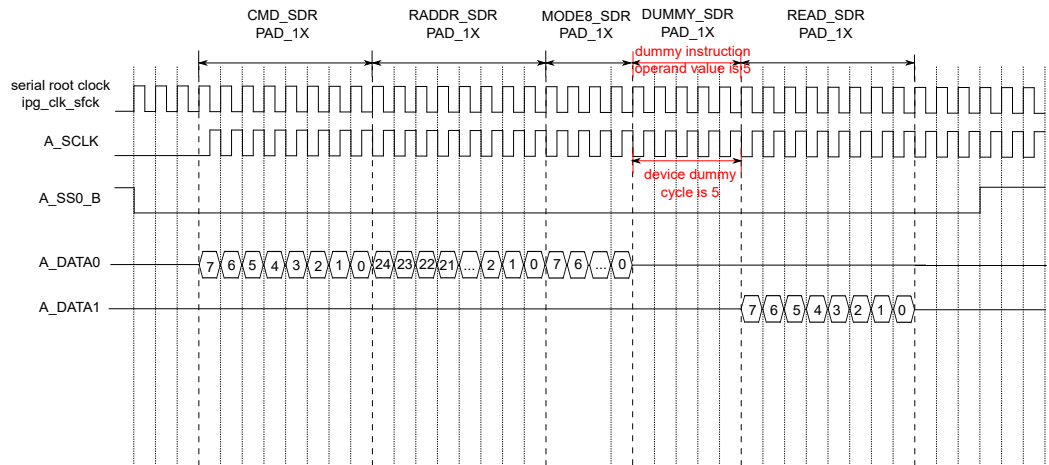


Figure 136. Flash access sequence example (SDR single I/O read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK (on Cypress S25FS512S data sheet).

Following is an example for SDR quad I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

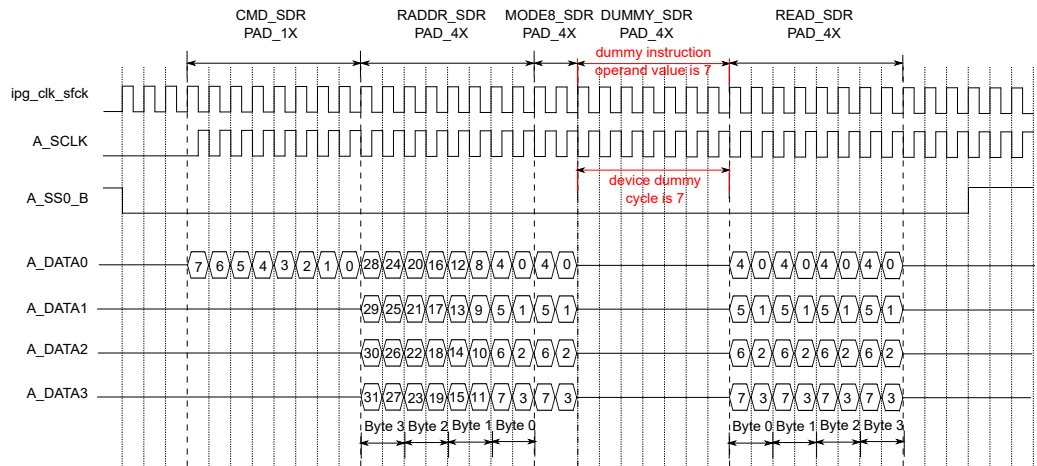


Figure 137. Flash access sequence example (SDR quad I/O read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK (on Cypress S25FS512S data sheet).

Following is an example for DDR quad I/O read sequence (Cypress Serial Nor Flash S25FS512S) in parallel mode.

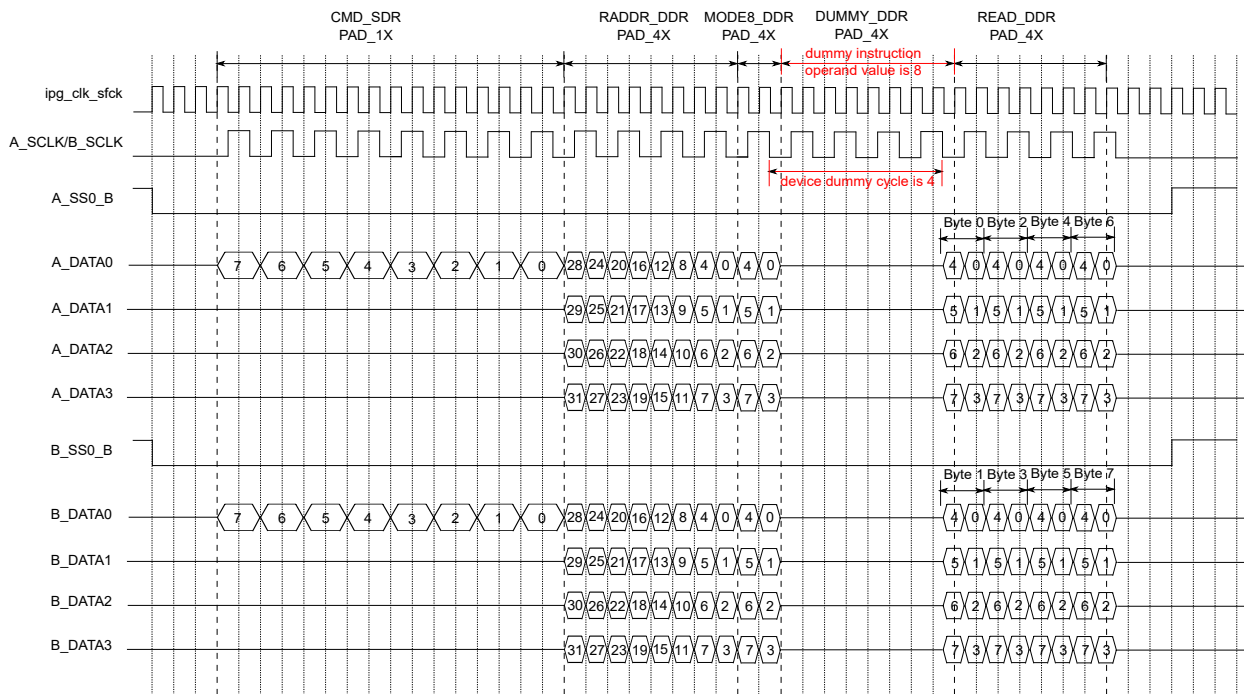


Figure 138. Flash access sequence example (DDR quad I/O read sequence)

Following is an example indicating learn instruction (not for a specified flash device) in individual mode.

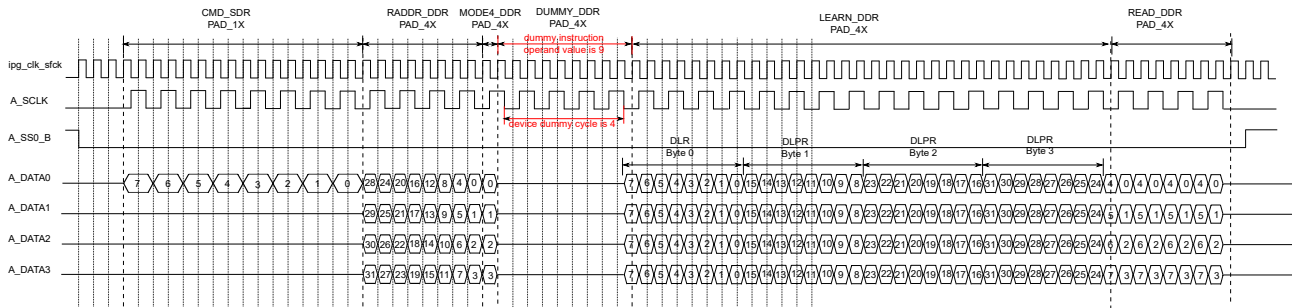


Figure 139. Flash access sequence example (data learning)

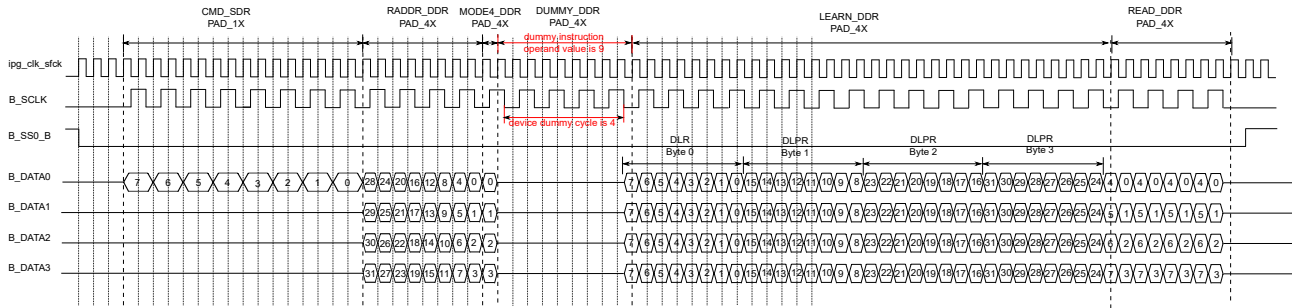


Figure 140. Flash access sequence example (data learning) (Channel B)

NOTE

- FlexSPI dummy instruction starts and ends at the rising edge of serial root clock.
- Device dummy cycle starts and ends at the falling edge of SCLK.
- The operand value of DUMMY_DDR instruction is odd because the total cycle number before DUMMY_DDR cycle is odd.

Following is an example for HyperBus device read transaction (single latency count) in individual mode.

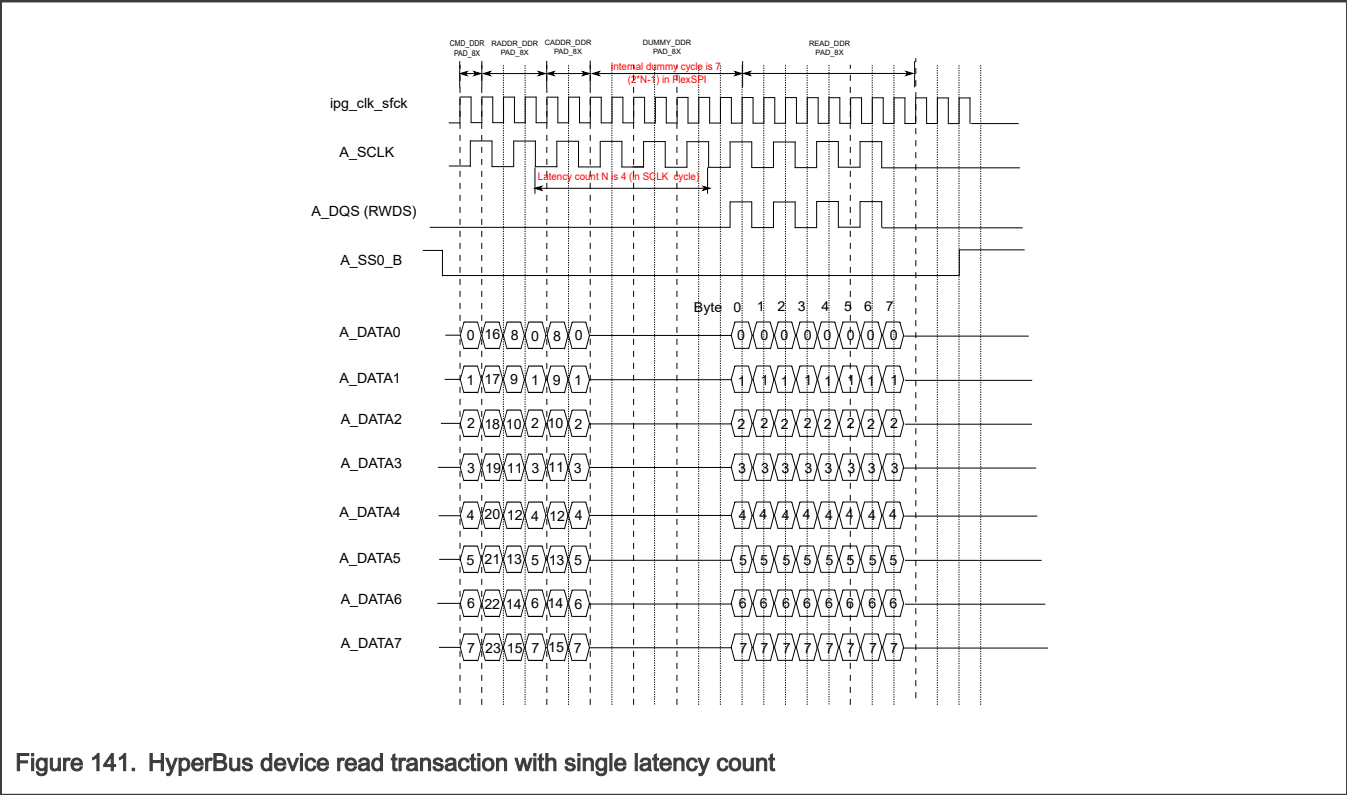


Figure 141. HyperBus device read transaction with single latency count

NOTE

The FlexSPI module continues to drive the clock until it has latched in all of the read data it is expecting. On-chip delay for the data to reach the FlexSPI module can increase the number of SCK clock pulses.

Following is an example for HyperBus device read transaction (additional latency count) in individual mode.

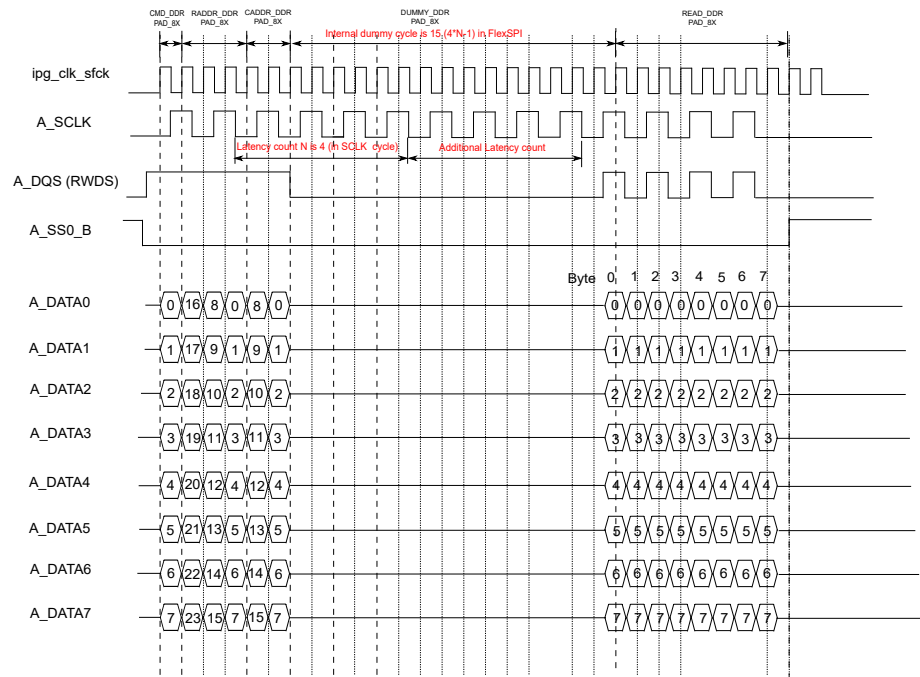


Figure 142. HyperBus device read transaction with additional latency count

Following is an example for HyperBus device write transaction (single latency count) in individual mode.

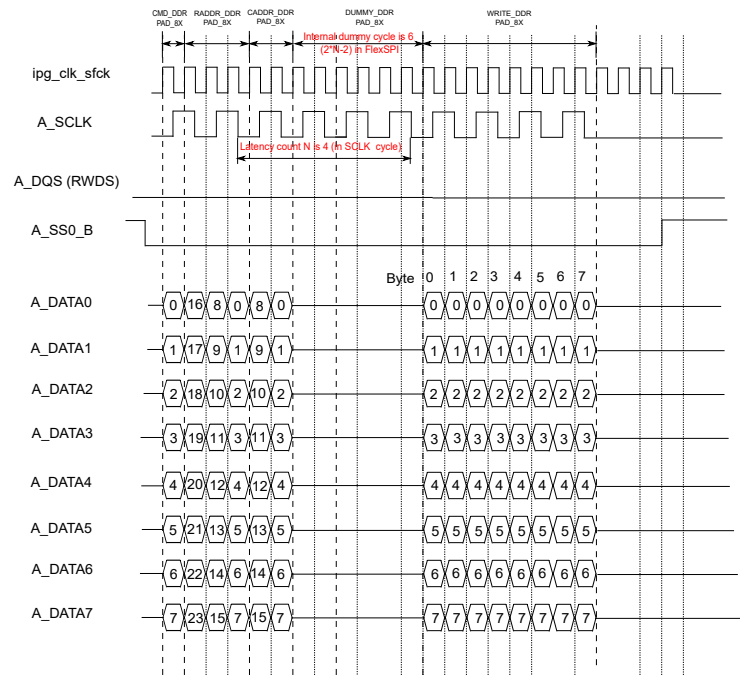


Figure 143. HyperBus device write transaction with single latency count

Following is an example for HyperBus device write transaction (additional latency count) in individual mode.

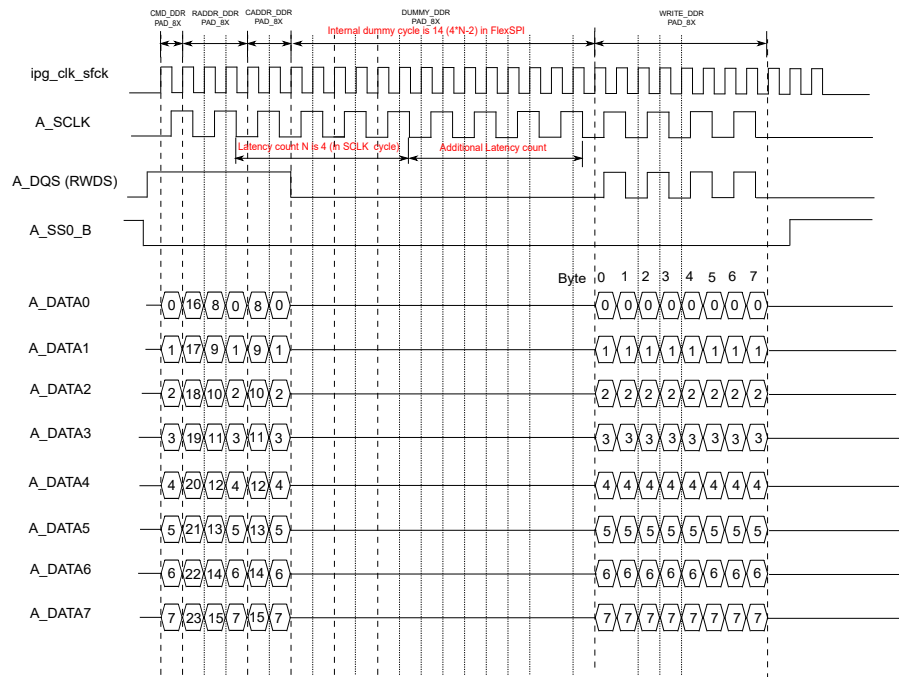


Figure 144. HyperBus device write transaction with additional latency count

39.5.7 Clocks

This section describes clocks and special clocking requirements of the FlexSPI module.

Table 421. Clock Usage

Clock Name	Description	Comment
serial clock root (ipg_clk_sfck)	Root clock for Serial domain	-
ahb clock (hclk)	AHB Bus clock	hclk frequency is higher than 1/4 of serial root clock so that internal async FIFO is never full. This clock is used for internal logic in FlexSPI. It's frequency will be same as SPI SCK frequency in SDR mode if MCR0[SERCLKDIV] = 0.
ipg clock (ipg_clk)	IPS Bus clock	Register access clock. ipg_clk must be equal to hclk or an integer divide of hclk. For example, ipg_clk = hclk/2.
SCLK	FlexSPI serial clock. Output clock on SCLK pin	Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence.
DQS_OUT	Dummy Read Strobe output	Same frequency as SCLK. Clock output toggles during READ/LEARN instructions.

Table continues on the next page...

Table 421. Clock Usage (continued)

Clock Name	Description	Comment
		The DQS ensures that the data can be strobed correctly. It is only used for data strobe instead of command or address strobe. The data includes real useful read data and data learning pattern.
DQS_IN	Sample clock for RX Data	Same frequency as SCLK. Sample clock comes from loopbacked dummy read strobe, loopbacked SCLK or Flash provided read strobe.

39.5.8 Interrupts

This section describes all the interrupts that the FlexSPI module generates.

- **IP command done interrupt**

When an IP command is finished, an interrupt is generated if INTEN[IPCMDDONEEN] is set to 0x1.

- **IP command grant error interrupt**

When an IP command grant timeout occurs (bus not granted after MCR0[IPGRANTWAIT] * 1024 AHB clock cycles), an interrupt is generated if INTEN[IPCMDGEEEN] is set to 0x1. See [Overview of error category, flags and triggered sources](#) for more details.

- **AHB command grant error interrupt**

When an AHB command grant timeout occurs (bus not granted after MCR0[AHBGRANTWAIT] * 1024 AHB clock cycles), an interrupt is generated if INTEN[AHBCMDGEEEN] is set to 0x1. See [Overview of error category, flags and triggered sources](#) for more details.

- **IP command error interrupt**

When there is a command check error or a command execution error for an IP command, an interrupt is generated if INTEN[IPCMDDERREN] is set to 0x1. See [Overview of error category, flags and triggered sources](#) for more details.

- **AHB command error interrupt**

When there is a command check error or a command execution error for AHB command, an interrupt is generated if INTEN[AHBCMDDERREN] is set to 0x1. See [Overview of error category, flags and triggered sources](#) for more details.

- **IP_RX_FIFO watermark available interrupt**

When the fill level of IP_RX_FIFO reaches the watermark level (IPRXFCR[RXWMRK]), an interrupt is generated if INTEN[IPRXWAEN] is set to 0x1.

- **IP_TX_FIFO watermark empty interrupt**

When the empty level of IP_TX_FIFO reaches the watermark level (IPTXFCR[TXWMRK]), an interrupt is generated if INTEN[IPTXWAEN] is set to 0x1.

- **Data learning failed interrupt**

When there is no valid sample clock phase found after LEARN instruction is executed, an interrupt is generated if INTEN[DATALEARNFAILEN] is set to 0x1. See [Overview of error category, flags and triggered sources](#) for more details.

- **Sequence execution timeout interrupt**

When a sequence execution time exceeds the timeout wait time (MCR1[SEQWAIT]), an interrupt is generated if INTEN[SEQTIMEOUTEN] is set to 0x1. For example, the following flash read command sequence lasts about 8000000 cycle (ipg_clk_sfck). If SEQWAIT is set to 0xFFFF, a sequence timeout interrupt is generated.

- Triggered by IP command
- Flash read data size is 0x1000000 bytes
- Flash accesses in single mode and SDR mode

- **AHB bus error interrupt**

When AHB bus response timeout occurs, or AHB read access during OTFAD is enabled and FlexSPI is still fetching the key blob, an interrupt is generated if INTEN[AHBBUSERROREN] is set to 0x1. For example, AHB read sequence is not configured properly in LUT (such as without READ instruction). No data is read from external device. As a result, the FlexSPI cannot reach the read data in AHB_RX_Buffers for AHB read command.

- **SCLK stopped by write command interrupt**

When IP_TX_FIFO is empty during write command sequence execution, the FlexSPI stops SCLK output clock toggling and waits for write data filling. At this time, this interrupt is generated if it is enabled by INTEN register.

- **SCLK stopped by read command interrupt**

When IP_RX_FIFO is full during read command sequence execution, the FlexSPI stops SCLK output clock toggling and waits for reading the read data from IP_RX_FIFO. At this time, this interrupt is generated if it is enabled by INTEN register.

- **Key done interrupt**

When OTFAD is enabled and key blob processing is enabled, an interrupt is generated after key blob processing is done and no error is detected if INTEN[KEYDONEEN] is set to 0x1.

- **Key error interrupt**

When OTFAD is enabled and key blob processing is enabled, an interrupt is generated if an error occurs during key blob processing if INTEN[KEYERROREN] is set to 0x1. Once this interrupt is set, it can only be cleared by reset.

39.5.9 OTFAD Support

This section describes OTFAD(On-The-Fly AES Decryption) support of the FlexSPI module.

After power up if the OTFAD and key fetches are enabled in the system, FlexSPI automatically starts fetching the key blobs from location 0 of the flashA1. The key fetches continue until the OTFAD receives and processes all key blobs. During the key blob fetch duration any AHB access is responded to with an error response and INT[AHBBUSERROR] is set. During Key blob fetch any IPS access is not allowed. Key blob processing result will be represented by INT[KEYDONE] or INT[KEYERROR]

After the OTFAD processes all key blobs, any reads from the FlexSPI are first decrypted by the OTFAD engine and then returned on the AHB bus. In case OTFAD is enabled, the FlexSPI will always enable prefetch, only AHBCR[READSZALIGN] can disable the prefetch. Read Resume function will also be disabled. All read access address will be 8 bytes aligned automatically regardless the setting of AHBCR[ADDROPT]. If OTFAD is enabled, the prefetch never crosses the 1KB boundary.

39.5.10 Flash access by IP command

Flash access could be triggered by IP command in following steps.

1. Fill IP TX FIFO with programming data if this is a programming command (for instance, programming flash data and flash status registers.)
2. Set flash access start address (IPCR0[SFAR]).
3. Write the IPCR1 with the read/program data size, sequence index, and sequence number (IPCR1[IDATSZ, ISEQID, and ISEQNUM]).
4. Trigger flash access command by writing 1 to IPCMD[TRG].
5. Wait for the INTR[IPCMDDONE] bit to set or for the IP command done interrupt to fire indicating the command has completed on the FlexSPI interface.

NOTE

- IP TX FIFO can be filled before or after writing IPCR0/IPCR1/IPCMD register. If the SFM command is started with IP TX FIFO empty, FlexSPI stops SCLK toggling to wait for TX data ready automatically.
- IPCMD register must be written after writing IPCR0/IPCR1 register.
- Multiple Command sequences (8 at most) could be issued by one IP command.
- It is not allowed to issue another IP command before the previous IP command is finished. The behavior is unknown in this case.

If this is a Read command, all read data from Flash is put into IP RX FIFO. Software needs to read out data from IP RX FIFO by IP bus. When IP RX FIFO is full and there is more data to be read from Flash device, the FlexSPI stops SCLK output clock toggling until there is free space in IP RX FIFO. See [SCLK stop feature](#) for more details.

The detail of triggered serial Flash command is as following:

- Flash access start address:
Determined by register field IPCR0[SFAR]
- Flash chip select:
Determined by flash access address and Flash size setting (FLSHxCR0[FLSHSZ]).
- Flash command sequence index and sequence number:
The sequences indexed from IPCR1[ISEQID] to (IPCR1[ISEQID] + IPCR1[ISEQNUM]) in LUT are executed by FlexSPI sequentially.
- Flash individual/parallel access mode:
Determined by IPCR1[IPAREN].
- Flash read/program data size:
 - If IPCR1[IDATSZ] value is non-zero, flash read/program data size (in bytes) is IPCR1[IDATSZ].
 - If IPCR1[IDATSZ] value is zero, flash read/program data size (in bytes) is determined by the operand value in the READ/WRITE instruction.

NOTE

- Software must make sure the last sequence index never exceeds the LUT sequence number (IPCR1[ISEQID] + IPCR1[ISEQNUM] < 16).
- Data size is applied to every command sequence if sequence number is more than one.
- Data size is ignored if there is no WRITE/READ instruction in the command sequence.

IP command request is sent to arbitrator after triggered by software. It is not executed on the FlexSPI interface until granted by arbitrator. See [Command arbitration](#) for more details.

39.5.10.1 Reading data from IP RX FIFO

The FlexSPI puts the read data from external device into IP RX FIFO for IP command.

The data stored in the IP RX FIFO can be read using either IPS/register or system/AHB bus transactions:

- By IP bus

The FlexSPI pushes read data into IP RX FIFO in terms of 64 bits every time it receives 64 bits data from external device. When the number of read data bits is not 64 bits aligned, the FlexSPI pushes additional zero bits into IP RX FIFO for last push.

The data in IP RX FIFO can be read by processor or DMA. The following shows the flow details for processor and DMA reading:

1. Reading by processor

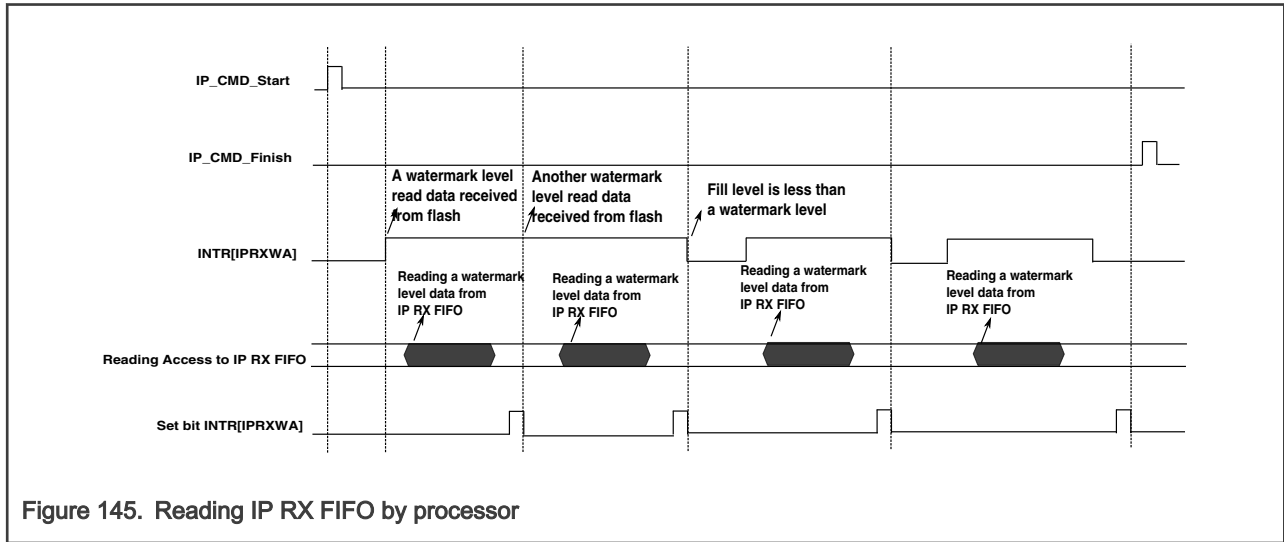
To read data in IP RX FIFO by processor, set the register as below:

- Set the register field IPRXFCR[RXDMAEN] to 0.
- Set the watermark level to $(IPRXFCR[RXWMRK]+1) * 8$ bytes by IPRXFCR[RXWMRK].
- Set the register field INTEN[IPRXWAEN] to enable IP RX FIFO watermark available interrupt (optional).

The processor needs to poll the register INTR[IPRXWA] or wait for IP RX FIFO watermark available interrupt before reading data in IP RX FIFO to make sure that there is a watermark level data filled in IP RX FIFO before reading.

After reading a watermark level data from IP RX FIFO, software needs to set the register bit INTR[IPRXWA] to pop out a watermark level data from IP RX FIFO.

The following diagram indicates the reading flow from IP RX FIFO by processor.



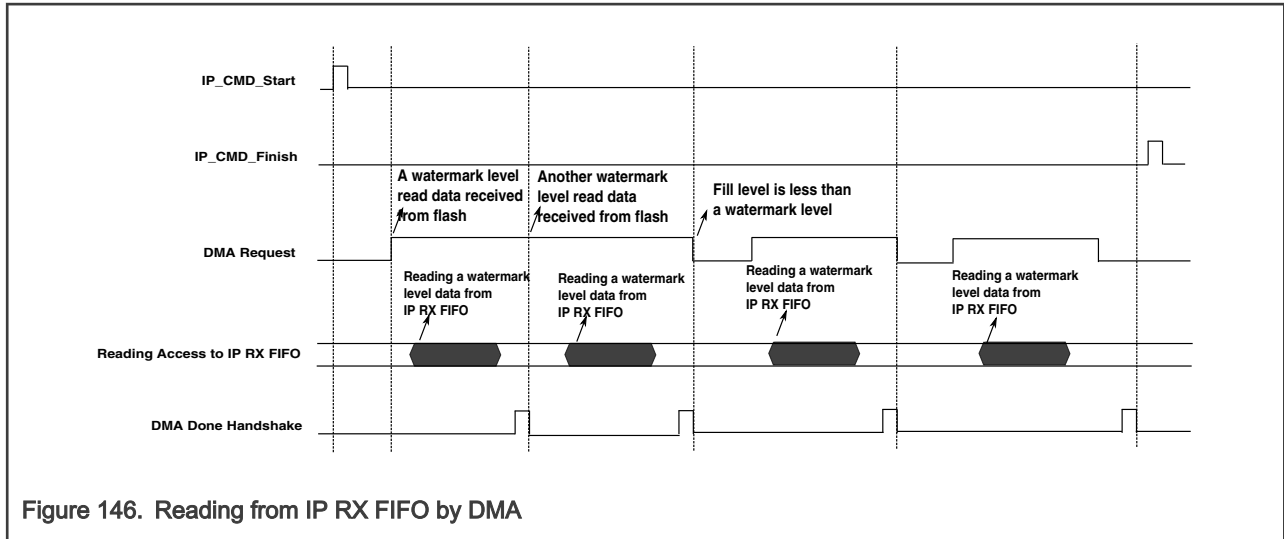
NOTE

- The processor needs to read out a watermark level data from IP RX FIFO each time before setting register bit INTR[IPRXWA].
- It is supported that the total flash read/program data size is not multiple of watermark level. In this case, the reading data size from IP RX FIFO is less than a watermark level for the last time, software polls IPRXSTS[FILL] field not INTR[IPRXWA]. After copying all the data from IP RX FIFO and all command sequences to Flash are finished ($INTR[IPCMDONE]=1$), software clears IP RX FIFO by setting IPRXFCR[CLRIPRXF]. Otherwise, the reading data is wrong for the next reading command to Flash.
- IP RX FIFO data is popped out by writing 0x1 to the register INTR[IPRXWA] bit, not by each read access to IP RX FIFO.

2. Reading by DMA

To read IP RX FIFO by DMA, set the register as below:

- Set the register field IPRXFCR[RXDMAEN] to 1.
- Set the watermark level to $(IPRXFCR[RXWMRK]+1) * 8$ bytes by IPRXFCR[RXWMRK].
- Configure DMA to transfer the same amount of data as indicated by watermark level: $(XFERCFGa[XFERCOUNT] + 1) * XFERCFGa[WIDTH]$.

**NOTE**

- DMA request is generated when the fill level of IP RX FIFO is higher than (or equal to) watermark level. This request is not pulse valid but level valid.
- DMA reads out watermark level data from IP RX FIFO each time (set DMA transfer size with the same value as indicated by watermark level).
- DMA needs to return a done handshake (pulse valid) to the FlexSPI each time it finishes reading a watermark level data.
- IP RX FIFO data is popped out by DMA done handshake, not by each read access.
- It is not supported that the total read/write data size (that is, when executing multiple DMA transfers) is not multiple of watermark level, because DMA does not know when the data is ready for the last reading. It makes the DMA driver too complex to poll IPRXFSTS [FILL] field.

39.5.10.2 Filling data to IP TX FIFO

The programming data is put into IP TX FIFO and then is transmitted to Flash by the FlexSPI. It can be filled by memory space

- By IP bus

IP TX FIFO is popped with 64 bits data every time the FlexSPI fetches data for transmitting. If the programming data size is not multiple of 64 bits, the number of last popped valid bits is less than 64 bits. No error occurs because these invalid bits are not transmitted to Flash at all.

IP TX FIFO can be filled with programming data by processor or DMA.

To fill data into IP TX FIFO by processor, set the register as below:

- Set the register field IPTXFCR[TXDMAEN] to 0.
- Set the watermark level to $(IPTXFCR[TXWMRK]+1) * 8$ bytes by IPTXFCR[TXWMRK].
- Set the register field INTEN[IPTXWEEN] to enable IP TX FIFO watermark empty interrupt (optional).

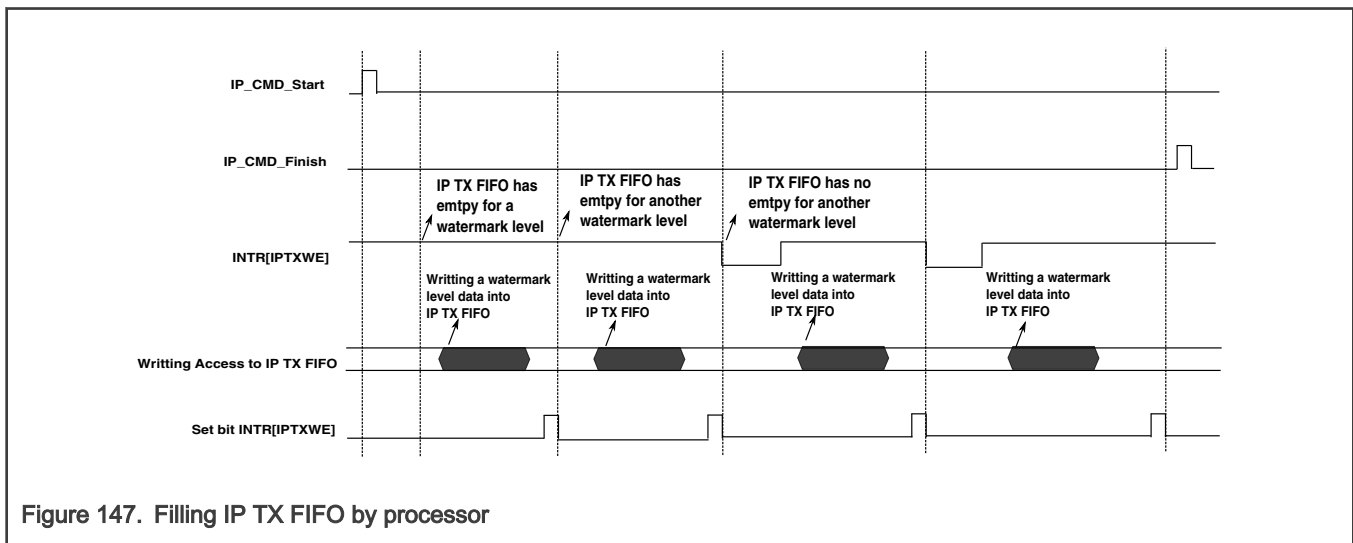
The processor needs to poll register INTR[IPTXWE] or wait for IP TX FIFO watermark empty interrupt before filling data into IP TX FIFO. This is to make sure there is enough space for a watermark level data filling before filling.

After filling a watermark level data into IP TX FIFO, set register bit INTR[IPTXWE]. This pushes the watermark level data into IP TX FIFO (write pointer is incremented).

NOTE

IP TX FIFO data is not pushed by each write access, only pushed by set INTR[IPTXWE] bit.

The following diagram indicates the filling flow to IP TX FIFO by processor.

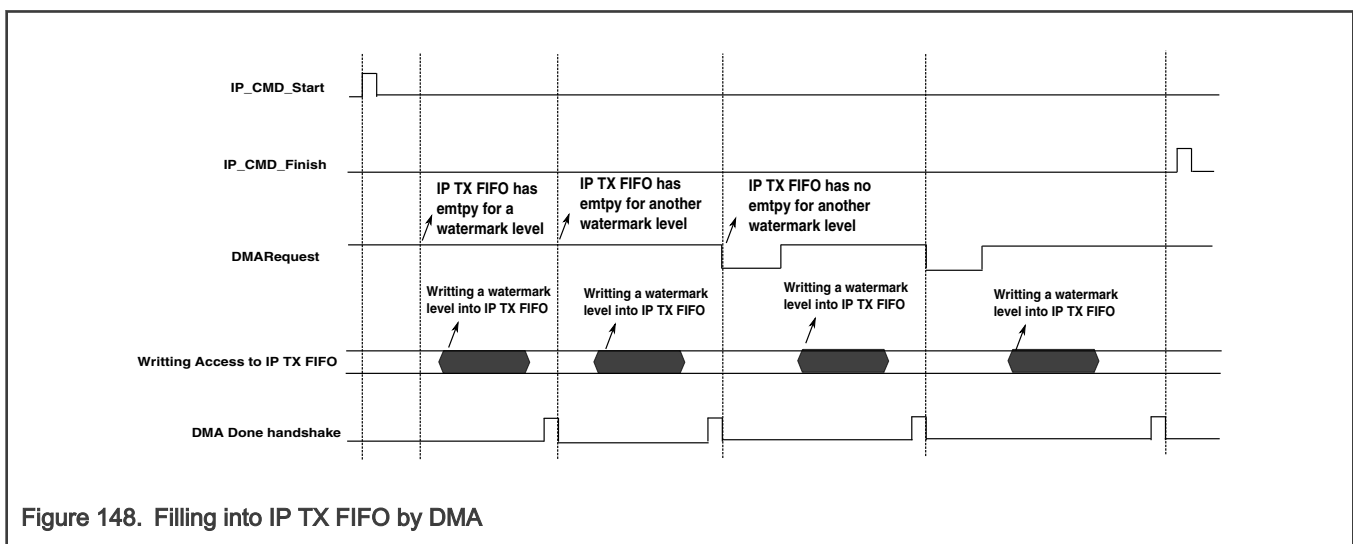


NOTE

- The processor needs to fill a watermark level data into IP TX FIFO each time.
- It is allowed that the total write data size is not multiple of a watermark level. In this case, the write data size is less than a watermark level for the last time. After filling all data into IP TX FIFO and all command sequences to Flash are finished (INTR[IPCMDONE]=1), the processor clears IP TX FIFO by setting IPTXFCR[CLRIPTXF]. Otherwise, the programming data is wrong for the next programming command to Flash.

To fill data into IP TX FIFO by DMA, set the register as below:

- Set the register field IPTXFCR[TXDMAEN] to 1.
- Set the watermark level to $(IPTXFCR[TXWMRK]+1) * 8$ bytes by IPTXFCR[TXWMRK].
- Configure DMA to transfer the same amount of data as indicated by watermark level: $(XFERCFGa[XFERCOUNT] + 1) * XFERCFGa[WIDTH]$.



NOTE

- DMA request is generated when there is empty space more than watermark level in IP TX FIFO. This request is not pulse valid but level valid.
- DMA fills watermark level data into IP TX FIFO each time (set DMA transfer size with the same value as indicated by watermark level).
- DMA returns a done handshake (pulse valid) to FlexSPI each time it finishes filling a watermark level data.
- IP TX FIFO data is pushed in by DMA done handshake, not by each read access.
- It's allowed that total program data size is not multiple of watermark (that is when executing multiple DMA transfers) level. After filling all data into IP TXFIFO and all command sequences to Flash are finished (INTR[IPCMDDONE]=1), clear IP TX FIFO by set IPTXFCR[CLRTXF]. Otherwise, the programming data is wrong for the next programming Command to Flash.
- For transfers larger than 1 KB using RX/TX FIFO, user must use DMA chained transfers (See DMA controller).
- DMA transfer using FlexSPI is channel bounded, even if it uses trigger instead of requests.

For FlexSPI:

- DMA_FLEXSPI0_RX: 22
- DMA_FLEXSPI0_TX: 23
- DMA_FLEXSPI1_RX: 25
- DMA_FLEXSPI1_TX: 26

39.5.11 Flash access by AHB Command

Flash can be accessed by AHB bus directly on AHB address space. This address space is mapped to serial Flash memory in the FlexSPI. AHB bus accesses to this address space trigger Flash access command sequences as needed.

For AHB read access to serial Flash memory, the FlexSPI fetches data from Flash into AHB RX Buffer and then return the data on AHB bus. For AHB write access to serial Flash memory, the FlexSPI buffers AHB bus write data into AHB TX Buffer and then transmit to serial Flash memory.

There is no software configuration or polling needed for AHB command, except for FlexSPI initialization. As with normal AHB slave, AHB master accesses external Flash device transparently.

AHB command is normally used to access serial Flash memory space. IP command must be used to access the control and status registers or other spaces such as OTP in external Flash device.

The following sections describe the AHB commands for read and write access in more detail.

39.5.11.1 AHB write access to Flash

For AHB write access to Flash, the FlexSPI buffers the write data from AHB bus into internal AHB TX Buffer and then transmit them to Flash. The FlexSPI only buffers write data for one AHB burst. The following diagram indicates the hardware operation in response to AHB write access to Flash.

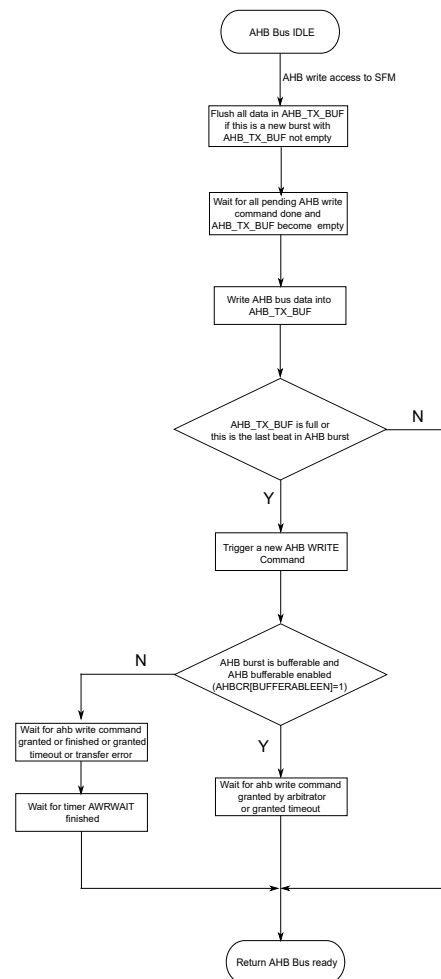


Figure 149. Hardware operation in response to AHB write access to Flash

NOTE

1. AHB TX Buffer may not be empty when AHB bus is IDLE or a new burst comes, if the previous AHB burst is the INCR write burst. A new AHB write command is triggered to flush all data in AHB TX Buffer to external Flash device. The new burst is held until this AHB write command finishes.
2. When AHB write command triggered, if current access is bufferable, AHB ready is returned after this AHB write command is granted. If current access is non-bufferable, AHB ready is returned after this AHB write command finishes.

The FlexSPI triggers new AHB write command in the following cases:

- This beat is the last one in the current AHB burst (any burst type except for INCR).
- AHB TX Buffer is full after the current beat data is buffered.
- AHB bus becomes IDLE or a new burst comes with AHB TX Buffer not empty.

Below shows the detail information about the triggered AHB write command:

- Flash Access Start Address:

Determined by AHB burst address. The FlexSPI records the start address for the data in AHB TX Buffer, which is used as flash access start address .

- Flash Chip Select:

The FlexSPI determines the chip selection by the settings of flash access start address and flash size.

- Flash Command Sequence Index:

Determined by FLSHxCR2[AWRSEQID].

- Flash Command Sequence Number:

Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash access command sequences are triggered every time for AHB write command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT are executed sequentially.

- Flash Access Mode (Individual/Parallel)

Determined by AHBCR[APAREN].

- Flash Data Size:

Determined by byte number of buffered data in AHB TX Buffer.

The following examples indicate internal logic for AHB write access to Flash that AHB_TX_BUF is 64 Bytes (8*64bits) .

- AHB INCR/64bit/bufferable burst with address sequence 0x8, 0x10, 0x18, 0x20, ..., 0x50 (10 beat total):

Two AHB write commands are triggered: the first command with flash start address 0x8 and data size 0x40; the second command with flash start address 0x48 and data size 0x10. See [Figure 150](#).

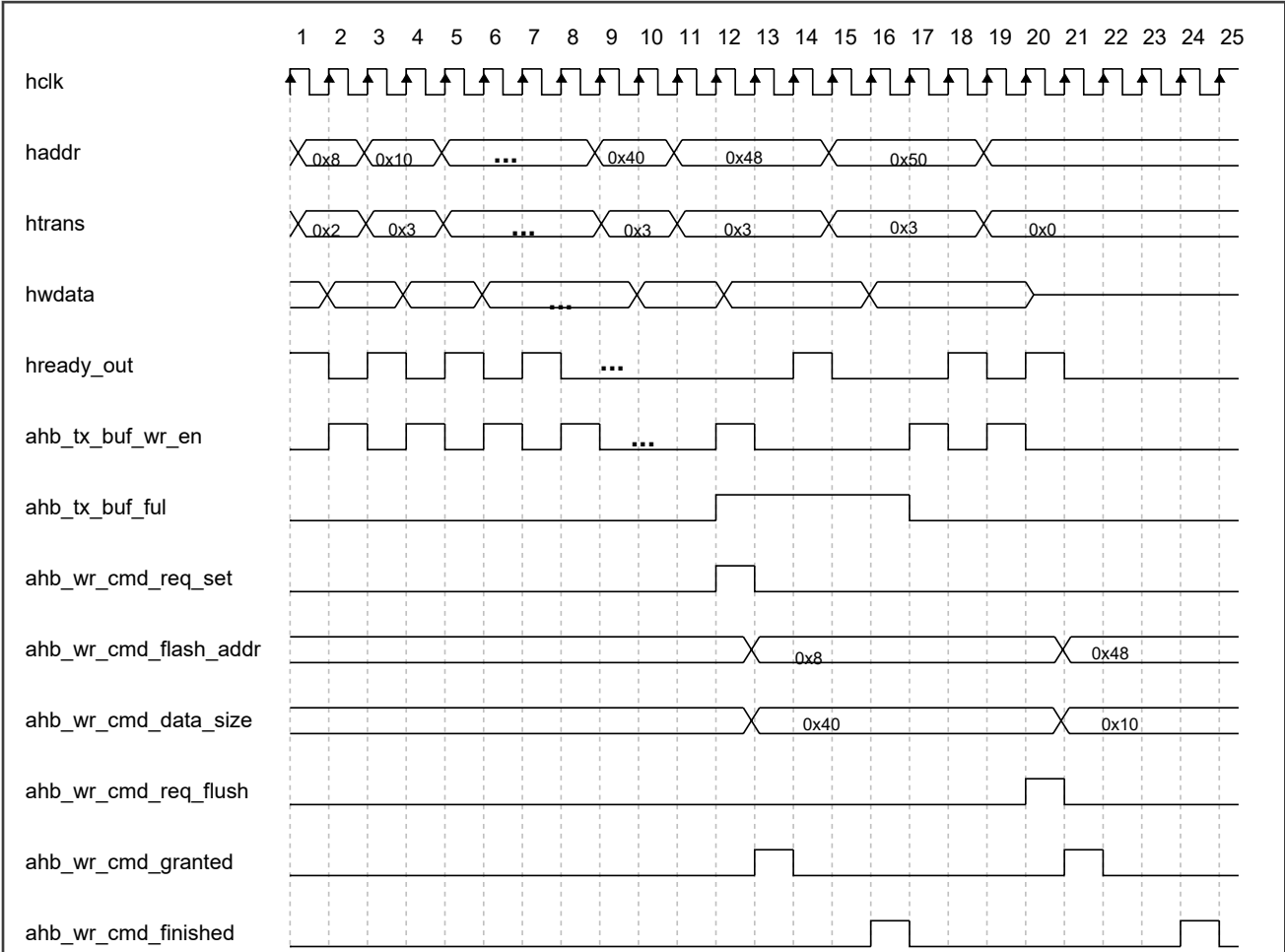
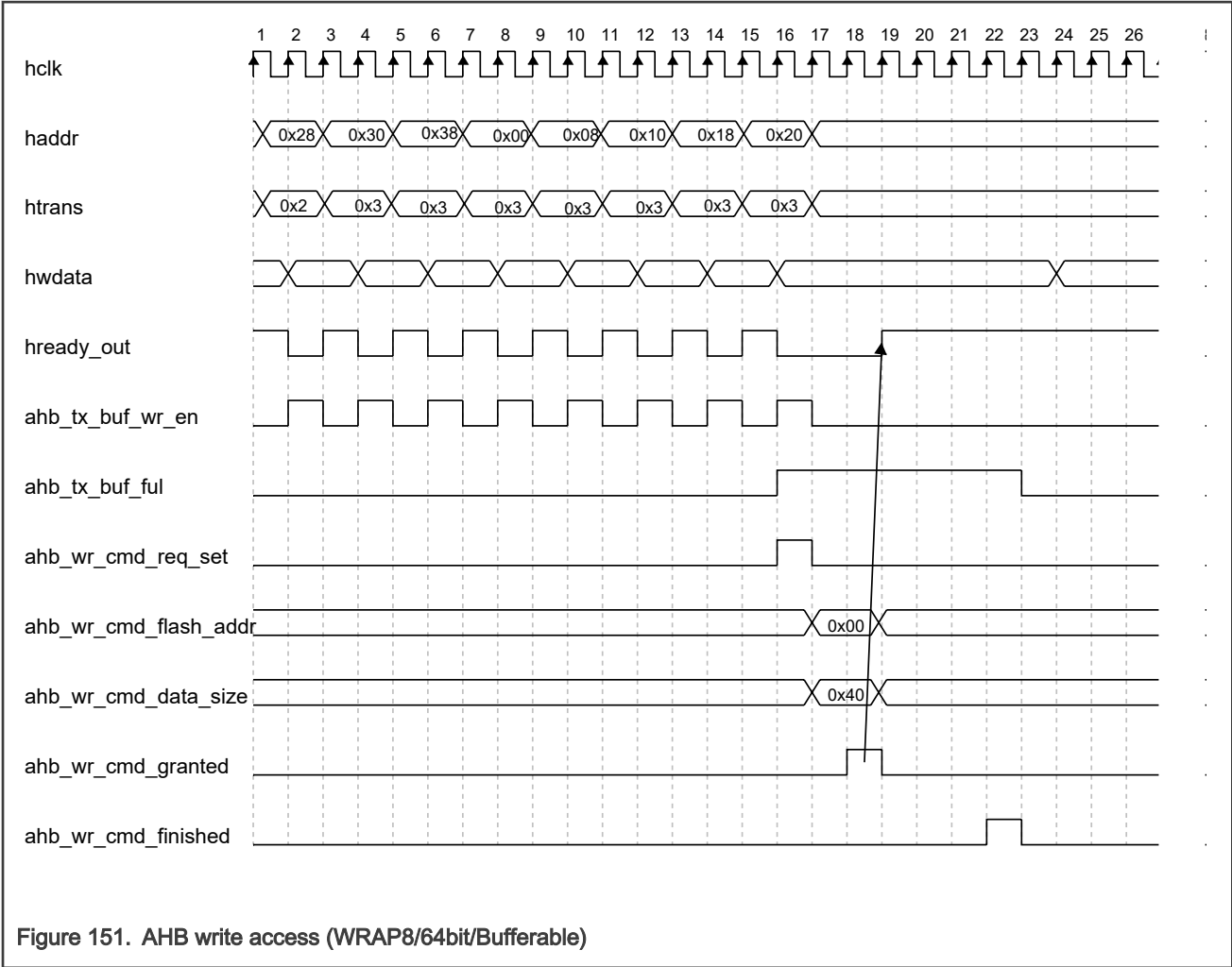
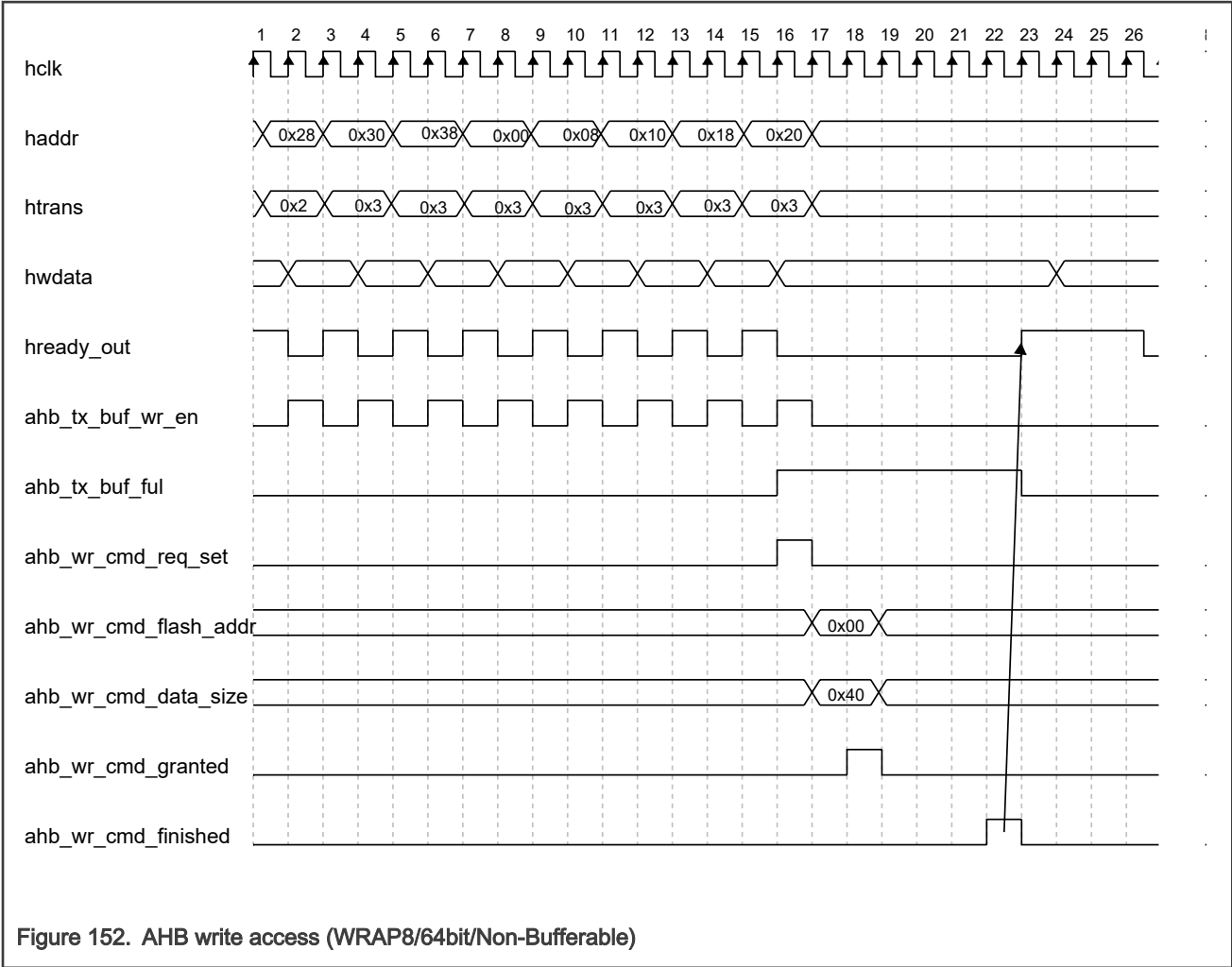


Figure 150. AHB write access (INCR/64bit/Bufferable)

- AHB WRAP8/64bit/bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20:
One AHB write command is triggered with flash start address 0x0 and data size 0x40. See [Figure 151](#).



- AHB WRAP8/64bit/non-bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20: One AHB write command is triggered with flash start address 0x0 and data size 0x40.



NOTE

The wrapper burst is not supported if burst data size (in byte) is larger than AHB TX Buffer size (in byte). For example, if AHB_TX_BUF is 64 bytes (8*64 bits), AHB WRAP16 * 64 bits write burst access is not supported.

39.5.11.2 AHB read access to Flash

For AHB read access to Flash, the FlexSPI checks whether hit AHB TX Buffer/AHB RX Buffer/pending AHB read command according to the burst access type and register setting. If all these miss, the FlexSPI trigger a new AHB read command to fetch data from Flash. The following diagram indicates the hardware operation in response to AHB read access to Flash.

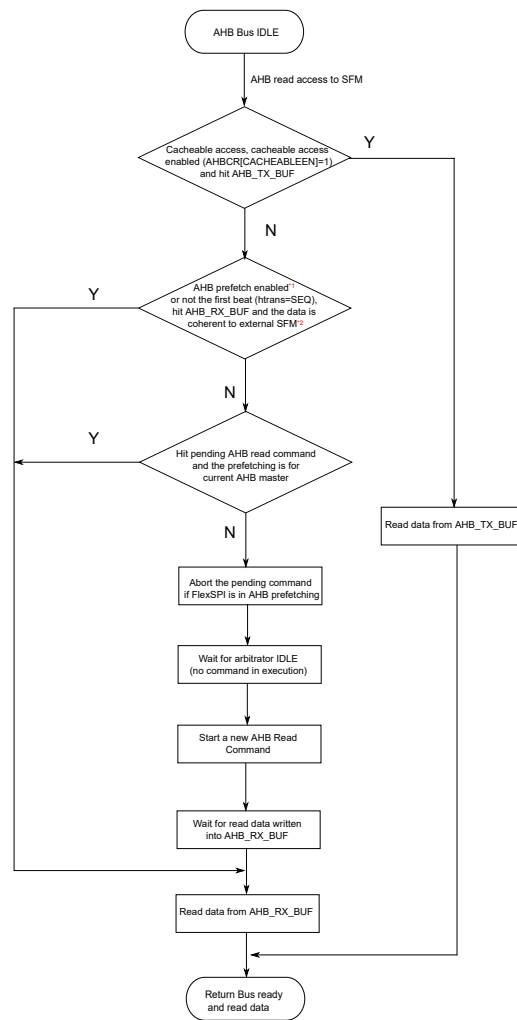


Figure 153. Hardware operation in response to AHB read access to Flash

NOTE

1. AHB prefetch is enabled when both AHBCR[PREFETCHEN]=0x1 and AHB_RXBUFxCR0[PREFETCHEN]=0x1 (x is the AHB RX buffer ID for the current AHB master) or OTFAD is enabled.
2. AHB RX Buffer holds the data read from Flash. This data may become incoherent if the AHB writes to the same flash address.

Below introduces the detailed information about the triggered AHB read command:

- Flash Access Start Address and Data Size:

Determined by AHB address, burst type and burst size. The FlexSPI fetches read data from the start address for the current burst or beat .

Table 422. AHB read command flash start address and data size

Prefetch Enable	Cross flash boundary	Burst Type	Flash start address [26:0]	Data Size
0	Never cross flash boundary because AHB burst never cross 1K Byte boundary.	SINGLE/INCR4/INCR8/INCR16	hbeat_start_address	(hburst_end_address-hbeat_start_address)
		INCR	hbeat_start_address	byte size of current beat For INCR burst with prefetch disabled, each beat is handled same as SINGLE burst.
		WRAP4/WRAP8/WRAP16	hburst_start_address	(hburst_end_address-hburst_start_address)
1	No	INCR/SINGLE/INCR4/INCR8/INCR16	hbeat_start_address	ahb_rx_buf_sz
	No	WRAP4/WRAP8/WRAP16	hburst_start_address	ahb_rx_buf_sz
	Yes	INCR/SINGLE/INCR4/INCR8/INCR16	hbeat_start_address	(flash_top_address-hbeat_start_address)
	Yes	WRAP4/WRAP8/WRAP16	hburst_start_address	(flash_top_address-hburst_start_address)

NOTE

- hbeat_start_address is HADDR input from the AHB master for current beat.
- hburst_start_address (for example 0x00) is the lowest address for current burst, and hburst_end_address (for example 0x10) is the highest address in current burst plus 1. For example, WRAP4 burst with HADDR 0x8, 0xC, 0x0, 0x4.
- ahb_rx_buf_sz is the buffer size in byte of AHB RX Buffer which is used by the current AHB master.
- flash_top_address is the top address of currently accessed flash.

- Flash Chip Select:

The FlexSPI determines the chip select by the settings of flash access start address and flash size.

- Flash Command Sequence Index:

Determined by FLSHxCR2[ARDSEQID].

- Flash Command Sequence Number:

Determined by FLSHxCR2[ARDSEQNUM]. If ARDSEQNUM is not zero, multiple flash access command sequences are triggered every time for an AHB read command. The sequences indexed from ARDSEQID to (ARDSEQID + ARDSEQNUM) in LUT are executed sequentially.

- Flash access mode (Individual/Parallel):

Determined by AHBCR[APAREN].

NOTE

- The FlexSPI determines which ARDSEQNUM/ARDSEQID fields are used as sequence ID for chip selection by Flash device automatically. See [MCR2\[SAMEDEVICEEN\]](#) for more details.
- The FlexSPI determines which AHB RX Buffer is used for current AHB read access by master ID. For more details about the AHB RX Buffer ID and AHB master ID mapping, see [AHB RX Buffer management](#).
- It is not allowed to allocate the size of AHB RX Buffer less than AHB Burst size. The behavior is unknown for this case.

39.5.11.3 AHB RX Buffer management

There are 8 buffers (Buffer 0 - Buffer 7) in AHB RX Buffer, which are transparent to AHB masters. The FlexSPI fetches flash data and return on AHB bus automatically. There is no status register polling needed for AHB read access to Serial Flash Memory.

See the chip-specific section for AHB buffer size. The buffer size is flexibly configurable for each buffer in AHB RX Buffers by register fields AHB RX BUF0CR0[BUFSZ] - AHB RX BUF7CR0[BUFSZ]. The buffer size for Buffer 0 to Buffer 7 could be set 0. If the buffer size is set to 0x0, the related MSTRID field setting (in same AHB RX BUFxCR0 register) is ignored by FlexSPI. Buffer 7 is used for all AHB masters which is not assigned to Buffer 0 - Buffer 6. Buffer 7 size setting field (AHB RX BUF7CR0[BUFSZ]) is ignored by FlexSPI, its buffer size is: AHB RX Buffer total size - sum of (Buffer 0 - Buffer 6 size).

When there is AHB read access to Serial Flash Memory, FlexSPI determines which AHB RX Buffer to use as following:

1. If master ID equal AHB RX BUF0CR0[MSTRID] and AHB RX BUF0CR0[BUFSZ] is not zero, Buffer 0 is used.
2. If master ID equal AHB RX BUF1CR0[MSTRID] and AHB RX BUF1CR0[BUFSZ] is not zero, Buffer 1 is used.
3. If master ID equal AHB RX BUF2CR0[MSTRID] and AHB RX BUF2CR0[BUFSZ] is not zero, Buffer 2 is used.
4. If master ID equal AHB RX BUF3CR0[MSTRID] and AHB RX BUF3CR0[BUFSZ] is not zero, Buffer 3 is used.
5. If master ID equal AHB RX BUF4CR0[MSTRID] and AHB RX BUF4CR0[BUFSZ] is not zero, Buffer 4 is used.
6. If master ID equal AHB RX BUF5CR0[MSTRID] and AHB RX BUF5CR0[BUFSZ] is not zero, Buffer 5 is used.
7. If master ID equal AHB RX BUF6CR0[MSTRID] and AHB RX BUF6CR0[BUFSZ] is not zero, Buffer 6 is used.
8. If all the above cases are not met, Buffer 7 is used.

NOTE

- Software must ensure the size of each buffer is not less than the maximum burst size of AHB Read access from the AHB master using this buffer. Otherwise, the behavior is undefined. The minimal buffer size request depends on the SoC implementation, for example, if the SoC's capability to send out a maximal burst is 64 Bytes, then minimal AHB RX buffer is 64 Bytes.
- Assigning multiple buffers for a single AHB master is not supported.
- When AHB read prefetch is enabled (AHBCR[PREFETCHEN] is set), the prefetch data size is determined by the size of AHB RX Buffer. The FlexSPI fetches data from external Flash device with buffer size if no flash boundary across.
- The priority setting of AHB master (AHB RX BUFxCR0[PRIORITY] is set) is used only for the suspending control of AHB prefetching. See [Command abort and suspend](#).

39.5.11.3.1 Prefetch buffer enhancement

If a given master frequently accesses memory at multiple address areas (non-sequential accesses), then the prefetch buffer can become less effective. For example, if the CPU is executing task A at a given address, then switches to task B at a different address, then when the CPU switches to task B (FlexSPI receives request for an access that does not hit in the prefetch buffer), then the prefetch buffer will be flushed and a new prefetch will start at the task B address. When the CPU switches back to task A (task A addresses now miss in the prefetch buffer, because the buffer holds task B addresses instead), then the prefetch buffer will be flushed again and start prefetching at the task A address.

If the switches between task A and task B happen frequently, then FlexSPI can end up re-reading items it had previously stored in the prefetch buffer as the buffer ends up being flushed often due to the task switches.

To increase the efficiency of the FlexSPI in situations like those described in the example above, the prefetch buffer enhancement feature can be enabled. When prefetch buffer enhancement is enabled (AHBRXBUFnCR0[REGIONEN]=1), then the prefetch buffer space assigned to associated master can be broken up into up to four sub-buffers.

When prefetch buffer enhancement is enabled, the selection of each sub-buffer is determined by address ranges for each sub-buffer (configurable using the AHBREGIONSTARTn and AHBREGIONENDn registers). FlexSPI compares the master ID along with corresponding address range of sub-buffer to select one buffer. Other buffers are not impacted. This way, the AHB master can prefetch the data from multiple address ranges in each of the sub-buffers.

Program sequence:

- Set AHBBUFREGIONSTARTn as system address with ASFM_BASE. ASFM_BASE refers to AHB address map for Serial Flash Memory access.
- Set AHBBUFREGIONENDn as system address with ASFM_BASE
- END must be larger than START
- Set AHBRXBUF (n) CR0[REGIONEN] = 1
- Set AHBRXBUF (n) CR0[MSTRID] with the MSTRID corresponding to the master that will own multiple buffers, for example 4'b0011

NOTE

- This function is only meaningful when prefetch is enabled.
- Only Buffer 0 - 3 support this enhancement when REGIONEN=1, other buffers work like before. Buffer 0 - 3 with REGIONEN=0 works like before.

39.5.12 Command arbitration

There are four Flash access command sources:

1. AHB command (triggered by AHB write access to SFM space)
2. AHB command (triggered by AHB read access to SFM space)
3. IP command (triggered by writing IPCMD[TRG])
4. Suspended command (AHB read prefetch sequence which is suspended)

NOTE

- An AHB bus access never triggers a write command and a read command at the same time.
- AHB prefetch sequence is an AHB command sequence triggered by AHB read access when AHB prefetch is enabled. After all read data is fetched for the current AHB read burst, the FlexSPI prefetches more data to reduce the read latency for the next AHB read access. AHB command for read is never aborted while fetching read data for the current AHB read burst. However, AHB read command can be aborted by any new IP command or AHB command request while prefetching data (not for the current read burst).

When the Arbitrator is idle (STS0[ARBIDLE]=1), the granted priority of these four command sources is shown as follows:

1. AHB command (Read/Write)
2. IP command
3. Suspended command

NOTE

Suspended command is not granted immediately when arbitrator is idle and there is no AHB or IP command request. Arbitrator waits for n AHB clock cycle idle state before resuming the suspended command (n is the register field value in MCR2[RESUMEWAIT]) to avoid AHB prefetch sequence being resumed and suspended frequently.

No new command requests are granted if arbitrator is busy in executing AHB or IP commands (not suspended commands), and there is an AHB or IP command granted error if the grant is timeout.

If a new AHB or IP command request comes while arbitrator is executing AHB read prefetch sequence, AHB read prefetch sequence is aborted (but not immediately). Arbitrator grants the AHB or IP command request after AHB read prefetch sequence is aborted on the FlexSPI interface and saved all internal data pointers.

39.5.12.1 Command abort and suspend

This section describes command abort and suspend mechanism.

1. Command Abort

As mentioned above, AHB read prefetch sequence can be aborted if new AHB or IP command request comes.

2. Command Suspend

When AHB read prefetch sequence is aborted on the FlexSPI interface, the FlexSPI saves this suspended sequence in following cases, and resume this sequence after arbitrator is idle for enough time:

- There is no valid suspended command (Register field AHBSPNDSTS[ACTIVE] is 0x0). This is possible if there is no suspended command or suspended command is resumed.
- The priority of aborted AHB read prefetch sequence is higher than the active suspend sequence.

NOTE

Original suspended sequence is ignored and never resumed by the FlexSPI.

3. Suspended Command

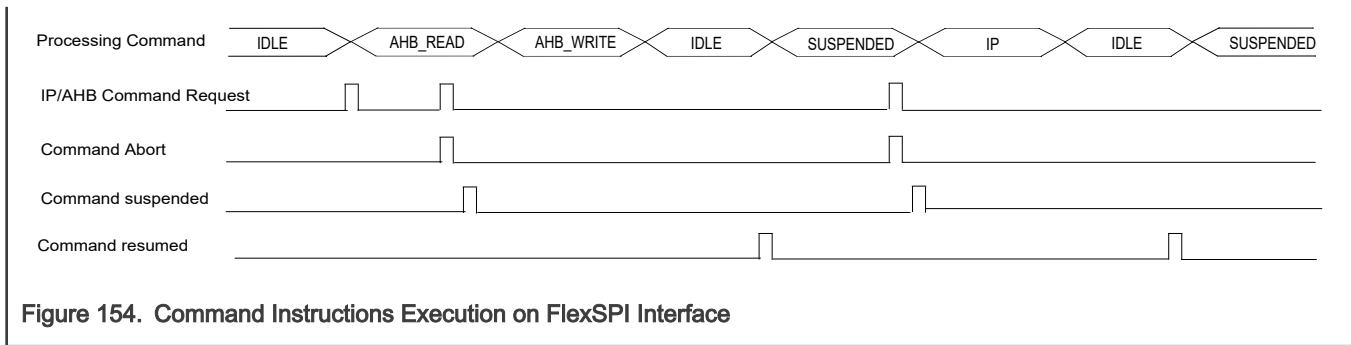
The suspended command (internal status) turns into active when there is any AHB prefetch command aborted and suspended. It turns into inactive in the following cases:

- Suspended command is resumed by arbitrator.
- There is a new AHB read command request that is triggered by AHB master using the same AHB RX Buffer (Buffer ID).

Following is an example indicating the flow of command abort, suspend or resume:

NOTE

- The AHB prefetch sequence is aborted, suspended or resumed twice in the below example. It is aborted by new AHB write command request at the first time and aborted by new IP command request at the second time.
- Suspend sequence is resumed after a wait time of idle state. The suspend and resume activities may be triggered frequently if there is frequent IP or AHB command request and wait time is not set reasonable.



39.5.13 SCLK stop feature

FlexSPI will stop SCLK output toggling when programming data is not ready for programming command sequence, or there is no space (in internal FIFO) to receive data for reading command sequence.

There may be certain devices that do not support SCLK stop during command sequence (chip select is valid). SCLK stopping can be avoided as following:

- For flash reading triggered by IP command
 - Never trigger a read command with data size larger than IP RX FIFO size.
 - Internal async FIFO for flash reading should never be full.

FlexSPI pop data from this async FIFO in 64 bits per AHB clock cycle, and receiving data from FlexSPI interface in serial root clock. The receiving speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual /Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI receives 16 bits per serial root clock cycle. This async FIFO is never full if AHB clock frequency is higher than 1/4 of serial root clock.

- For flash programming triggered by IP command
 - Never trigger a program command with the data size larger than IP TX FIFO size, and fill all programming data into IP TX FIFO before triggering the IP command.
 - Internal async FIFO for flash programming must never be empty.

FlexSPI fetches programming data into this async FIFO in 64 bits per AHB clock cycle, and transmits data to FlexSPI interface in serial root clock. The transmitting speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual /Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI transmits 16 bits per serial root clock cycle. This async FIFO is never empty if AHB clock frequency is higher than 1/4 of serial root clock.

- For flash reading/programming triggered by AHB command
 - Internal async FIFO for flash reading/programming must never be full or empty. The frequency ratio limitation is same as flash reading or flash programming triggered by IP command.

NOTE

- FlexSPI never triggers an AHB read command with data size larger than internal AHB RX Buffer size.
- FlexSPI never triggers an AHB program command with data size larger than internal AHB TX Buffer size.
- All programming data is buffered into AHB TX Buffer before triggering AHB program command in FlexSPI.

39.5.14 FlexSPI output timing

This section describes the output timing in FlexSPI.

39.5.14.1 Output timing between data and SCLK

This section describes the output timing relationship of data (on A_DATA/B_DATA) and SCLK. There are three cases:

- SDR instruction in SDR sequence

SDR sequence is a sequence which contains only SDR instructions. In this case, all data bits last one serial root clock cycle on the FlexSPI interface. The following diagram indicates the relationship of serial root clock, data and SCLK:

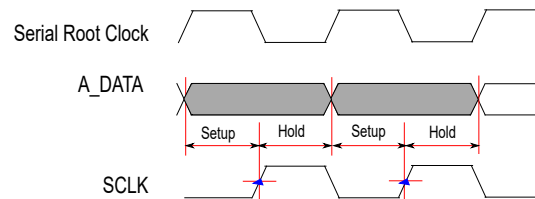


Figure 155. SDR instruction in SDR sequence (Channel A)

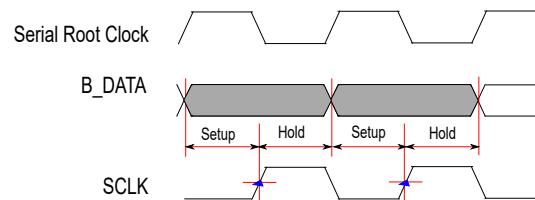


Figure 156. SDR instruction in SDR sequence (Channel B)

- **SDR instruction in DDR sequence**

DDR sequence is a flash access command sequence that contains DDR instructions other than DUMMY_DDR. It may contain SDR instructions optionally. In the case of SDR instruction in DDR sequence, all data bits last two serial root clock cycles on the FlexSPI interface. The following diagram indicates the relationship of serial root clock, data and SCLK:

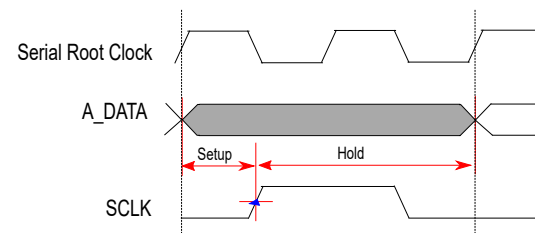


Figure 157. SDR instruction in DDR sequence (Channel A)

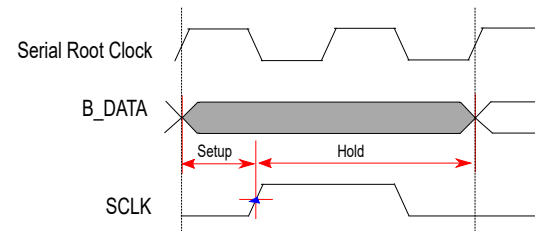


Figure 158. SDR instruction in DDR sequence (Channel B)

- **DDR instruction in DDR sequence**

In the case of DDR instruction in DDR sequence, all data bits last one serial root clock cycle on the FlexSPI interface. The following diagram indicates the relationship of serial root clock, data and SCLK:

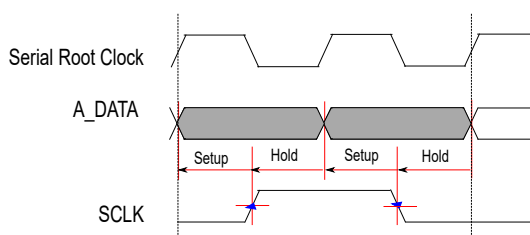


Figure 159. DDR instruction in DDR sequence (Channel A)

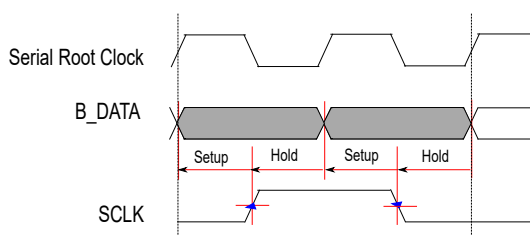


Figure 160. DDR instruction in DDR sequence (Channel B)

39.5.14.2 Output timing between chip select and SCLK

This section describes the output timing relationship of chip select (on A_SS0_B/A_SS1_B/B_SS0_B/B_SS1_B) and SCLK. The timing relationship is a little different for SDR sequence and DDR sequence.

- **Chip select timing in SDR sequence**

For SDR sequence, the delay from chip select assertion and the SCLK rising edge is $(FLSH \times CR1[TCSS] + 0.5)$ cycles of serial root clock. The delay from SCLK falling edge and chip select deassertion is $FLSH \times CR1[TCSSH]$ cycles of serial root clock. The following diagram indicates the timing relationship between chip select and SCLK:

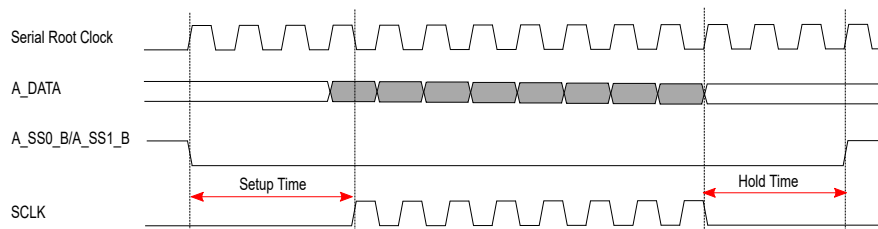


Figure 161. Chip select output timing for SDR sequence (Channel A)

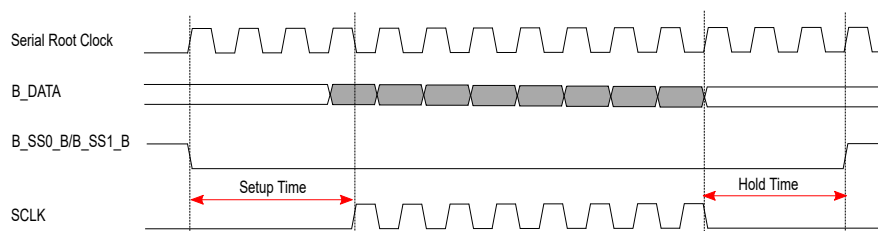


Figure 162. Chip select output timing for SDR sequence (Channel B)

• **Chip select timing in DDR sequence**

For DDR sequence, the delay from chip select assertion and SCLK rise edge is **(TCSS + 0.5)** cycles of serial root clock. The delay from SCLK fall edge and chip select deassertion is **(TCSH + 0.5)** cycles of serial root clock.

NOTE

When AHB RX prefetch is enabled, and prefetch can be aborted, set TCSH to be least 1 to guarantee positive chip select hold time after SCLK falling edge.

The following diagram indicates the timing relationship between chip select and SCLK:

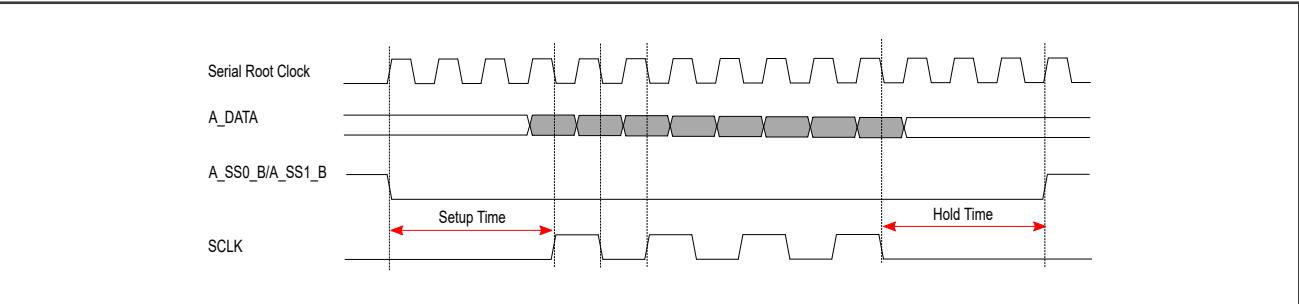


Figure 163. Chip Select Output Timing for DDR Sequence (Channel A)

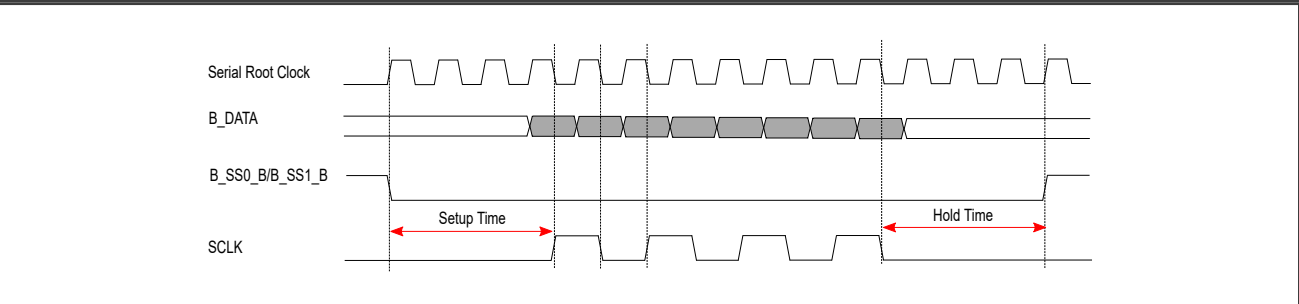


Figure 164. Chip Select Output Timing for DDR Sequence (Channel B)

For certain device (such as FPGA device), there can be a requirement for the minimum chip select negation time (time in between two chip select assertions). The FlexSPI ensures a delay time between chip select valid if register field FLSHxCR1[CSINTERVAL] is set to a non-zero value. The delay time equals to CSINTERVAL*256 cycle of serial root clock for either an SDR or DDR sequence. Set this register field value to zero if there is no limitation for the external device. The following diagram indicates the timing of chip select interval.

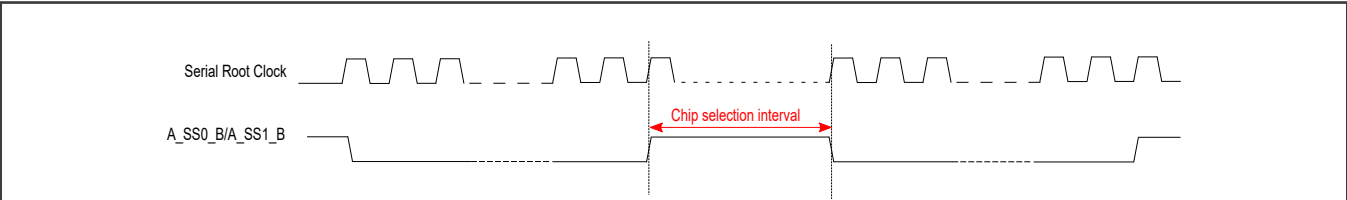


Figure 165. Chip Select Valid Interval (Channel A)

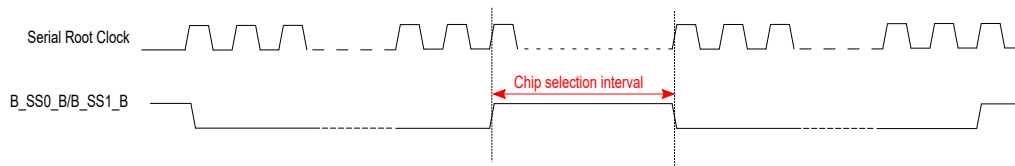


Figure 166. Chip Select Valid Interval (Channel B)

39.5.15 FlexSPI input timing

This section describes the input timing of FlexSPI.

39.5.15.1 Receiving block diagram

This section describes the receiving block in FlexSPI.

FlexSPI determines SDR mode or DDR mode by decoding instruction code, and determines whether both the rising and falling edges used or not for flash data sampling. FlexSPI samples the data line for Learn instruction and Read instruction.

There are four clock sources for the sampling clock, and the source selection is configured by the register field MCR0[RXCLKSRC]. For more details about DLL configuration, see [DLL configuration for sampling](#)

- Internal dummy read strobe and loopbacked internally
- Internal dummy read strobe and loopbacked from DQS pad
- SCK output and loopbacked from SCK pad
- Flash provided read strobe

The following diagram indicates the sampling block in FlexSPI:

FlexSPI uses both rising and falling edges to sample data from flash_A and flash_B. The signals ipp_ind_io_fa, ipp_ind_io_fb are generated by 2 mux, which have 4 clock source inputs. See [RX clock source features](#) for more details. ipp_ind_io_fa_int and ipp_ind_io_fb_int are reversed from ipp_ind_io_fa and ipp_ind_io_fb. Both the sample flip-flops are used when FlexSPI is in DDR mode. Only falling edge sample flip-flop is used in SDR mode.

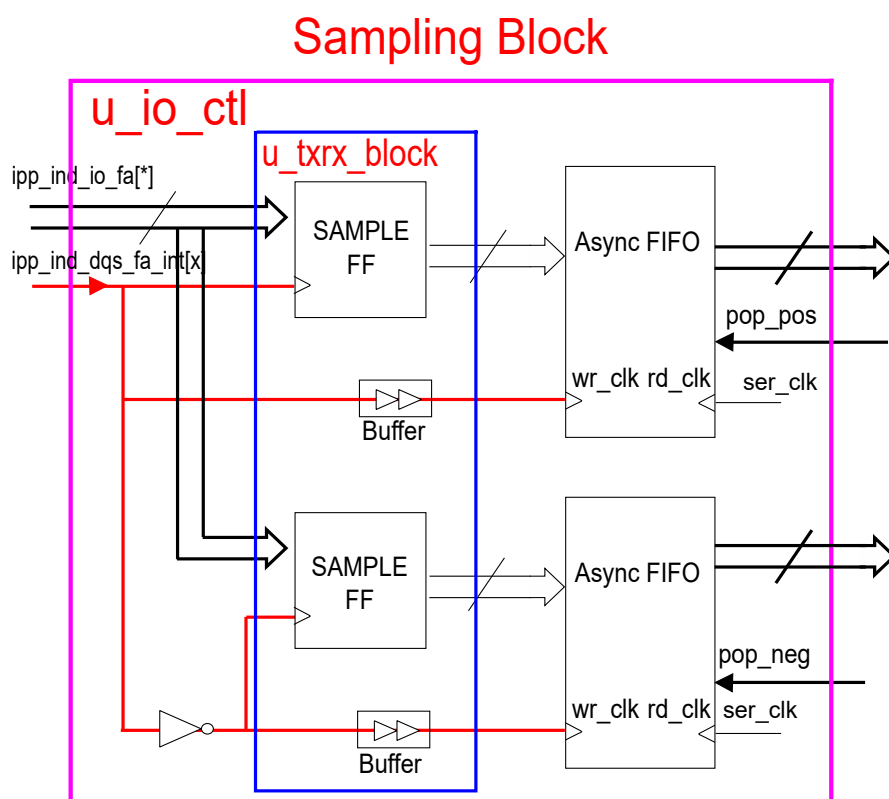
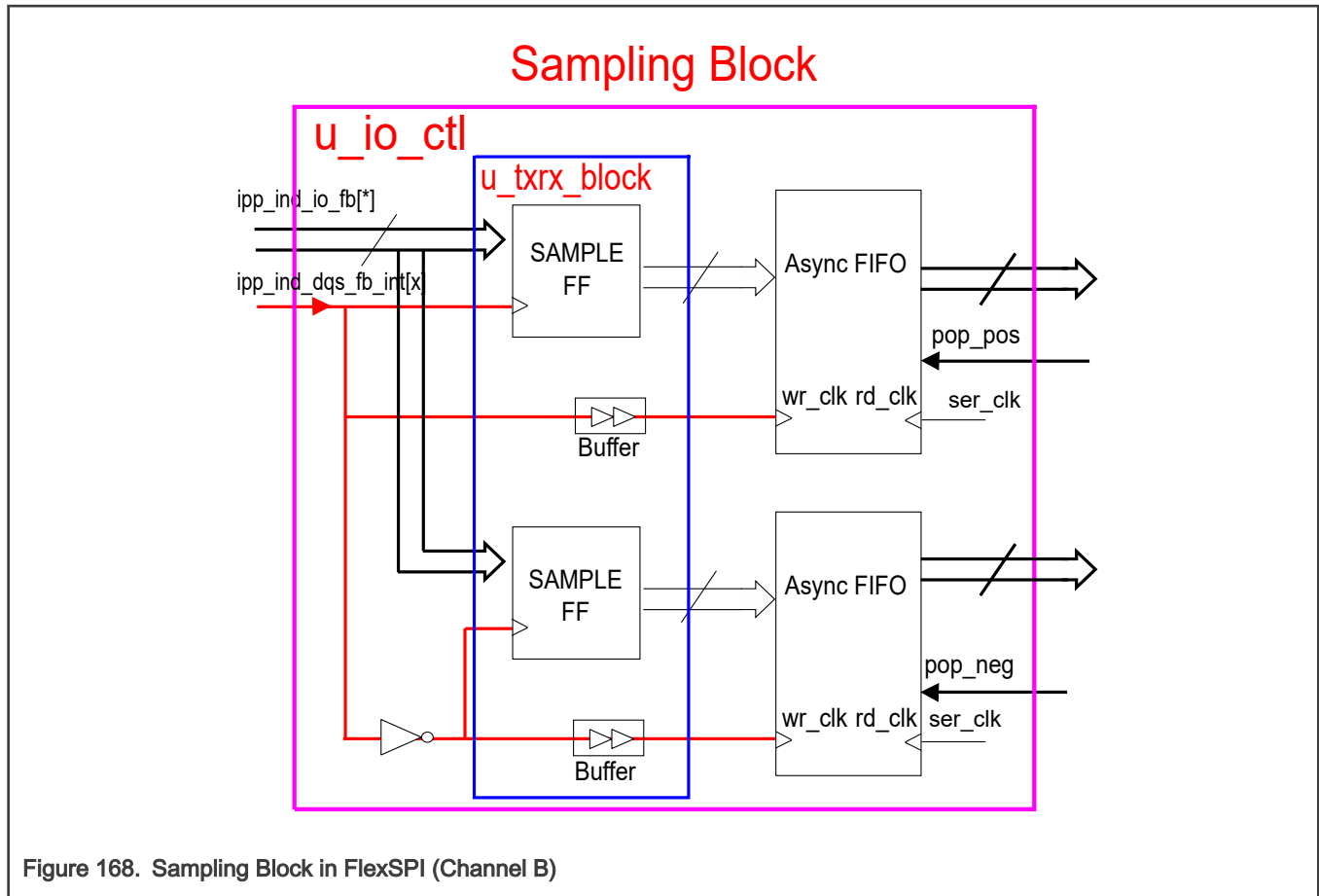


Figure 167. Sampling Block in FlexSPI (Channel A)



The following is the RX block diagram for data learning feature.

There are 16 clock phases for sampling data. The clock phases are delayed from internal delay chain. The delay gap is determined by register MISCCR2[LEARNPHASEGAP].

See [Data learning feature](#) for more details.

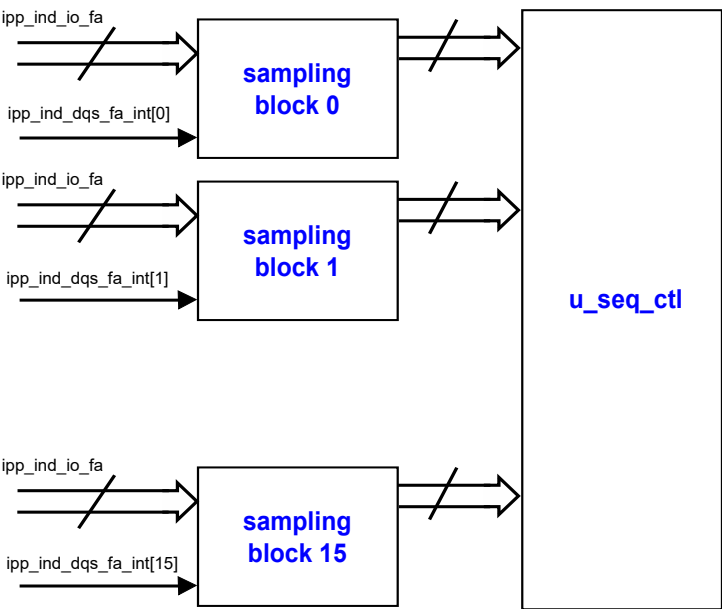


Figure 169. Sampling Blocks for Data Learning Feature (Channel A)

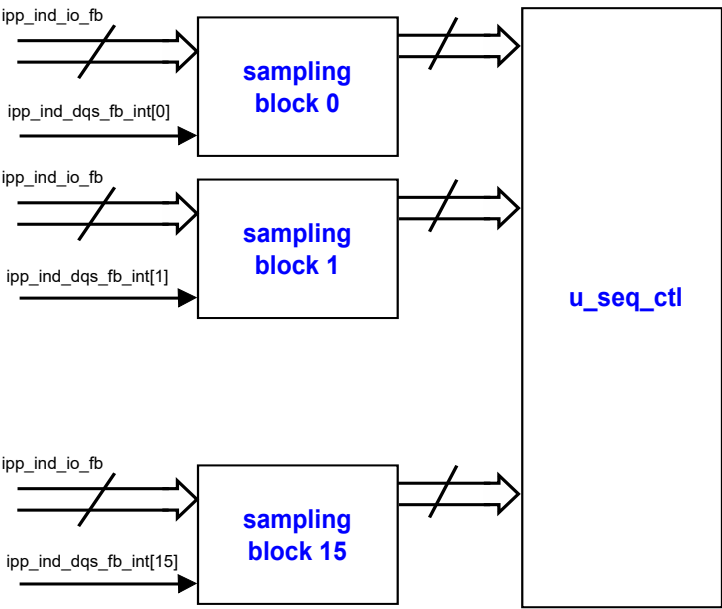
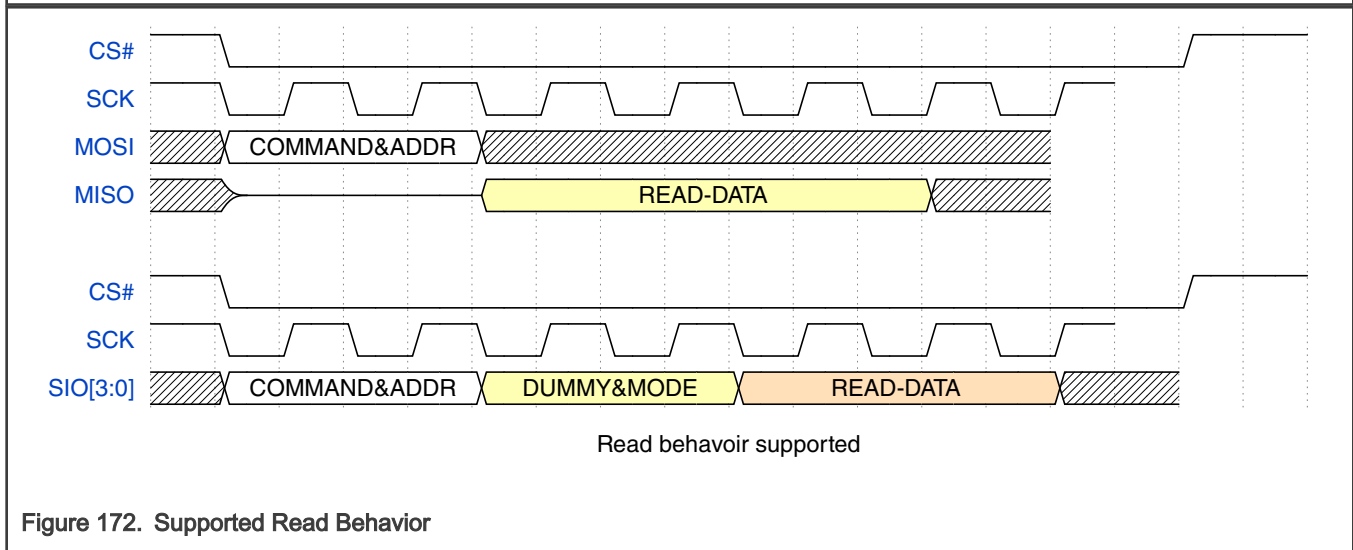
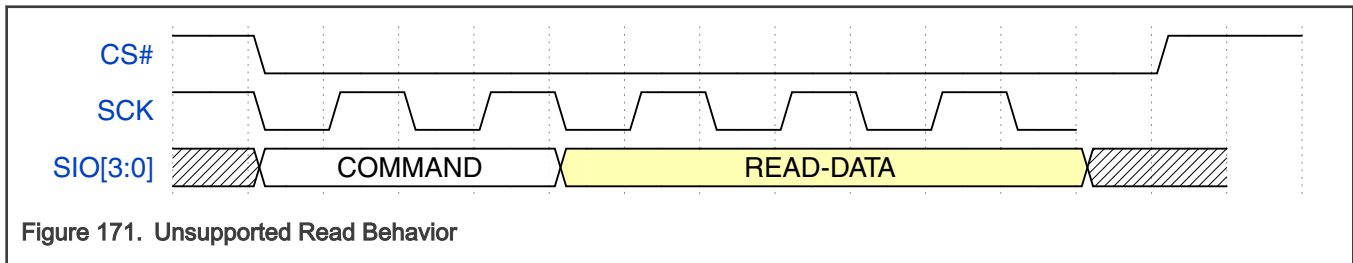


Figure 170. Sampling Blocks for Data Learning Feature (Channel B)

NOTE

- FlexSPI does not support read with 0 turn around cycle for direction switching on I/O pins. This read often happens on QPI configuration register read. For example, in below waveform, "QuadSPI configuration register read", IO[3:0] is output for FlexSPI in CMD phase, but is input for FlexSPI in DATA phase. And there is no time left for FlexSPI to switch the direction. This kind of read is not supported. Moreover, FlexSPI and Flash drive the I/O pins at the same time, which may cause a problem. So, SPI mode or Read with dummy cycles is required to avoid described behavior on I/O pins.
- Cycle number is not accurate in the example below, for accurate cycle number, see device's spec.



39.5.15.2 RX clock source features

This section describes the features of each RX clock source.

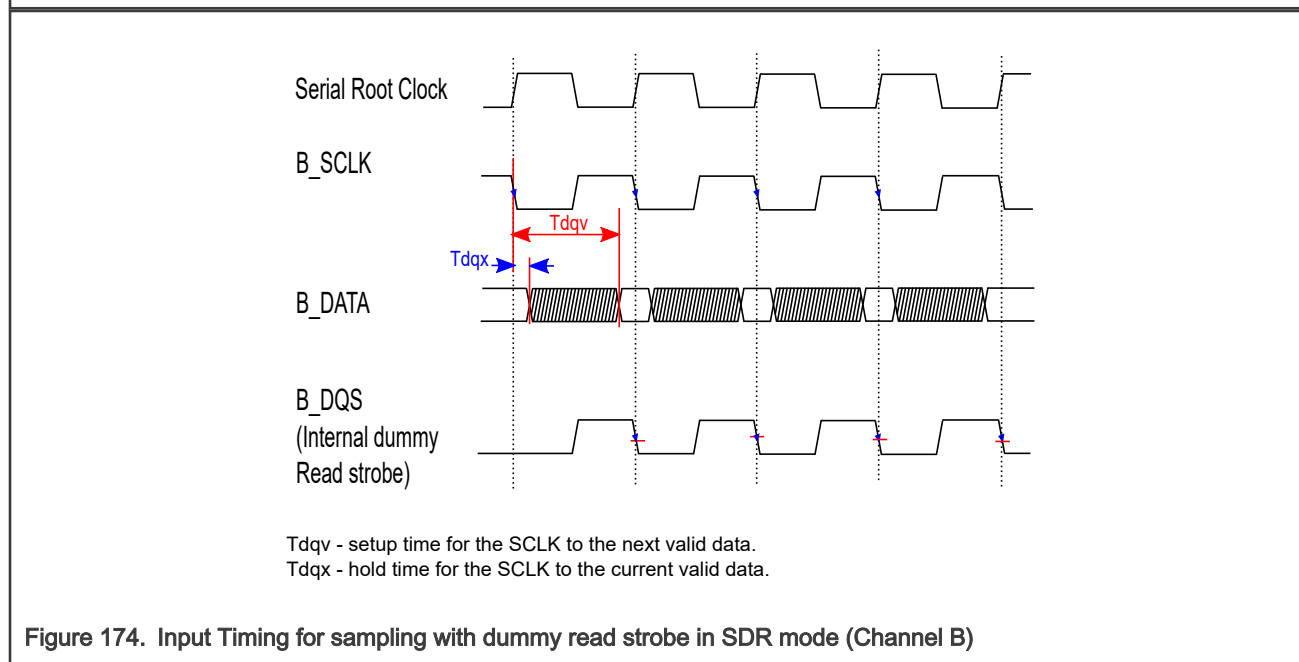
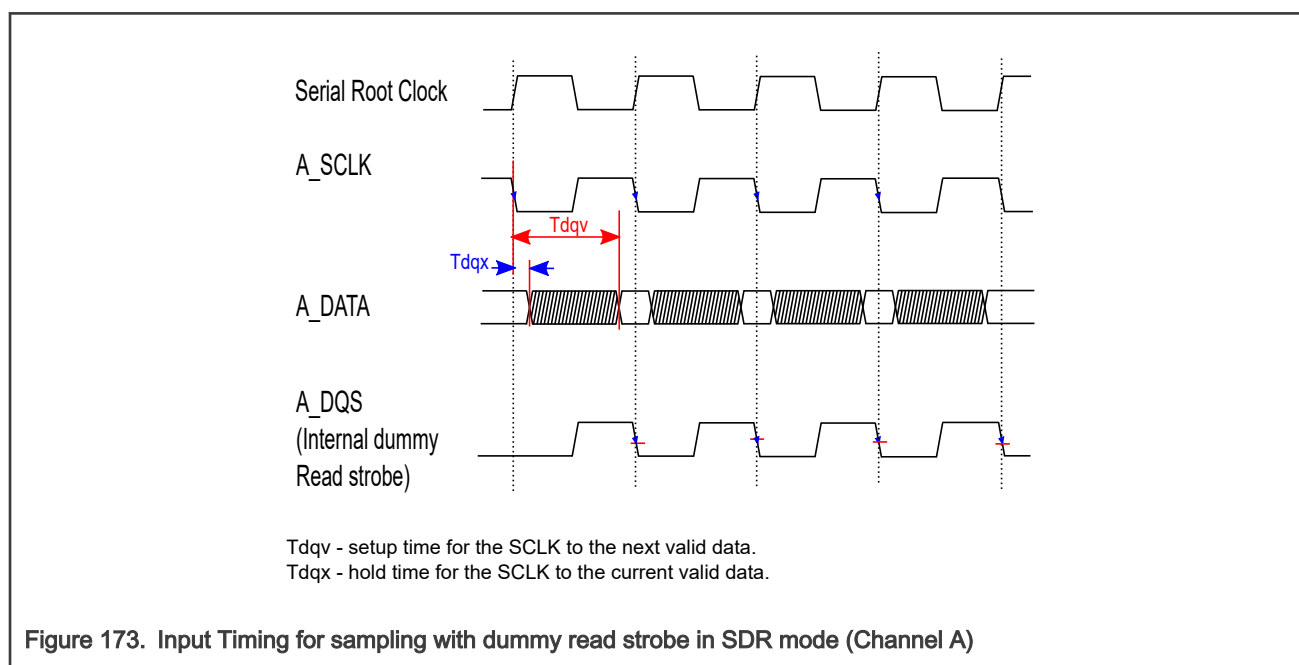
- Internal dummy read strobe and internal loopback (MCR0[RXCLKSRC]==0)**
Supporting legacy device with zero device output hold time.
Saving one pad (DQS pad).
Supporting low frequency clock for boot up usage.
- Internal dummy read strobe and loopback from DQS pad (MCR0[RXCLKSRC]==1)**
Supporting higher frequency than mode "MCR0[RXCLKSRC]==0".
Supporting device does not provide read strobe.
- Flash provided read strobe (MCR0[RXCLKSRC]==3)**
Supporting the highest frequency.
External device provides read strobe.

39.5.15.3 Input timing for sampling with dummy read strobe

This section describes the input timing when sampling with internal dummy read strobe ($\text{MCR0[RXCLKSRC]} = 0x0$ or $0x1$). The timing is very similar for sampling with dummy read strobe loopback internally and loopback from pad. But it could achieve higher read frequency by sampling with dummy read strobe loopback from DQS pad because it will compensate the delay of SCLK output path and Data pin input path. The input timing is different for SDR mode and DDR mode.

- **Input timing for sampling with dummy read strobe in SDR mode**

For SDR Read/Learn instruction, FlexSPI samples input data pins with the falling edge of dummy read strobe. Following diagram indicates the input timing for sampling with dummy read strobe in SDR mode:



- **Input timing for sampling with dummy read strobe in DDR mode**

For DDR Read or Learn instruction, the FlexSPI samples input data pins with both rising and falling edges of dummy read strobe. The following diagram indicates the input timing for sampling with dummy read strobe in DDR mode

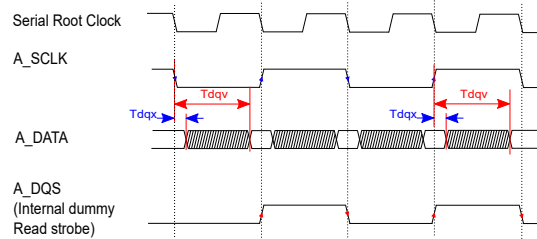


Figure 175. Input timing for sampling with dummy read strobe in DDR mode (Channel A)

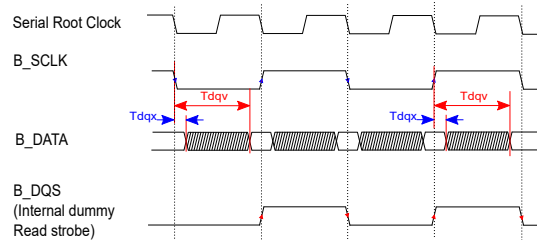


Figure 176. Input timing for sampling with dummy read strobe in DDR mode (Channel B)

39.5.15.4 Input timing for sampling with SCK output loopbacked from SCK pad

This section describes the input timing when sampling with SCK output loopbacked from SCK pad. The input timing is similar as sampling with dummy read strobe. SCK output toggles for all types of instructions, but internal dummy strobe just toggles for Read and Learn instruction. So FlexSPI gets more data bits when sampling with SCK output. FlexSPI ignores the redundant data bits sampled automatically.

39.5.15.5 Input timing for sampling with flash provided read strobe

This section describes the input timing when sampling with flash provided read strobe (MCR0[RXCLKSRC] = 0x3). The input timing is different for SDR mode and DDR mode.

NOTE

There are no known devices that provide read strobe and support SDR mode operation.

• Flash provide read strobe with SCLK

For certain flash devices, it provides both read data and read strobes with SCLK. Then the read strobe edge is aligned with read data change. FlexSPI controller delays read strobe by half cycle in serial root clock (with DLL), and then sample read data with delayed strobe. See [DLL configuration for sampling](#) for more details. The following diagrams indicate the input timing for sampling with flash read strobe in SDR mode and DDR mode:

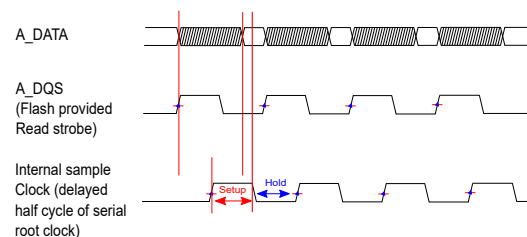


Figure 177. Input timing 2 for Flash provided read strobe in SDR mode (Channel A)

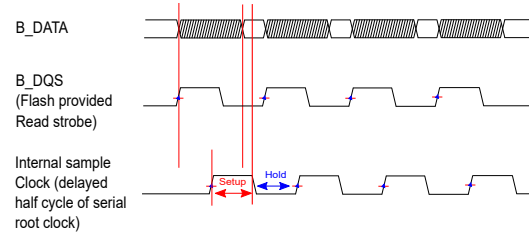


Figure 178. Input timing 2 for Flash provided read strobe in SDR mode (Channel B)

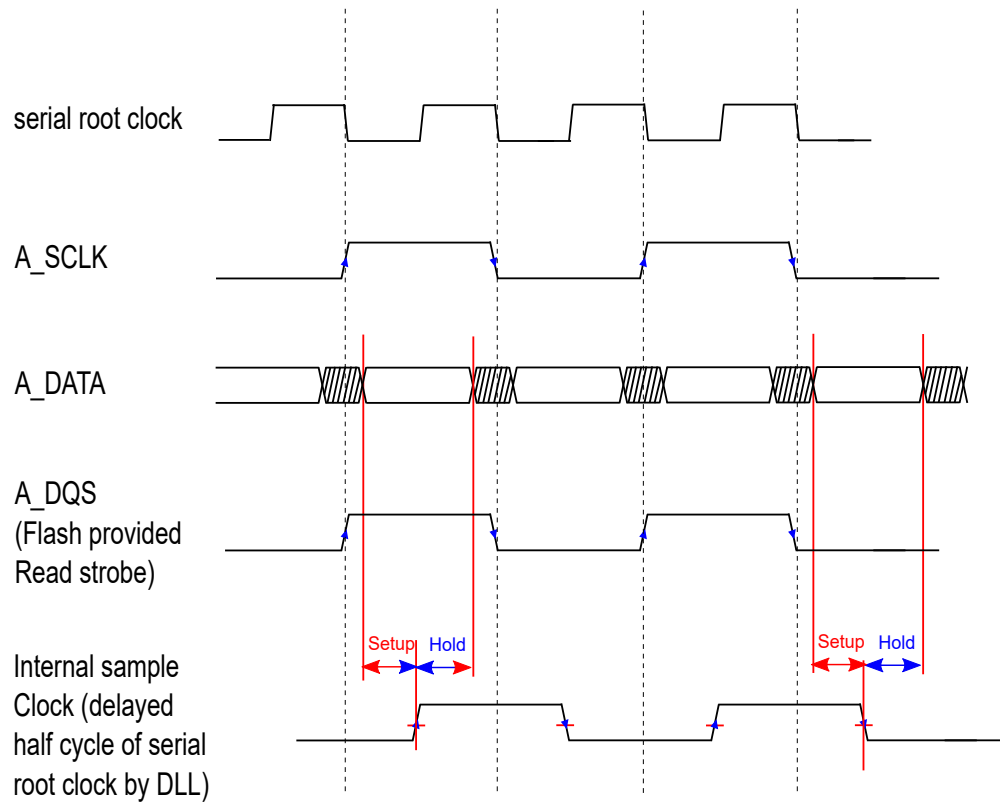
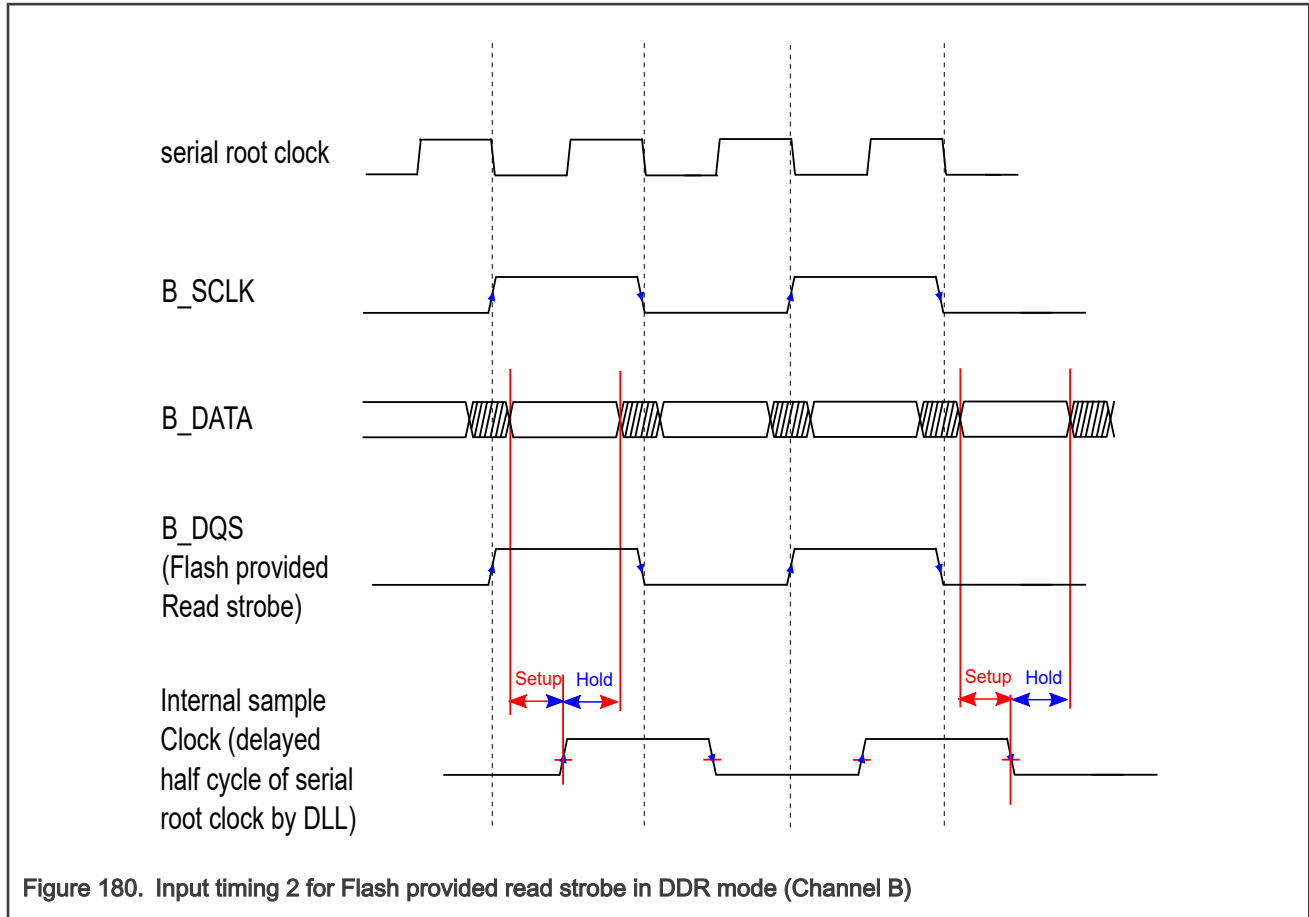


Figure 179. Input timing 2 for Flash provided read strobe in DDR mode (Channel A)



39.5.15.6 DLL configuration for sampling

For four sampling clock sources outlined in previous sections, the input timing is different by setting register DLLxCR differently according to the sampling clock source mode. DLL is a delay line chain, which can be set to a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to the reference clock.

- In the following cases, DLLxCR must be set to 0x00000100 (1 fixed delay cell in DLL delay chain) :
 - Sampling data with dummy read strobe loopbacked internally (MCR0[RXCLKSRC]=0x0)
 - Sampling data with dummy read strobe loopbacked from DQS pad (MCR0[RXCLKSRC]=0x1)
- When data is sampled with Flash provided read strobe (MCR0[RXCLKSRC]=0x3) and Flash provides read strobe with SCLK, DLL must be set as follows to lock on half cycle of the reference clock (serial root clock)
 - SLVDLYTARGET = 0
 - DLEN = 0x1
 - OVRDEN = 0x0
 - Other fields in DLLxCR must be kept as reset value (all zero)

NOTE

If serial root clock is lower than 100 MHz, DLL cannot lock on half cycle of serial root clock because the delay cell number is limited in delay chain. Then, DLL must be configured as follows instead:

- OVRDEN = 0x1
- OVRDVAL = N

Each delay cell in DLL is about 75 ps - 225 ps. The delay of DLL delay chain is (N * Delay_cell_delay). N is set based on the maximum DDR frequency ($N = \text{equation}$). This value is calculated based on the parameter MAX_DDR_FREQ. N = 18. Note that this is a recommended value, which may need to be adjusted in real application if a failure occurs.

- Other fields in DLLxCR must be kept as reset value (all zero).

39.5.16 Data learning feature

FlexSPI controller generates 16 clock phases (Phase 0 - Phase 15) by Delay Cell Line (DCL) with the selected sample clock. Clock phase 0 is actually the selected sample clock (DQS_IN) without any delay cell. There are 16 sampling blocks implemented for Port A.

During Learn instruction, FlexSPI compares the sampled data bits with internal data learn pattern (DLPR register), and determines the correct sampling clock phase. The phase selection is automatically updated after Learn instructions and applied to the subsequent Read instructions or sequences.

When the data learning feature is disabled (MCR0[LEARNEN] = 0x0), FlexSPI always uses clock phase 0 to sample FlexSPI data lines. And the attempt to execute a LEARN instruction results in an error flag (INTR[IPCMDERR] or INTR[AHBCMDERR]).

Data learning feature is supported in both SDR mode and DDR mode in FlexSPI. Data learning feature is not supported if sampling clock source is flash provided read strobe (MCR0[RXCLKSRC] = 0x3).

Data learning feature can achieve high read frequency, but the highest frequency is still limited by Flash input timing (Flash needs to receive command code and Flash address bits).

If data learning feature is enabled, FlexSPI uses clock phase 0 after reset and before executing a Learn instruction. The clock phase selection is updated after the Learn instruction is executed by FlexSPI. The selected sample clock phase can be polled by register field STS0[DATALEARNPHASEA] and STS0[DATALEARNPHASEB]. When data learning fails, the interrupt bit (INTR[DATALEARNFAIL]) is set, and the previous clock phase selection is kept. Internal clock phase selection can be reset to clock phase 0 by writing 0x1 to MCR2[CLRLEARNPHASE].

39.5.16.1 Data learning with Flash providing preamble bit

Certain flash devices support driving with preamble bits (which may be also called DLP - Data Learning Pattern) before driving read data in each read command sequence. The DLP is programmable by configuration register in flash device.

For specified read command sequence, flash returns preamble bit after dummy cycles and before returning read data.

For these flash devices, the operation flow with data learning is shown as follows:

1. Set data learning pattern in DLPR register.
2. Set same data learning pattern to the flash device by triggering IP command.
3. Enable data learning feature in the flash device by triggering IP command.
4. Configure LUT sequence with valid read command sequence which contains Learn instruction.
5. Trigger Flash read command by AHB or IP command as normal.

NOTE

The first four steps are not needed for every flash read command, they are executed only once.

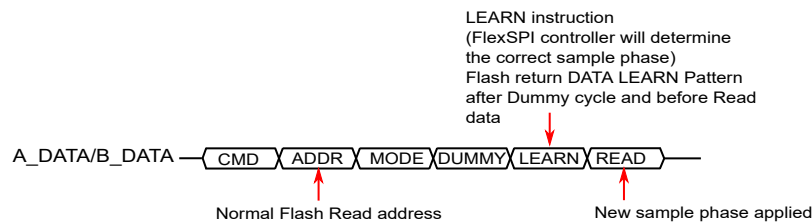


Figure 181. Data learning flow with flash providing preamble bit

39.5.16.2 Data learning with Flash not providing preamble bit

For flash devices not providing preamble bit, there is no way to return data learning pattern automatically by these flash devices for each read command sequence. The following sequence is data learning with flash not providing preamble bit:

1. Set data learning pattern in DLPR register.
2. Reserve a certain flash memory area and program data (according to data learning pattern and flash access mode) to this area by IP command.
3. Configure LUT sequence with read sequence (using LEARN instruction instead of READ instruction), and trigger Read sequence to the reserved flash area by IP command.
4. Trigger Flash read command by AHB or IP command as normal. No LEARN instruction is needed in this Read sequence.

NOTE

- The first three steps are not needed for every flash read command, they are executed only once.
- Step 4 should be executed with a certain interval to make sure internal sampling clock phase is adjusted in time.

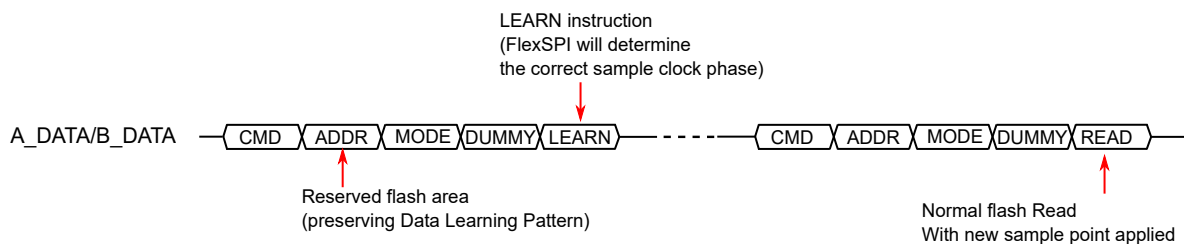


Figure 182. Data learning flow with Flash not providing preamble bit

NOTE

- For Flash not providing preamble bit, there is software overhead because it is needed to read the reserved flash area at interval.

The pre-programming data is determined by:

- Data Learning Pattern (DLPR register setting)
- Data Learning Pattern bit length
- Individual/Parallel mode for read command
- Octal/Quad/Dual/Single mode for read command

Below is an example for pre-programming data when DLPR value is 0x43, 8 bits, and flash is read in Quad mode and Parallel mode.

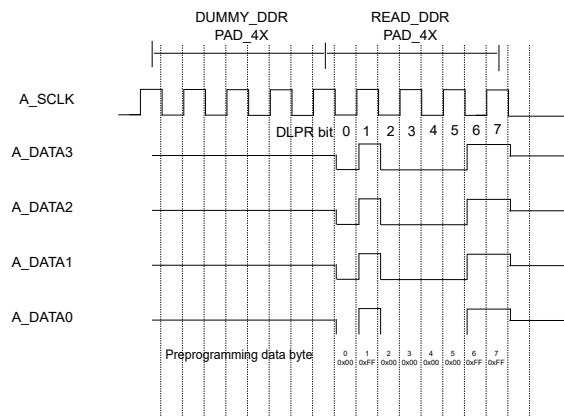


Figure 183. Preprogramming data for data learning (Channel A)

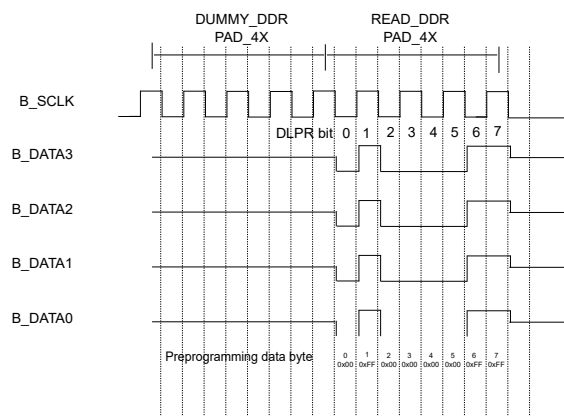


Figure 184. Preprogramming data for data learning (Channel B)

39.5.17 XIP enhanced mode

FlexSPI always supports Execute-In-Place (XIP), no matter external devices provide XIP enhanced mode or not. XIP is supported by putting program codes on an external device, and then reading or executing these program codes on the external device directly by AHB read access to SFM space. There is no configuration or status polling needed during AHB read access to the memory of the external device, and AHB RX Buffer is fully transparent to software.

Certain devices provide XIP enhanced mode to improve code execution. In this mode, the command code is only sent in the first read instruction. It saves many cycles for Command Instructions and improves code execution. This XIP enhanced mode is entered or exit by a special sequence that is specified by external devices. See external device data sheet for more details.

Normally, the XIP enhanced mode is entered by following sequence:

1. Enable XIP enhanced mode in external flash device by IP command.
2. Send the first read sequence to external flash device with correct mode bits. Command code is included in this read sequence.
3. Send the subsequent read sequences to external flash device with correct mode bits. Command code is not included in these read sequences. But mode bits must be sent according to the flash specified. Otherwise the flash exits XIP enhanced mode.

The instruction JMP_ON_CS in FlexSPI must be used to support XIP enhanced mode in external devices. This instruction is only allowed in AHB read command. Otherwise, there is IPCMDERR or AHBCDMERR error interrupt generated when this interrupt is enabled. JMP_ON_CS must never be used when external devices do not support XIP enhanced mode. To support XIP enhanced mode, the first instruction in the read sequence must be command instruction and the last valid instruction must be JMP_ON_CS (with operand 0x1).

For the first AHB read command triggered, FlexSPI executes the instructions from the instruction pointer 0 in the sequence (which is command instruction). After this sequence is executed, FlexSPI saves operand in JMP_ON_CS instruction as start pointer for the next command to current device internally. For the following AHB read command triggered, FlexSPI executes from the instruction pointer 0x1 and command instruction is bypassed.

After XIP enhanced mode is started, it is possible to access flash by other commands like any program commands, but you have to use a special sequence to exit enhanced XIP mode before you can run a different command. The following diagram indicates XIP operation with Flash XIP enhanced mode:

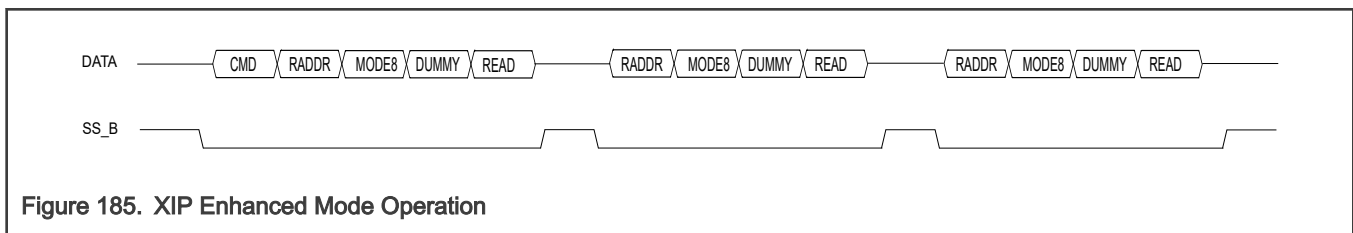


Figure 185. XIP Enhanced Mode Operation

39.6 Initialization

The FlexSPI controller initialization sequence is shown as follows:

1. Enable controller clocks (AHB clock, IP bus clock, or serial root clock) at system level.
2. Set MCR0[MDIS] to 0x1 (make sure that the FlexSPI controller is configured in module stop mode).
3. Configure module control registers: MCR0, MCR1, MCR2. (do not change MCR0[MDIS]).
4. Configure AHB bus control register (AHBCR) and AHB RX Buffer control register (AHBRXBUFxCRO) optionally when AHB command is used.
5. Configure Flash control registers (FLSHxCRO, FLSHxCR1, or FLSHxCR2) according to external device type.
6. Configure DLL control register (DLLxCR) according to sample clock source selection.
7. Set MCR0[MDIS] to 0x0 (exit module stop mode).
8. Configure LUT as needed (for AHB command or IP command).
9. Reset controller optionally (by set MCR0[SWRESET] to 0x1).

External device needs to be configured by IP command normally after initialization. For example, the device configuration is done by WRITE STATUS command for most serial NOR Flash.

39.7 Application information

This section describes applications supported by the FlexSPI module.

39.7.1 Application on serial NOR Flash device

This section provides the example LUT instruction sequences for serial NOR flash device (Cypress Flash S25FS128S).

39.7.1.1 Write enable command

The following table shows WRITE ENABLE command sequence.

Table 423. WRITE ENABLE command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0	0x06	command name:WREN
1-7	STOP (0x00)	0x0	0x00	

39.7.1.2 Write registers command

The following table shows Write Registers command sequence.

Table 424. Write Registers command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x01	command name:WRR
1	WRITE_SDR	0x0 (Single)	0x1 or 0x2	Data size is 1 or 2byte. Byte 0 write data for Status Register 1; Byte 1 write data for Configuration Register 1 This value could be overridden by IPCR1[IDATSZ].
2-7	STOP (0x00)	0x0	0x00	

39.7.1.3 Page Program command

The following table shows Page Program command sequence.

Table 425. PAGE PROGRAM command (Cypress serial NOR flash)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x02 or 0x12	command name: PP or 4PP PP is 3-Byte address mode. 4PP is 4-Byte address mode
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	WRITE_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default programming data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3-7	STOP (0x00)	0x0	0x00	

The following table shows Page Program command sequence (QPI mode).

Table 426. PAGE PROGRAM (QPI mode) command (Cypress serial NOR flash)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0x02 or 0x12	command name: PP or 4PP PP is 3-Byte address mode. 4PP is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	WRITE_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default programming data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3-7	STOP (0x00)	0x0	0x00	

39.7.1.4 Read Status 1 command

The following table shows READ STATUS 1 command sequence.

Table 427. READ STATUS 1 command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x05	command name:RDSR1
1	READ_SDR	0x0 (Single)	0x1	1 Byte for Status register 1
2-7	STOP (0x00)	0x0	0x00	

39.7.1.5 Read command

The following table shows READ command sequence.

Table 428. READ command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x03 or 0x13	command name:READ or 4READ READ is 3-Byte address mode. 4READ is 4-Byte address mode
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	READ_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3-7	STOP (0x00)	0x0	0x00	

39.7.1.6 Fast Read command

The following table shows Fast Read command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8).

Table 429. FAST_READ command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x0B or 0x0C	command name:FAST_READ or 4FAST_READ FAST_READ is 3-Byte address mode. 4FAST_READ is 4-Byte address mode
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	DUMMY_SDR	0x0 (Single)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
3	READ_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
4-7	STOP (0x00)	0x0	0x00	

39.7.1.7 Dual IO Fast Read command

The following table shows Dual IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8, Continuous Read mode).

Table 430. Dual IO FAST_READ command (Continuous Read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xBB or 0xBC	command name:DIOR or 4DIOR DIOR is 3-Byte address mode. 4DIOR is 4-Byte address mode
1	ADDR_SDR	0x1 (Dual)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x1 (Dual)	0xAx	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x1 (Dual)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x1 (Dual)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.

Table continues on the next page...

Table 430. Dual IO FAST_READ command (Continuous Read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
5	JMP_ON_CS	0x0 (or Don't care)	0x01	The CMD instruction will be bypassed after the first read access.
6-7	STOP (0x00)	0x0	0x00	

The following table shows Dual IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8, Non-continuous Read mode).

Table 431. Dual IO FAST_READ command (Non-continuous Read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xBB or 0xBC	command name:DIOR or 4DIOR DIOR is 3-Byte address mode. 4DIOR is 4-Byte address mode
1	ADDR_SDR	0x1 (Dual)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x1 (Dual)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_SDR	0x1 (Dual)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x1 (Dual)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5-7	STOP (0x00)	0x0	0x00	

39.7.1.8 Quad IO Fast Read command

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Non-continuous read mode).

Table 432. Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xEB or 0xEC	command name:QIOR or 4QIOR QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.

Table continues on the next page...

Table 432. Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
2	MODE8_SDR	0x2 (Quad)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5-7	STOP (0x00)	0x0	0x00	

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Continuous read mode).

Table 433. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xEB or 0xEC	command name:QIOR or 4QIOR QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5	JMP_ON_CS	0x0 (or Don't care)	0x01	The CMD instruction will be bypassed after the first read access.
6-7	STOP (0x00)	0x0	0x00	

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=1, CR2V[3:0]=0x8, QPI mode, Continuous read mode).

Table 434. Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0xEB or 0xEC	command name:QIOR or 4QIOR

Table continues on the next page...

Table 434. Quad IO FAST_READ command (QPI mode, Continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5	JMP_ON_CS	0x0 (or Don't care)	0x01	The CMD instruction will be bypassed after the first read access.
6-7	STOP (0x00)	0x0	0x00	

39.7.1.9 DDR Quad IO Fast Read command

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Non-continuous read mode).

Table 435. DDR Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if

Table continues on the next page...

Table 435. DDR Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
6-7	STOP (0x00)	0x0	0x00	

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Continuous read mode).

Table 436. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
6	JMP_ON_CS	0x0 (or Don't care)	0x01	The CMD instruction will be bypassed after the first read access.
7	STOP (0x00)	0x0	0x00	

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=1, CR2V[3:0]=0x8, QPI mode, Continuous read mode).

Table 437. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode

Table continues on the next page...

Table 437. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern) is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
6	JMP_ON_CS	0x0 (or Don't care)	0x01	The CMD instruction will be bypassed after the first read access.
7	STOP (0x00)	0x0	0x00	

39.7.2 Application on HyperBus device

This section provides the example LUT instruction sequences for HyperBus device (Cypress RPC flash/HyperRam/HyperFlash).

39.7.2.1 HyperFlash

This section provides the example sequences for HyperFlash devices (Cypress S26KS series).

The following table shows read status command sequence.

Table 438. Read status command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0x70)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0070
	7	CMD_DDR	0x3 (Octal)	0x70	

Table continues on the next page...

Table 438. Read status command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1 (Read - Addr=xxx, Data= Status register data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWD S_DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	0x4	4 Byte read
	5-7	STOP (0x00)	0x0	0x00	

The following table shows read (memory) command sequence.

Table 439. Read (memory) command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Read - Addr=xxx, Data= memory data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWD S_DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
	5-7	STOP (0x00)	0x0	0x00	

The following table shows word program command sequence.

Table 440. Word program command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	
1 (Write - Addr=0x2AA, Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	
2 (Write - Addr=0x555, Data=0xA0)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x55	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00A0

Table continues on the next page...

Table 440. Word program command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	7	CMD_DDR	0x3 (Octal)	0xA0	
3 (Word Program)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte written data
	4-7	STOP (0x0)	0x0	0x00	

The following table shows Write-to-Buffer and Program-Buffer-to-Flash command sequence.

Table 441. Write-to-Buffer and Program-Buffer-to-Flash command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	
1 (Write - Addr=0x2AA, Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	

Table continues on the next page...

Table 441. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	
2 (Write - Addr=SA, Data=0x25)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0025
	4	CMD_DDR	0x3 (Octal)	0x25	
2 (Write - Addr=SA, Data=WC)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	WC	Write Data: WC
	4	CMD_DDR	0x3 (Octal)		WC is word count
3 - N (Write - Addr=WBL, Data=PD) N is the word count + 2	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: WBL (24 bit) WBL is write buffer location. Set IPCR0[SFAR]=WBL

Table continues on the next page...

Table 441. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte write data
	4-7	STOP (0x0)	0x0	0x00	
N+1 (Write - Addr=SA, Data=29) Program Buffer to Flash	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x29
	4	CMD_DDR	0x3 (Octal)	0x29	
	5-7	STOP (0x0)	0x0	0x00	

39.7.2.2 HyperRAM

This section provides the example sequences for HyperRAM (Cypress S27KL series).

Read (memory) command sequence is same as HyperFlash. The following table shows Write (memory) command sequence.

Table 442. Write (memory) command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
3	DUMMY_RWDS_D DR	0x3 (Octal)	0x0B	In case of latency count=11
4	WRITE_DDR	0x3 (Octal)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ]

Table continues on the next page...

Table 442. Write (memory) command (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				is zero. This value is ignored for AHB command.
5-7	STOP (0x00)	0x0	0x00	

39.7.3 Application on Serial NAND Flash device

This section provides the example LUT instruction sequences for serial NAND flash device (Micron Flash MT29 series). The operation to serial NAND flash is quite similar to serial NOR flash.

READ operation sequence is as following:

- Page Read (Transfer the data from the NAND Flash array to the cache register)
- Get Feature to read the status
- Random Data Read

The following table shows Page Read command sequence.

Table 443. Page Read command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x13	Command code: 0x13
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

The following table shows Get Feature command sequence.

Table 444. Get Feature command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x0F	Command code: 0x0F
1	CMD_SDR	0x1 (Single)	0xC0	Status register address (0xC0)
2	READ_SDR	0x1 (Single)	0x02	2 Byte read data
3-7	STOP (0x0)	0x0	0x00	

The following table shows Random Data Read command sequence.

Table 445. Read from Cache x4 command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x6B	Command code: 0x6B
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if

Table continues on the next page...

Table 445. Read from Cache x4 command (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle number: 8 (serial root clock)
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero.
5-7	STOP (0x0)	0x0	0x00	

Program operation sequence is as follows:

- Write Enable
- Program Load (Transfer the write data to the cache register)
- Program Execute
- Get Feature to read the status

The following table shows Program Load command sequence.

Table 446. Program Load command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x02	Command code: 0x02
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	WRITE_SDR	0x1 (Single)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ] is zero.
4-7	STOP (0x0)	0x0	0x00	

The following table shows Program Execute command sequence.

Table 447. Program Execute command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x10	Command code: 0x10
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

39.7.4 Application on FPGA device

FPGA device must be accessed by AHB command. All AHB accesses to FPGA are transparent to SW driver (no SW intervention). There may be some special requirements from FPGA device.

1. Device type may be different on A1/A2/B1/B2

In this case, clear the MCR2[SAMEDEVICEEN] bit and configure FLSHxCR0 and FLSHxCR1 register separately for up to four external devices.

2. Device needs different wait cycle for programming.

The AHB write wait cycle number can be set separately for these four external devices (by register field FLSHxCR2[AWRWAIT]). Software can configure the sequences in LUT with different DUMMY instructions (operand determines dummy cycle). Note that FlexSPI holds AHB bus ready for this wait time, so AHB bus performance may decrease when this wait time is very long.

3. Device needs different wait cycle for reading.

The AHB Read Sequence index and Sequence Number can be set separately for these four external devices (by register field FLSHxCR2[ARDSEQID] and FLSHxCR2[ARDSEQNUM]). Software can configure the sequences in LUT with different DUMMY instruction (operand determines dummy cycle).

4. Device may be sensitive to read instruction clock cycle number

Device is sensitive to read instruction clock cycle number if its internal memory is implemented similar as FIFO. In this case, software can send the data size information to external device by DATSZ instruction. FPGA device decodes the data size information and determines how much data bytes must be popped.

5. Device may need interval time between chip select valid

This can be handled by register field FLSHxCR1[CSINTERVAL] setting.

6. Device may use SCLK as reference clock for its internal PLL

In this case, SCLK must be free-running and clock frequency must be stable. This can be achieved by setting MCR0[SCKFREERUNEN] and using SDR sequence only.

39.7.5 Overview of error category, flags and triggered sources

The following table gives an overview of the category, flags and triggered sources of errors.

Table 448. Error category, triggered sources and flags in FlexSPI

Error Category	Triggered Sources	Description	Error Flags
Command grant error	AHB write command	Command grant timeout	INTR[AHBCMDGE] is set AHB bus error response

Table continues on the next page...

Table 448. Error category, triggered sources and flags in FlexSPI (continued)

Error Category	Triggered Sources	Description	Error Flags
	AHB read command		INTR[AHBCMDGE] is set AHB bus error response
	IP command		INTR[IPCMDGE] is set
Command check error	AHB write command	- AHB write command with JMP_ON_CS instruction used in the sequence - There is unknown instruction opcode in the sequence. - Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. - Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] is set Command is not executed when an error is detected in command check
	AHB read command	- There is unknown instruction opcode in the sequence. - Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. - Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] is set Command is not executed when an error is detected in command check
	IP command	- IP command with JMP_ON_CS instruction used in the sequence - There is unknown instruction opcode in the sequence - Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence - Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence - Flash boundary across	INTR[IPCMDERR] is set Command is not executed when an error is detected in command check
Command execution error	AHB write command	Command timeout during execution	INTR[AHBCMDERR] is set INTR[SEQTIMEOUT] is set

Table continues on the next page...

Table 448. Error category, triggered sources and flags in FlexSPI (continued)

Error Category	Triggered Sources	Description	Error Flags
			There is AHB bus error response except following case: - AHB write command is triggered by flush (INCR burst ended with AHB_TX_BUF not empty) - AHB bufferable write access and bufferable enabled (AHBCR[BUFFER ABLEEN]=0x1)
	AHB read command		INTR[AHBCMDERR] is set INTR[SEQTIMEOUT] is set AHB bus error response
	IP command		INTR[IPCMDERR] is set INTR[SEQTIMEOUT] is set
AHB Bus timeout	AHB write command	AHB bus timeout (no bus ready return)	INTR[AHBBUSTIMEOUT] is set AHB bus error response
	AHB read command		
Data Learning Failed	Any command	There is no valid sample clock phase found after LEARN instruction is executed.	INTR[DATALEARNFAIL] is set
Security error	IP command	The IP command is not allowed to trigger	INTR[IPCMDSECUREVIO] is set
	AHB command	IP command GCM data check error	INTR[AHBGCMERR] is set

39.8 Memory Map and register definition

This section includes the FlexSPI module memory map and detailed descriptions of all registers.

39.8.1 Register Access

All registers can be accessed with 8-bit, 16-bit, and 32-bit width operations. Never change the setting value of reserved fields in control registers. Changing the value of reserved fields may impact the normal functioning of the controller.

NOTE

For usage that FlexSPI is capable of accessing sensitive memory contents, protection should be done to FlexSPI controller using resource isolation (for example, XRDC, TrustZone, and so on) to make sure only trusted software be allowed to access the FlexSPI controller register interface.

39.8.2 FlexSPI register descriptions

39.8.2.1 FlexSPI memory map

FLEXSPI0 base address: 4013_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register 0 (MCR0)	32	RW	FFFF_80C2h
4h	Module Control Register 1 (MCR1)	32	RW	FFFF_FFFFh
8h	Module Control Register 2 (MCR2)	32	RW	2000_81F7h
Ch	AHB Bus Control Register (AHBCR)	32	RW	0000_0018h
10h	Interrupt Enable Register (INTEN)	32	RW	0000_0000h
14h	Interrupt Register (INTR)	32	RW	0000_0000h
18h	LUT Key Register (LUTKEY)	32	RW	5AF0_5AF0h
1Ch	LUT Control Register (LUTCR)	32	RW	0000_0002h
20h	AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)	32	RW	8000_0010h
24h	AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)	32	RW	8001_0010h
28h	AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)	32	RW	8002_0010h
2Ch	AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)	32	RW	8003_0010h
30h	AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0)	32	RW	8004_0010h
34h	AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0)	32	RW	8005_0010h
38h	AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0)	32	RW	8006_0010h
3Ch	AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0)	32	RW	8007_0010h
60h	Flash Control Register 0 (FLSHA1CR0)	32	RW	0001_0000h
64h	Flash Control Register 0 (FLSHA2CR0)	32	RW	0001_0000h
68h	Flash Control Register 0 (FLSHB1CR0)	32	RW	0001_0000h
6Ch	Flash Control Register 0 (FLSHB2CR0)	32	RW	0001_0000h
70h	Flash Control Register 1 (FLSHA1CR1)	32	RW	0000_0063h
74h	Flash Control Register 1 (FLSHA2CR1)	32	RW	0000_0063h
78h	Flash Control Register 1 (FLSHB1CR1)	32	RW	0000_0063h
7Ch	Flash Control Register 1 (FLSHB2CR1)	32	RW	0000_0063h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
80h	Flash Control Register 2 (FLSHA1CR2)	32	RW	0000_0000h
84h	Flash Control Register 2 (FLSHA2CR2)	32	RW	0000_0000h
88h	Flash Control Register 2 (FLSHB1CR2)	32	RW	0000_0000h
8Ch	Flash Control Register 2 (FLSHB2CR2)	32	RW	0000_0000h
94h	Flash Control Register 4 (FLSHCR4)	32	RW	0000_0000h
A0h	IP Control Register 0 (IPCR0)	32	RW	0000_0000h
A4h	IP Control Register 1 (IPCR1)	32	RW	0000_0000h
B0h	IP Command Register (IPCMD)	32	RW	0000_0000h
B4h	Data Learn Pattern Register (DLPR)	32	RW	0000_0000h
B8h	IP RX FIFO Control Register (IPRXFCR)	32	RW	0000_0000h
BCh	IP TX FIFO Control Register (IPTXFCR)	32	RW	0000_0000h
C0h	DLL Control Register 0 (DLLACR)	32	RW	0000_0100h
C4h	DLL Control Register 0 (DLLBCR)	32	RW	0000_0100h
E0h	Status Register 0 (STS0)	32	R	0000_0002h
E4h	Status Register 1 (STS1)	32	R	0000_0000h
E8h	Status Register 2 (STS2)	32	R	0100_0100h
ECh	AHB Suspend Status Register (AHBSPNDSTS)	32	R	0000_0000h
F0h	IP RX FIFO Status Register (IPRXFSTS)	32	R	0000_0000h
F4h	IP TX FIFO Status Register (IPTXFSTS)	32	R	0000_0000h
100h - 17Ch	IP RX FIFO Data Register a (RFDR0 - RFDR31)	32	R	0000_0000h
180h - 1FCh	IP TX FIFO Data Register a (TFDR0 - TFDR31)	32	W	0000_0000h
200h - 2FCh	LUT a (LUT0 - LUT63)	32	RW	See section
420h	HADDR REMAP START ADDR (HADDRSTART)	32	RW	0000_0000h
424h	HADDR REMAP END ADDR (HADDREND)	32	RW	0000_0000h
428h	HADDR REMAP OFFSET (HADDROFFSET)	32	RW	0000_0000h

39.8.2.2 Module Control Register 0 (MCR0)

Offset

Register	Offset
MCR0	0h

Diagram



Fields

Field	Function
31-24 AHBGRANTWAIT	Timeout wait cycle for AHB command grant. If AHB Triggered Command is not granted by the arbitrator, it will timeout after AHBGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is IP triggered and the read/write data size is too large. When an AHB command grant timeout occurs, an interrupt is generated (INTR[AHBCMDGE]). When INTEN[AHBCMDGEEN] = 1, AHB command is ignored by the arbitrator. NOTE This field is for debug only, keep the default value! It is not allowed to set this field to value 0x0.
23-16 IPGRANTWAIT	Timeout wait cycle for IP command grant. If IP Triggered Command is not granted by the arbitrator, it will timeout after IPGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is AHB triggered and read/write data size is too large. When IP command grant timeout occurs, an interrupt is generated (INTR[IPCMDGE]). When INTEN[IPCMDGEEN] = 0x1, IP command is ignored by the arbitrator. NOTE This field is for debug only, keep the default value! It is not allowed to set this field to value 0x0.
15 LEARNEN	This bit is used to enable/disable the data learning feature. When data learning is disabled, the sampling clock phase 0 is always used for RX data sampling even if LEARN instruction is correctly executed. See Data learning feature for more details. 0b - Disable the data learning feature. 1b - Enable the data learning feature.
14 SCKFREERUNEN	This bit is used to enable SCLK output free-running. For FPGA applications, the external device may use SCLK as reference clock to its internal PLL. 0b - Disable SCLK output free-running. 1b - Enable SCLK output free-running.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 —	Reserved
12 DOZEEN	Doze mode enable bit 0b - Doze mode support disabled. AHB clock and serial clock will not be gated off when there is doze mode request from system. 1b - Doze mode support enabled. AHB clock and serial clock will be gated off when there is doze mode request from system.
11 HSEN	Half Speed Serial Flash Access Enable. This bit enables the divide by 2 of the clock to external serial flash devices (A_SCLK/B_SCLK) for all commands (for both SDR and DDR mode). FlexSPI need to be set into MDIS mode before changing value of HSEN. Otherwise, it is possible to cause issue on internal logic/state machine. 0b - Disable divide by 2 of serial flash clock for half clock frequency. 1b - Enable divide by 2 of serial flash clock for half clock frequency.
10-8 SERCLKDIV	Serial root clock The serial root clock could be divided inside FlexSPI . See Clocks chapter for more details on clocking. NOTE Do not modify this field while the FlexSPI is not active (MCR0[MDIS] = 0). 000b - Divided by 1 001b - Divided by 2 010b - Divided by 3 011b - Divided by 4 100b - Divided by 5 101b - Divided by 6 110b - Divided by 7 111b - Divided by 8
7 —	Reserved
6 —	Reserved
5-4 RXCLKSRC	Sample Clock source selection for Flash Reading See RX Clock Source Features for more details. 00b - Dummy Read strobe generated by FlexSPI Controller and loopback internally.

Table continues on the next page...

Table continued from the previous page...

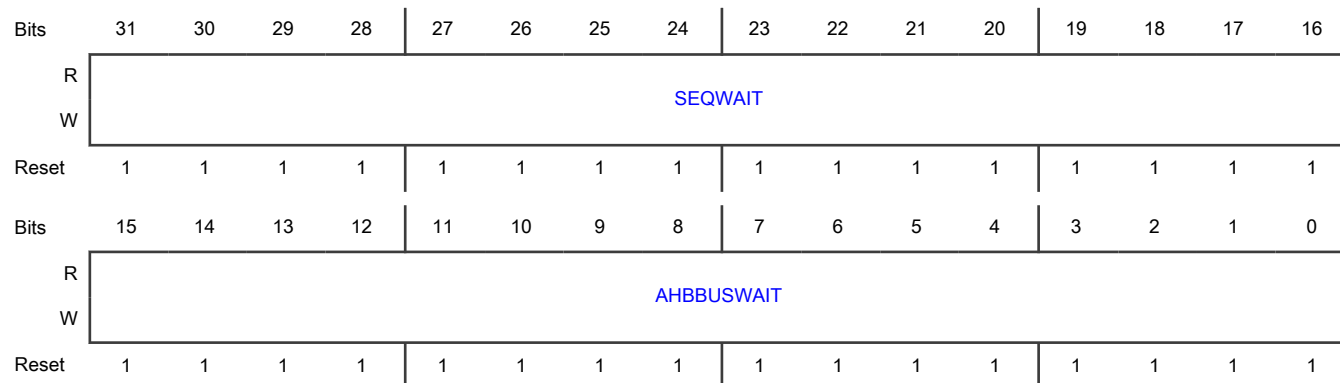
Field	Function
	01b - Dummy Read strobe generated by FlexSPI Controller and loopback from DQS pad. 10b - Reserved 11b - Flash provided Read strobe and input from DQS pad
3-2 —	Reserved
1 MDIS	Module Disable When the module is disabled, AHB/serial clock will be gated off internally to save power. Only register access (except LUT/IP RX FIFO/IP TX FIFO) is allowed. 0b - No impact 1b - Module disable
0 SWRESET	Software Reset This bit is auto-cleared by hardware after software reset is done. Configuration registers will not be reset. 0b - No impact 1b - Software reset

39.8.2.3 Module Control Register 1 (MCR1)

Offset

Register	Offset
MCR1	4h

Diagram



Fields

Field	Function
31-16 SEQWAIT	Command Sequence Execution will timeout and abort after SEQWAIT * 1024 Serial Root Clock cycles. When sequence execution timeout occurs, there will be an interrupt generated (INTR[SEQTIMEOUT]) if this interrupt is enabled (INTEN[SEQTIMEOUTEN] is set 0x1) and AHB command is ignored by arbitrator. NOTE It is not allowed to set this field to value 0x0.
15-0 AHBBUSWAIT	AHB Bus wait AHB Read/Write access to Serial Flash Memory space will timeout if data is not received from Flash or if data is not transmitted after AHBBUSWAIT * 1024 AHB clock cycles, AHB Bus will get an error response. When AHB bus timeout occurs, an interrupt is generated (INTR[AHBBUSERROREN]). When (INTR[AHBBUSERROREN]) = 0x1, AHB command is ignored by the arbitrator. NOTE It is not allowed to set this field to value 0x0.

39.8.2.4 Module Control Register 2 (MCR2)

Offset

Register	Offset
MCR2	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RESUMEWAIT								Reserved		Reserv ed	Reserv ed	Reserv ed	Reserved	Reserved	Reserved
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAME DEV...	0	Reserv ed	Reserv ed	CLR A HBB...	Reserved		Reserved								
W		CLR LE AR...														
Reset	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1

Fields

Field	Function
31-24 RESUMEWAIT	Wait cycle (in AHB clock cycle) for idle state before suspended command sequence resumed. See Command abort and suspend for more details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-21 —	Reserved
20 —	Reserved
19 —	Reserved
18-17 —	Reserved
16 —	Reserved
15 SAMEDEVICEEN	All external devices are same devices (both in type and size) for A1/A2/B1/B2. 0b - In Individual mode, FLSHA1CRx/FLSHA2CRx/FLSHB1CRx/FLSHB2CRx register setting will be applied to Flash A1/A2/B1/B2 separately. In Parallel mode, FLSHA1CRx register setting will be applied to Flash A1 and B1, FLSHA2CRx register setting will be applied to Flash A2 and B2. FLSHB1CRx/FLSHB2CRx register setting will be ignored. 1b - FLSHA1CR0/FLSHA1CR1/FLSHA1CR2 register setting will be applied to Flash A1/A2/B1/B2. FLSHA2CRx/FLSHB1CRx/FLSHB2CRx will be ignored.
14 CLRLEARNPBASE	The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately. 0b - No impact 1b - The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately.
13 —	Reserved
12 —	Reserved
11 CLRAHBBufferPT	Clear AHB buffer This bit determines whether AHB RX Buffer and AHB TX Buffer will be cleared automatically when FlexSPI returns STOP mode ACK. Software should set this bit if AHB RX Buffer or AHB TX Buffer will be powered off in STOP mode. Otherwise AHB read access after exiting STOP mode may hit AHB RX Buffer or AHB TX Buffer but their data entries are invalid. 0b - AHB RX/TX Buffer will not be cleared automatically when FlexSPI returns Stop mode ACK. 1b - AHB RX/TX Buffer will be cleared automatically when FlexSPI returns Stop mode ACK.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-9 —	Reserved
8-0 —	Reserved

39.8.2.5 AHB Bus Control Register (AHBCR)

Offset

Register	Offset
AHBCR	Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserved				Reserv	Reserv	Reserv	Reserv
W													ed	ed	ed	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	Reserved		Reserv	Reserv	READ	Reserv	Reserv	RESU	READ	PREF	BUFF	CACH	CLRA	Reserv	APAR
W	ed			ed	ed	SZA...	ed	ed	MED...	ADD...	ETC...	ERA...	ABL...	HBT...	ed	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Fields

Field	Function
31-27 —	Reserved
26-22 —	Reserved
21-20 —	Reserved
19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11 —	Reserved
10 READSZALIGN	AHB Read Size Alignment NOTE When this bit is set to 1, data prefetching will be disabled regardless any other setting(PREFETCH_EN,OTFAD_EN...). And read size will be up sized to 8 bytes aligned. For example, if current AHB read size is 12 bytes, it will be aligned to 16 bytes. This bit is expected to be used when fetch size need to be 8 bytes aligned but no speculative fetching is wanted. 0b - AHB read size will be decided by other register setting like PREFETCH_EN,OTFAD_EN... 1b - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 RESUMEDISABLE	AHB Read Resume Disable NOTE When OTFAD is enabled, this bit won't affect resume function, and AHB read resume is always disabled. When OTFAD is disabled, suspended AHB read prefetch will start to resume based on the value of this bit and MRC2.RESUMEWAIT

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Suspended AHB read prefetch will start to resume when AHB is IDLE 1b - Suspended AHB read prefetch will not resume once it is aborted
6 READADDROP T	AHB Read Address option bit. This option bit is intended to remove AHB burst start address alignment limitation. When FlexSPI controller is used for FPGA application, it may be required for the FlexSPI to fetch exactly the byte number as AHB burst. In this case, FPGA device should be designed as non-word addressable and this option bit must be cleared. NOTE When OTFAD is enabled, FlexSPI will work as READADDROPT set as 1 regardless the value of this bit. 0b - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is word-addressable. 1b - There is no AHB read burst start address alignment limitation. FlexSPI will fetch more data than AHB burst required to meet the alignment requirement.
5 PREFETCHEN	AHB Read Prefetch Enable. When AHB read prefetch is enabled, FlexSPI will fetch more flash read data than current AHB burst needed so that the read latency for next AHB read access will be reduced. NOTE When OTFAD enabled, AHB prefetch function will be enabled regardless of the value of this field.
4 BUFFERABLEEN	Enable AHB bus bufferable write access support. This field affects the last beat of AHB write access, see AHB write access to Flash for more details. 0b - Disabled. For all AHB write accesses (bufferable or non-bufferable), FlexSPI will return AHB Bus ready after all data is transmitted to external device and AHB command finished. 1b - Enabled. For AHB bufferable write access, FlexSPI will return AHB Bus ready when the AHB command is granted by arbitrator and will not wait for AHB command finished.
3 CACHABLEEN	Enable AHB bus cachable read access support. 0b - Disabled. When there is AHB bus cachable read access, FlexSPI will not check whether it hit AHB TX Buffer. 1b - Enabled. When there is AHB bus cachable read access, FlexSPI will check whether it hit AHB TX Buffer first.
2 CLRAHBTXBUF	Clear the status/pointers of AHB TX Buffer. Auto-cleared. 0b - No function. This bit will clear automatically when the clear operation completes. This bit always reads 0. 1b - Clear operation enable.
1	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
0 APAREN	Parallel mode enabled for AHB triggered Command (both read and write). 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.

39.8.2.6 Interrupt Enable Register (INTEN)

Offset

Register	Offset
INTEN	10h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		KEYE RRO...	KEYD ONE...	SEQTI ME...	AHBB USE...	SCKS TOP...	SCKS TOP...	DATA LEA...	IPTXW EEN	IPRX WAEN	AHBC MDE...	IPCM DER...	AHBC MDG...	IPCM DGE...	IPCM DDO...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13 KEYERROREN	OTFAD key blob processing error interrupt enable. See Interrupts chapter for more details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable interrupt or no impact 1b - Enable interrupt
12 KEYDONEEN	OTFAD key blob processing done interrupt enable. See Interrupts chapter for more details. 0b - Disable interrupt or no impact 1b - Enable interrupt
11 SEQTIMEOUT EN	Sequence execution timeout interrupt enable. See Interrupts chapter for more details. 0b - Disable interrupt or no impact 1b - Enable interrupt
10 AHBBUSERRO REN	AHB Bus error interrupt enable. See Interrupts chapter for more details. 0b - Disable interrupt or no impact 1b - Enable interrupt
9 SCKSTOPBYW REN	SCLK is stopped during command sequence because Async TX FIFO empty interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
8 SCKSTOPBYR DEN	SCLK is stopped during command sequence because Async RX FIFO full interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
7 DATALEARNF AILEN	Data Learning failed interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
6 IPTXWEEN	IP TX FIFO WaterMark empty interrupt enable. IP TX FIFO has more empty space than WaterMark level interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
5 IPRXWAEN	IP RX FIFO WaterMark available interrupt enable. IP RX FIFO has more valid data than WaterMark level interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 AHBCMDERREN	AHB triggered Command Sequences Error Detected interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
3 IPCMDERREN	IP triggered Command Sequences Error Detected interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
2 AHBCMDGEEN	AHB triggered Command Sequences Grant Timeout interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
1 IPCMDGEEN	IP triggered Command Sequences Grant Timeout interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
0 IPCMDDONEEN	IP triggered Command Sequences Execution finished interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt

39.8.2.7 Interrupt Register (INTR)

Offset

Register	Offset
INTR	14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		KEYE	KEYD	SEQTI	AHBB	SCKS	SCKS	DATA	IPTXW	IPRX	AHBC	IPCM	AHBC	IPCM	IPCM
W			RROR	ONE	ME...	USE...	TOP...	TOP...	LEA...	E	WA	MDE...	DERR	MDGE	DGE	DDO...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13 KEYERROR	OTFAD key blob processing error interrupt.
12 KEYDONE	OTFAD key blob processing done interrupt.
11 SEQTIMEOUT	Sequence execution timeout interrupt.
10 AHBBUSERRO R	AHB Bus timeout or AHB bus illegal access Flash during OTFAD key blob processing interrupt.
9 SCKSTOPBYW R	SCLK is stopped during command sequence because Async TX FIFO empty interrupt.
8 SCKSTOPBYR D	SCLK is stopped during command sequence because Async RX FIFO full interrupt.
7 DATALEARNF AIL	Data Learning failed interrupt.
6 IPTXWE	IP TX FIFO watermark empty interrupt. IP TX FIFO has more empty space than WaterMark level interrupt.
5 IPRXWA	IP RX FIFO watermark available interrupt. IP RX FIFO has more valid data than WaterMark level interrupt.
4 AHBCMDERR	AHB triggered Command Sequences Error Detected interrupt. When an error detected for AHB command, this command will be ignored and not executed at all.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 IPCMDERR	IP triggered Command Sequences Error Detected interrupt. When an error detected for IP command, this command will be ignored and not executed at all.
2 AHBCMDGE	AHB triggered Command Sequences Grant Timeout interrupt.
1 IPCMDGE	IP triggered Command Sequences Grant Timeout interrupt.
0 IPCMDDONE	IP triggered Command Sequences Execution finished interrupt. This interrupt is also generated when there is IPCMDGE or IPCMDERR interrupt generated.

39.8.2.8 LUT Key Register (LUTKEY)

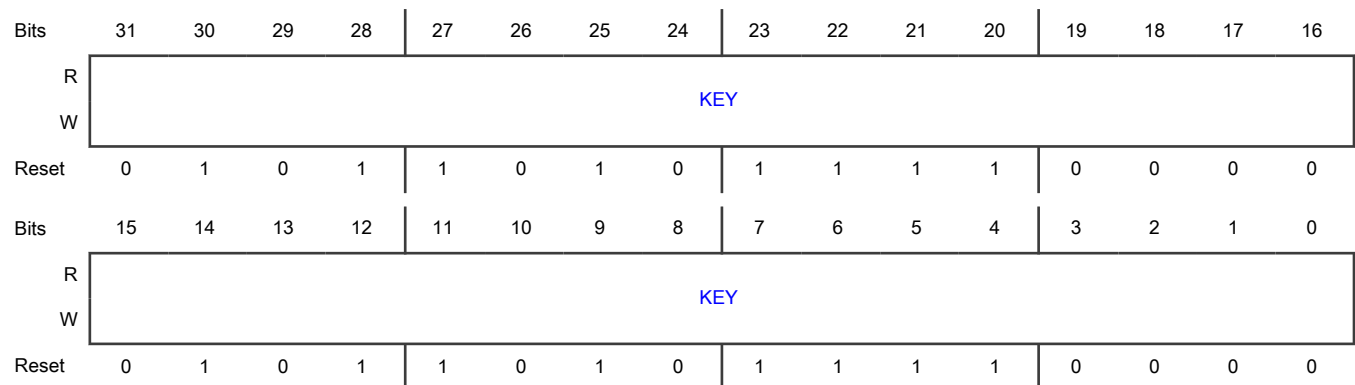
Offset

Register	Offset
LUTKEY	18h

Function

The LUT Key Register contains the key to lock and unlock LUT. See [LUT](#) for details.

Diagram



Fields

Field	Function
31-0 KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.

39.8.2.9 LUT Control Register (LUTCR)

Offset

Register	Offset
LUTCR	1Ch

Function

The LUT control register is used along with LUTKEY register to lock or unlock LUT. This register has to be written immediately after writing 0x5AF05AF0 to LUTKEY register for the lock or unlock operation to be successful. See [LUT](#) for details on locking/unlocking LUT. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Diagram

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

Fields

Field	Function
31-3 —	Reserved
2 —	Reserved
1 UNLOCK	Unlock LUT 0b - No impact 1b - Unlock LUT, the LUT can be written
0 LOCK	Lock LUT 0b - No impact 1b - Lock LUT, LUT will be locked and can't be written

39.8.2.10 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

Offset

Register	Offset
AHBRXBUF0CR0	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.11 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

Offset

Register	Offset
AHBRXBUF1CR0	24h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.12 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

Offset

Register	Offset
AHBRXBUF2CR0	28h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserved	Reserved		PRIORITY				Reserved				MSTRID			
W	ETC...															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

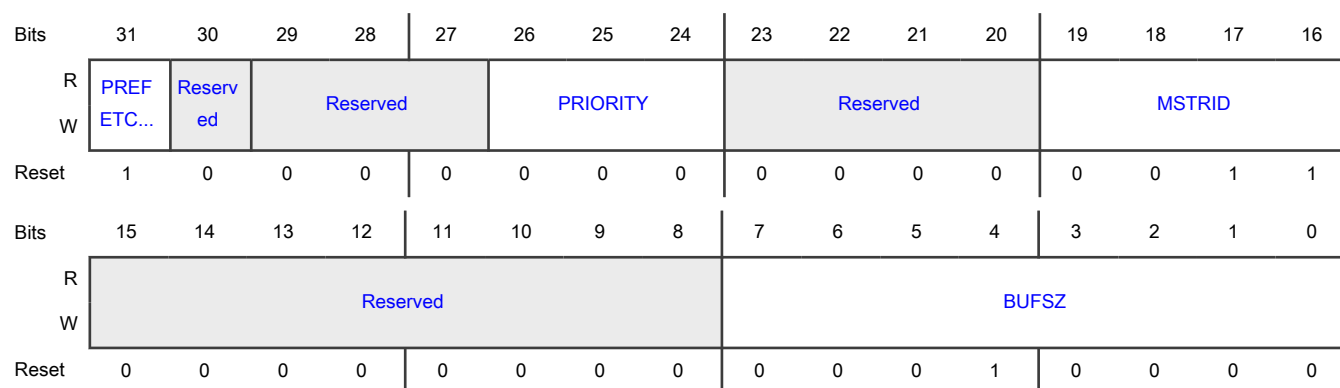
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.13 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

Offset

Register	Offset
AHBRXBUF3CR0	2Ch

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.14 AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0)

Offset

Register	Offset
AHBRXBUF4CR0	30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.15 AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0)

Offset

Register	Offset
AHBRXBUF5CR0	34h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.16 AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0)

Offset

Register	Offset
AHBRXBUF6CR0	38h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserved	Reserved		PRIORITY				Reserved				MSTRID			
W	ETC...															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

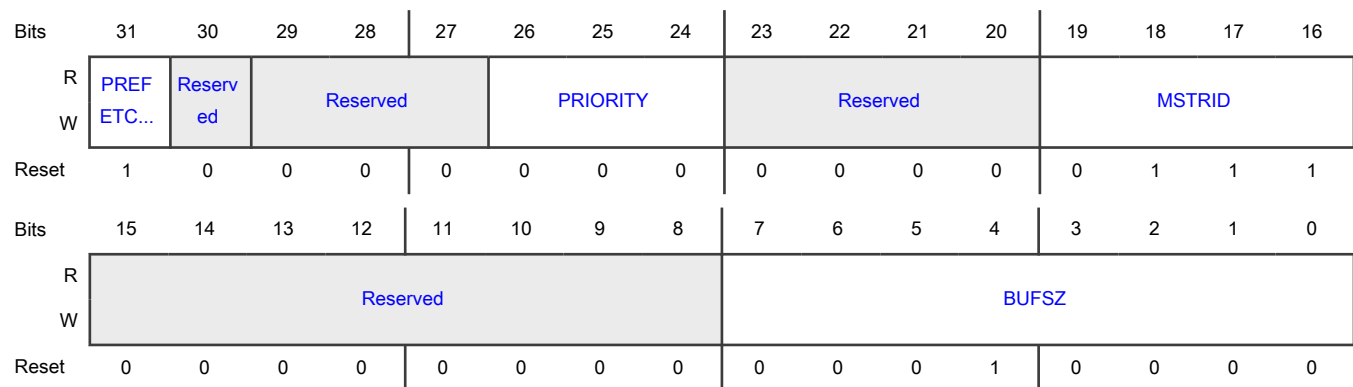
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.17 AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0)

Offset

Register	Offset
AHBRXBUF7CR0	3Ch

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.2.18 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

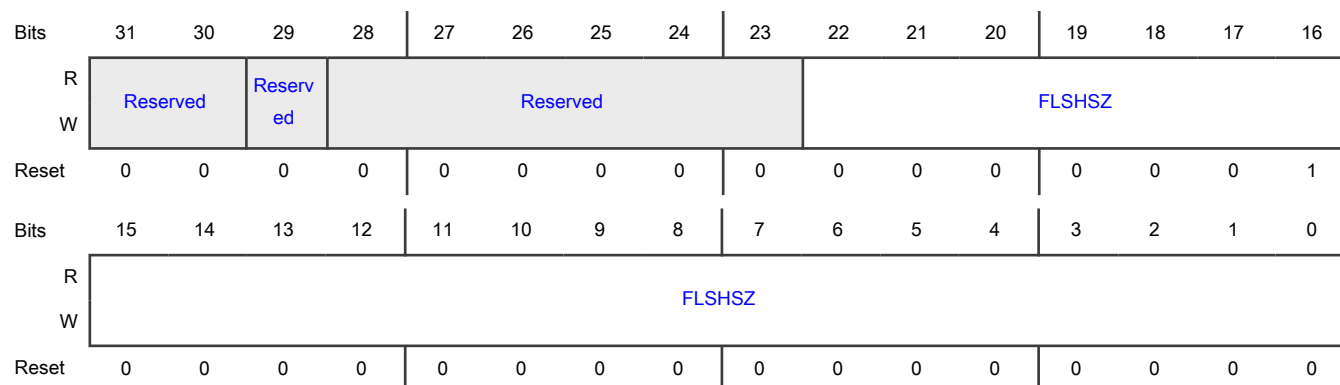
Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

Function

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Select).

Diagram



Fields

Field	Function
31-30 —	Reserved
29 —	Reserved
28-23 —	Reserved
22-0 FLSHSZ	Flash Size in KByte. Refer to the chip-specific section for the maximum flash size supported. When the FLSHSZ setting value is greater than the maximum size, the device flash size is interpreted as the maximum. The maximum flash size is the largest flash size supported for a single device, and also the maximum total flash size supported for all devices (up to 4). If the total flash size is larger than the maximum size, then only the maximum flash size address space is accessible.

39.8.2.19 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

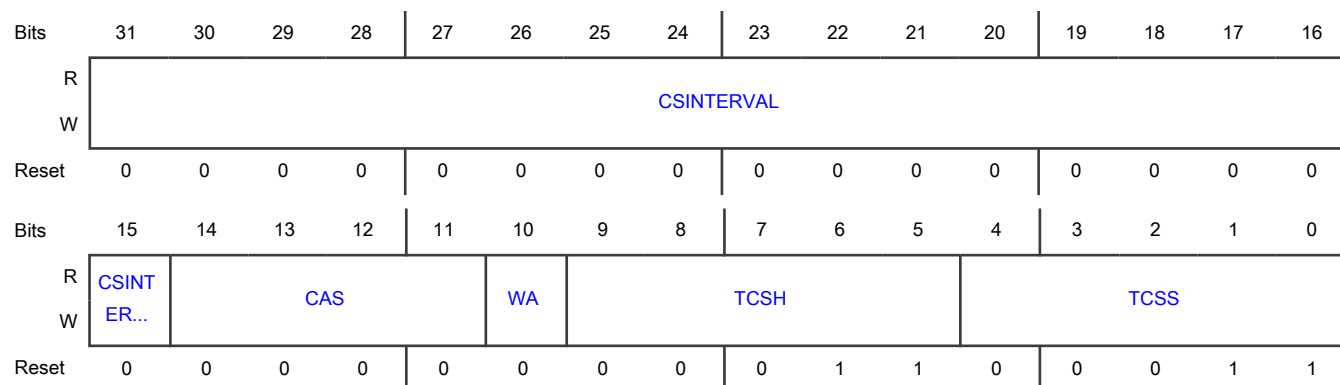
Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

Function

The Flash control register 1 contains the setting for FlexSPI to meet Flash device specific timings and Flash internal address space.

Diagram



Fields

Field	Function
31-16 CSINTERVAL	This field is used to set the minimum interval between flash device chip select deassertion and flash device chip select assertion. If external flash has a limitation on the interval between command sequences, this field should be set accordingly. If there is no limitation, set this field with value 0x0. When CSINTERVALUNIT is 0x0, the chip selecti invalid interval is: CSINTERVAL * 1 serial clock cycle; When CSINTERVALUNIT is 0x1, the chip Select invalid interval is: CSINTERVAL * 256 serial clock cycle. NOTE The chip Select interval is 2 cycle at least even if CSINTERVAL is less than 2.
15 CSINTERVALU NIT	CS interval unit 0b - The CS interval unit is 1 serial clock cycle 1b - The CS interval unit is 256 serial clock cycle
14-11	Column Address Size.

Table continues on the next page...

Table continued from the previous page...

Field	Function
CAS	When external flash has separate address field for row address and column address, this field should be set to flash column address bit width. FlexSPI will automatically split flash mapped address to Row address and Column address according to CAS field and WA field setting. This bit should be set to 0x0 when external Flash don't support column address. FlexSPI will transmit all flash address bits as Row address. For flash address mapping, see AHB access memory map for more detail.
10 WA	Word Addressable. This bit should be set when external Flash is word addressable. If Flash is word addressable, it should be access in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash. For flash address mapping, see AHB access memory map for more detail. 0b - This bit should be set as 0 when external Flash is byte addressable. 1b - This bit should be set as 1 when external Flash is word addressable. If Flash is word addressable, it should be accessed in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash.
9-5 TCSH	Serial Flash CS Hold time. This field is used to meet flash TCSH timing requirement. Serial flash CS Hold time promised by FlexSPI is: (TCSH + 1/2) serial clock cycles for DDR mode, and TCSH serial clock cycles for SDR mode. See Output timing between chip select and SCLK for more detail.
4-0 TCSS	Serial Flash CS setup time. This field is used to meet flash TCSS timing requirement. Serial flash CS Setup time promised by FlexSPI is: (TCSS + 1/2) serial root clock cycles (for both SDR and DDR mode). See Output timing between chip select and SCLK for more details.

39.8.2.20 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

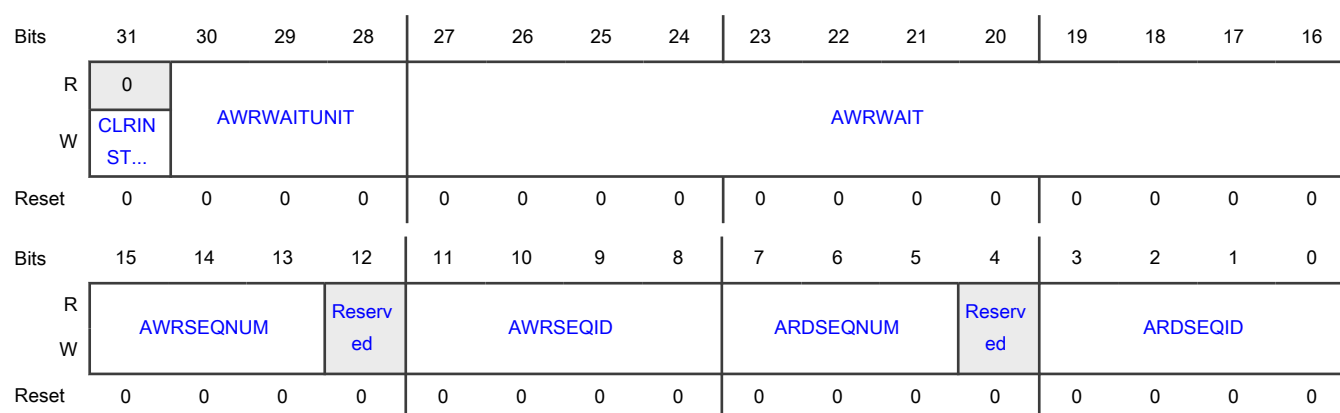
Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

Function

Flash Control Register 2 contains setting field for AHB Bus access configuration. If the 4 external device are in different types, AHB read/write command may use different command sequences and AHB bus ready wait time may be also different.

Diagram



Fields

Field	Function
31 CLRINSTRPTR	Clear the instruction pointer which is internally saved pointer by JMP_ON_CS. See Programmable Sequence Engine for details. This field is used for AHB Read access to external Flash supporting XIP (Execute-In-Place) mode.
30-28 AWRWAITUNIT	AWRWAIT unit 000b - The AWRWAIT unit is 2 AHB clock cycle 001b - The AWRWAIT unit is 8 AHB clock cycle 010b - The AWRWAIT unit is 32 AHB clock cycle 011b - The AWRWAIT unit is 128 AHB clock cycle 100b - The AWRWAIT unit is 512 AHB clock cycle 101b - The AWRWAIT unit is 2048 AHB clock cycle 110b - The AWRWAIT unit is 8192 AHB clock cycle 111b - The AWRWAIT unit is 32768 AHB clock cycle
27-16 AWRWAIT	For certain devices (such as FPGA), it need some time to write data into internal memory after the command sequences finished on FlexSPI interface. If another Read command sequence comes before previous programming finished internally, the read data may be wrong. This field is used to hold AHB Bus ready for AHB write access to wait the programming finished in external device. Then there will be no AHB read command triggered before the programming finished in external device. The Wait cycle between AHB triggered command sequences finished on FlexSPI interface and AHB return Bus ready: AWRWAIT * AWRWAITUNIT
15-13 AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices (e.g. HyperFlash/HyperRam/Serial NAND flash), a Flash programming access is done by several command sequences. AHB write Command will trigger (AWRSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE • Software should make sure the last sequence index never exceed LUT sequence numbers: $AWRSEQID + AWRSEQNUM < 16$ • Software needs to make sure that the fields AWRSEQNUM and LUT are configured correctly according to external device spec. FlexSPI doesn't check the sequence and just executes them one by one.
12 —	Reserved
11-8 AWRSEQID	Sequence Index for AHB Write triggered Command.
7-5 ARDSEQNUM	Sequence Number for AHB Read triggered Command in LUT. For certain flash devices (e.g. HyperFlash/HyperRam/Serial NAND flash), a Flash read access is done by several command sequences. AHB read Command will trigger (ARDSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. NOTE • Software should make sure the last sequence index never exceed LUT sequence numbers: $ARDSEQID + ARDSEQNUM \leq 16$ • Software needs to make sure that the fields ARDSEQNUM and LUT are configured correctly according to external device spec. FlexSPI does not check the sequence and just executes them one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read triggered Command in LUT.

39.8.2.21 Flash Control Register 4 (FLSHCR4)

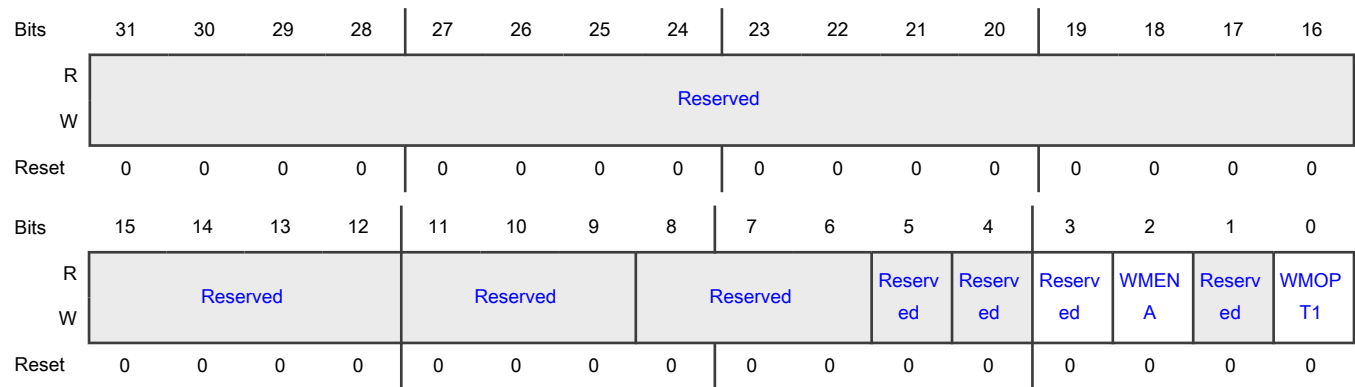
Offset

Register	Offset
FLSHCR4	94h

Function

The flash control register 4 provides the configuration for all external devices.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 WMENA	Write mask enable bit for flash device on port A. When write mask function is needed for memory device on port A, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will not be driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
1 —	Reserved
0 WMOPT1	Write mask option bit 1. This option bit could be used to remove AHB and IP write burst start address alignment limitation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - DQS pin will be used as Write Mask when writing to external device. There is no limitation on AHB/IP write burst start address alignment when flash is accessed in individual mode.
	1b - DQS pin will not be used as Write Mask when writing to external device. There is limitation on AHB/IP write burst start address alignment when flash is accessed in individual mode.

39.8.2.22 IP Control Register 0 (IPCR0)

Offset

Register	Offset
IPCR0	A0h

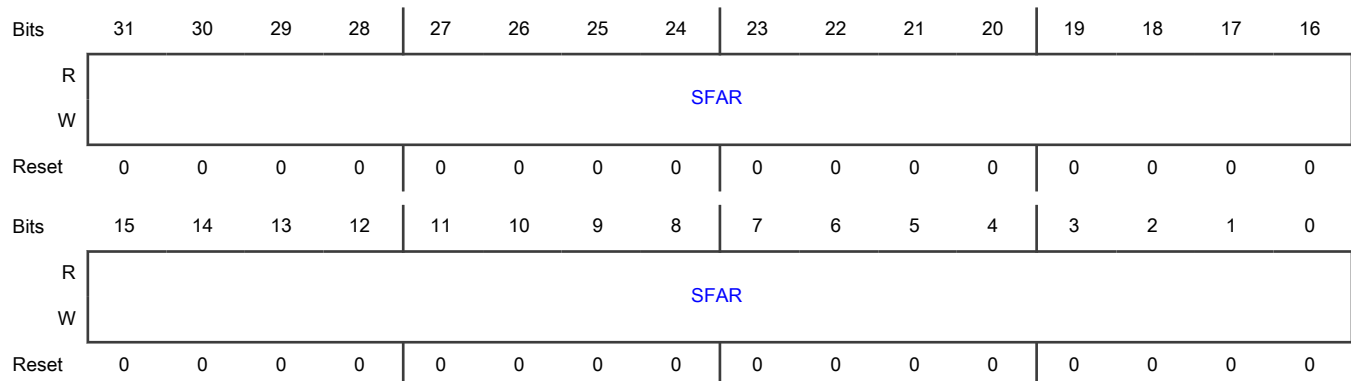
Function

The IP control registers provide all the configuration required for IP commands. This register provides the flash device's start address, instead of SoC address, to be accessed for IP command. FlexSPI will determine the chip select automatically according to this start address.

NOTE

- It's not allowed to issue IP command crossing Flash device boundaries. Otherwise there will be IPCMDERR interrupt generated.
- This register should be set before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

Diagram



Fields

Field	Function
31-0	Serial Flash Address for IP command.
SFAR	For IP command, the address in IPCR0[SFAR] should be the flash device's address without base address.

39.8.2.23 IP Control Register 1 (IPCR1)

Offset

Register	Offset
IPCR1	A4h

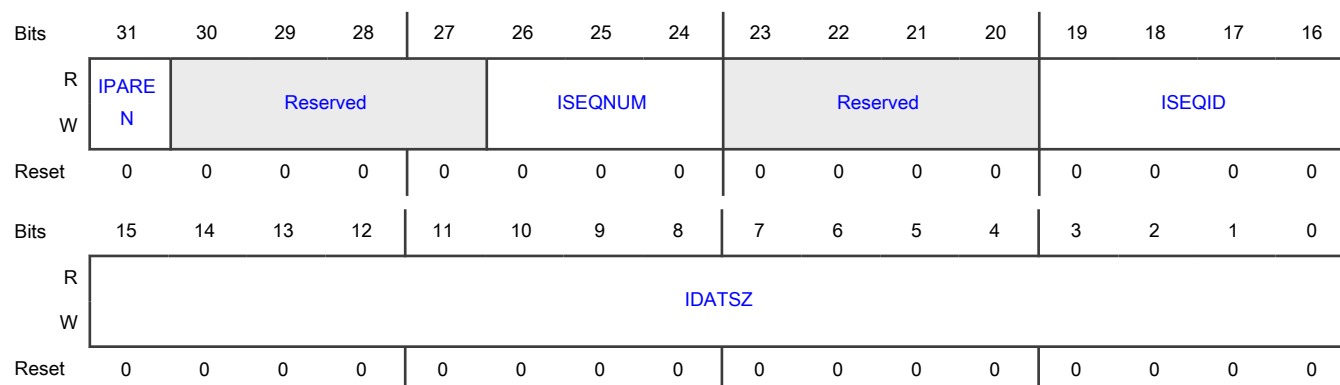
Function

The IP control registers provide all the configuration required for IP command. This register provides the flash read/program data size, sequence index in LUT, sequence number and individual/parallel mode setting for IP command.

NOTE

- This register should be written before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

Diagram



Fields

Field	Function
31 IPAREN	Parallel mode Enabled for IP command. 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.
30-27 —	Reserved
26-24 ISEQNUM	Sequence Number for IP command: ISEQNUM+1.
23-20 —	Reserved
19-16	Sequence Index in LUT for IP command.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ISEQID	
15-0 IDATSZ	Flash Read/Program Data Size (in Bytes) for IP command.

39.8.2.24 IP Command Register (IPCMD)

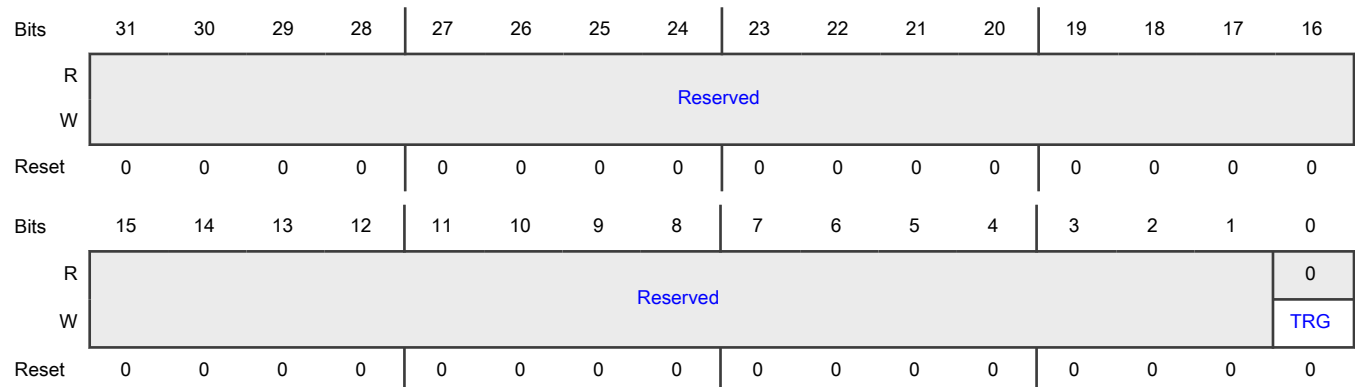
Offset

Register	Offset
IPCMD	B0h

Function

This register is used to trigger an IP command to access external flash device. IP command will be executed on FlexSPI interface after granted by arbitrator.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 TRG	Setting this bit will trigger an IP Command. NOTE It's not allowed to trigger another IP command before previous IP command is finished on FlexSPI interface. Software need to poll register bit INTR[IPCMDDONE] or wait for this interrupt in order to wait for IP command finished.

39.8.2.25 Data Learn Pattern Register (DLPR)

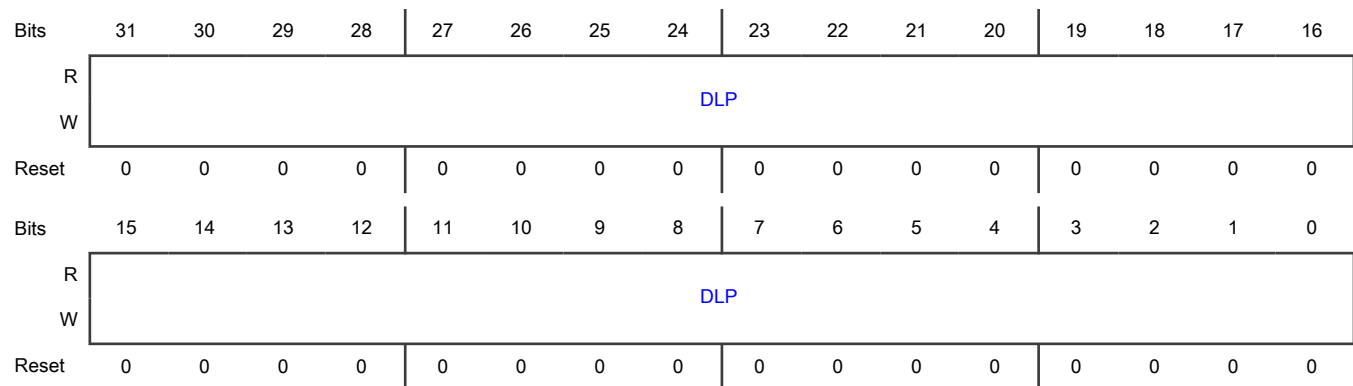
Offset

Register	Offset
DLPR	B4h

Function

This register provides the pattern to be used during Data Learning in FlexSPI.

Diagram



Fields

Field	Function
31-0	Data Learning Pattern.
DLP	The data learning pattern bit number is determined by operand in LEARN_SDR/LEARN_DDR Instruction Code. Data learning pattern bit number will be never more than 32 bits. If the operand in Instruction code is more than 32, 32 bits pattern will be used. If data learning pattern bit number is less than 32 bits, the lower pattern bits will be used. For more details, See Data learning feature .

39.8.2.26 IP RX FIFO Control Register (IPRXFCR)

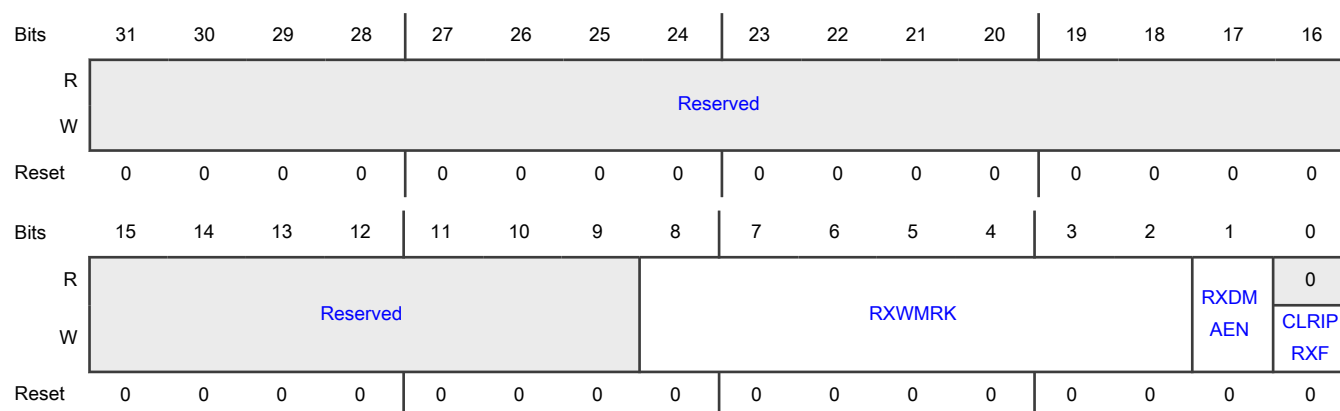
Offset

Register	Offset
IPRXFCR	B8h

Function

This register provides the configuration fields for IP RX FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 RXWMRK	Watermark level is $(RXWMRK+1)*64$ bits. INTR[IPRXWA] is set when filling level in IP RX FIFO is more than Watermark level by FlexSPI. There will be a DMA request when the filling level is more than Watermark level and DMA read is enabled (register bit RXDMAEN is set). There will be an IPRXWA (IP RX FIFO watermark available) interrupt generated when the filling level is more than Watermark level and IPRXWA interrupt is enabled (register bit INTEN[IPRXWAEN] is set). NOTE After write-1-clear to INTR[IPRXWA], read address should be rolled back to start address (memory mapped).
1 RXDMAEN	IP RX FIFO reading by DMA enabled. 0b - IP RX FIFO would be read by processor. 1b - IP RX FIFO would be read by DMA.
0 CLRIPRXF	Clear all valid data entries in IP RX FIFO. The read/write pointers in IP RX FIFO will be reset. 0b - No function. 1b - A clock cycle pulse to clear all valid data entries in IP RX FIFO.

39.8.2.27 IP TX FIFO Control Register (IPTXFCR)

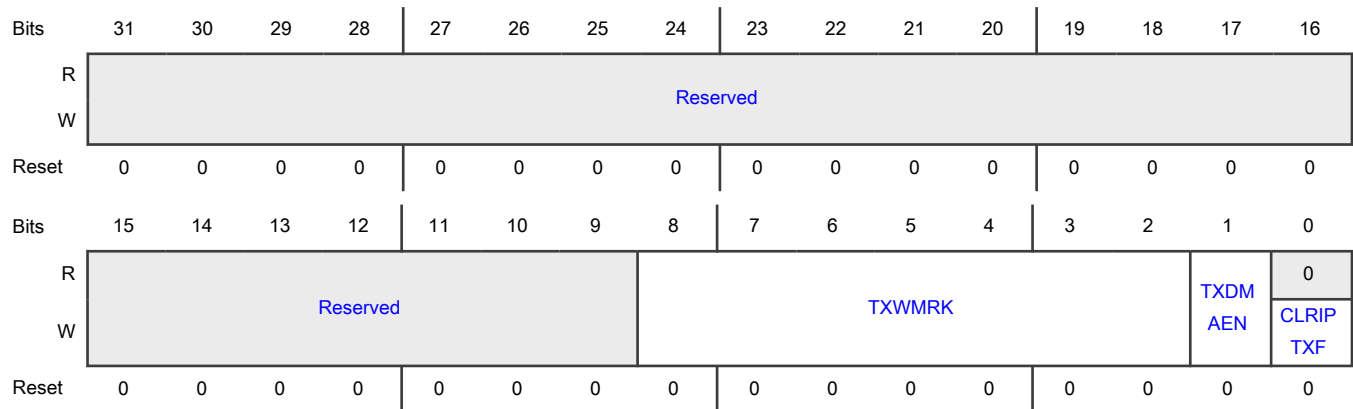
Offset

Register	Offset
IPTXFCR	BCh

Function

This register provides the configuration fields for IP TX FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 TXWMRK	Watermark level is $(TXWMRK+1)*64$ Bits. Interrupt register bit INTR[IPTXWE] is set when empty level in IP TX FIFO is more than Watermark level by FlexSPI. There will be a DMA request when empty level is more than Watermark level and DMA filling is enable (register bit TXDMAEN is set). There will be an IPTXWE (IP TX FIFO Watermark Empty) interrupt generated when empty level is more than Watermark level and IPTXWE interrupt is enable (register bit INTEN[IPTXWEEN] is set). NOTE - The watermark level should be no more than the write window. - The watermark level should be no more than IP TX FIFO size. - The write address to IP RX FIFO should roll back to the start address of write window after pushing IP TX FIFO by writing-one-clear to INTR[IPTXWE].
1 TXDMAEN	IP TX FIFO filling by DMA enabled. 0b - IP TX FIFO would be filled by processor. 1b - IP TX FIFO would be filled by DMA.
0 CLRIP TXF	Clear all valid data entries in IP TX FIFO. The read/write pointers in IP TX FIFO will be reset. 0b - No function. 1b - A clock cycle pulse to clear all valid data entries in IP TX FIFO.

39.8.2.28 DLL Control Register 0 (DLLACR - DLLBCR)

Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

Function

This register provides the configuration fields for Flash A/B sample clock DLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	OVRDVAL						OVRDEN	Reserved	SLVDLYTARGET				Reserved	DLLRESET	DLLLEN
W	Reserved							OVRDEN	Reserved					Reserved	DLLRESET	DLLLEN
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1. See DLL configuration for sampling for more details.
8 OVRDEN	Slave clock delay line delay cell number selection override enable. 0b - Slave clock delay line delay cell number selection override is disabled. 1b - Slave clock delay line delay cell number selection override is enabled.
7 —	Reserved
6-3 SLVDLYTARGET	The delay target for slave delay line is: $((\text{SLVDLYTARGET}+1) * 1/32 * \text{clock cycle of reference clock (serial root clock)})$. If serial root clock is $\geq 100 \text{ MHz}$, DLLLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0 is recommended.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	Reserved
1 DLLRESET	DLL reset Software could force a reset on DLL by setting this field to 0x1. This will cause the DLL to lose lock and re-calibrate to detect a ref_clock half period phase shift. The reset action is edge triggered, so the software needs to clear this bit after setting this bit (no delay limitation). 0b - No function. 1b - Software could force a reset on DLL by setting this field to 0x1.
0 DLLEN	DLL calibration enable. When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored. 0b - DLL calibration is disabled 1b - DLL calibration is enabled

39.8.2.29 Status Register 0 (STS0)

Offset

Register	Offset
STS0	E0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved				DATALEARNPHASEA				ARBCMDSRC		ARBID LE	SEQID LE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-12 —	Reserved
11-8 —	Reserved
7-4 DATALEARNP HASEA	Indicate the sampling clock phase selection on Port A after Data Learning. There are 16 clock phases for sampling clock which is generated by delay cell line. When data learning feature is not enabled, the default clock phase 0 will be used to sample Flash read data. When data learning feature is enabled and LEARN_SDR/LEARN_DDR instruction executed correctly on FlexSPI Interface, FlexSPI will determine the correct clock phase to sample Flash read data. See Data Learning Feature for more details.
3-2 ARBCMDSRC	This status field indicates the trigger source of current command sequence granted by arbitrator. This field value is meaningless when ARB_CTL is not busy (STS0[ARBIDLE]=0x1). 00b - Triggered by AHB read command. 01b - Triggered by AHB write command. 10b - Triggered by IP command (triggered by setting register bit IPCMD[TRG]). 11b - Triggered by suspended command (resumed).
1 ARBIDLE	This status bit indicates the state machine in ARB_CTL is busy and there is command sequence granted by arbitrator and not finished yet on FlexSPI interface. When ARB_CTL state (ARBIDLE=0x1) is idle, there will be no transaction on FlexSPI interface also (SEQIDLE=0x1). So this bit should be polled to wait for FlexSPI controller become idle instead of SEQIDLE.
0 SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface. 0b - State machine in SEQ_CTL is not idle. 1b - State machine in SEQ_CTL is idle.

39.8.2.30 Status Register 1 (STS1)

Offset

Register	Offset
STS1	E4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 IPCMDERRCODE	Indicates the Error Code when IP command Error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - IP command with JMP_ON_CS instruction used in the sequence. 0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 0110b - Flash access start address exceed the whole flash address range (A1/A2/B1/B2). 1110b - Sequence execution timeout. 1111b - Flash boundary crossed.
23-20 —	Reserved
19-16 IPCMDERRID	Indicates the sequence Index when IP command error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c).
15-12 —	Reserved
11-8 AHBCMDERRCODE	Indicates the Error Code when AHB command Error detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - AHB Write command with JMP_ON_CS instruction used in the sequence.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 1110b - Sequence execution timeout.
7-4 —	Reserved
3-0 AHBCMDERRI D	Indicates the sequence index when an AHB command error is detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c).

39.8.2.31 Status Register 2 (STS2)

Offset

Register	Offset
STS2	E8h

Function

This register indicates the status of Flash A and B sample clock DLLs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				BREFSEL				BSLVSEL				BREFL OCK		BSLVL OCK	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AREFSEL				ASLVSEL				AREFL OCK		ASLVL OCK	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29-24 BREFSEL	Flash B sample clock reference delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
23-18 BSLVSEL	Flash B sample clock slave delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
17 BREFLOCK	Flash B sample clock reference delay line locked. 0b - Flash B sample clock reference delay line is not locked. 1b - Flash B sample clock reference delay line is locked.
16 BSLVLOCK	Flash B sample clock slave delay line locked. 0b - Flash B sample clock slave delay line is not locked. 1b - Flash B sample clock slave delay line is locked.
15-14 —	Reserved
13-8 AREFSEL	Flash A sample clock reference delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
7-2 ASLVSEL	Flash A sample clock slave delay line delay cell number selection . There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
1 AREFLOCK	Flash A sample clock reference delay line locked. 0b - Flash A sample clock reference delay line is not locked 1b - Flash A sample clock reference delay line is locked
0 ASLVLOCK	Flash A sample clock slave delay line locked. 0b - Flash A sample clock slave delay line is not locked 1b - Flash A sample clock slave delay line is locked

39.8.2.32 AHB Suspend Status Register (AHBSPNDSTS)

Offset

Register	Offset
AHBSPNDSTS	ECh

Function

Indicates the status of Suspended AHB Read Prefetch command sequence. When there is IP/AHB command triggered and arbitrator is processing an AHB Read sequence (prefetching more data not for current AHB burst), the prefetch sequence will be suspended and may be resumed when there is no transaction on FlexSPI any more. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence suspended already, previous suspended sequence will be removed and never resumed. See [Command abort and suspend](#) for more details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATLFT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												BUFID		ACTIVE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 DATLFT	The Data size left for suspended command sequence (in byte).
15-4 —	Reserved
3-1 BUFID	AHB RX BUF ID for suspended command sequence.
0 ACTIVE	Indicates if an AHB read prefetch command sequence has been suspended. 0b - No suspended AHB read prefetch command. 1b - An AHB read prefetch command sequence has been suspended.

39.8.2.33 IP RX FIFO Status Register (IPRXFSTS)

Offset

Register	Offset
IPRXFSTS	F0h

Function

This status register indicates the status of IP RX FIFO.

Diagram

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

Fields

Field	Function
31-16 RDCNTR	Total Read Data Counter: RDCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP RX FIFO. Valid Data entries in IP RX FIFO is: FILL * 64 Bits.

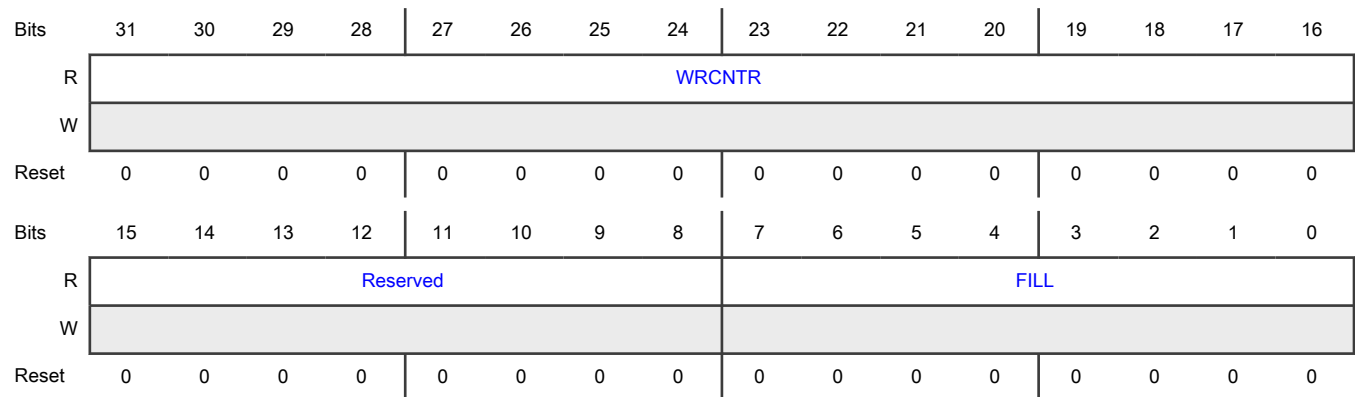
39.8.2.34 IP TX FIFO Status Register (IPTXFSTS)

Offset

Register	Offset
IPTXFSTS	F4h

Function

This status register indicates the status of IP TX FIFO.

Diagram**Fields**

Field	Function
31-16 WRCNTR	Total Write Data Counter: WRCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP TX FIFO. Valid Data entries in IP TX FIFO is: FILL * 64 Bits.

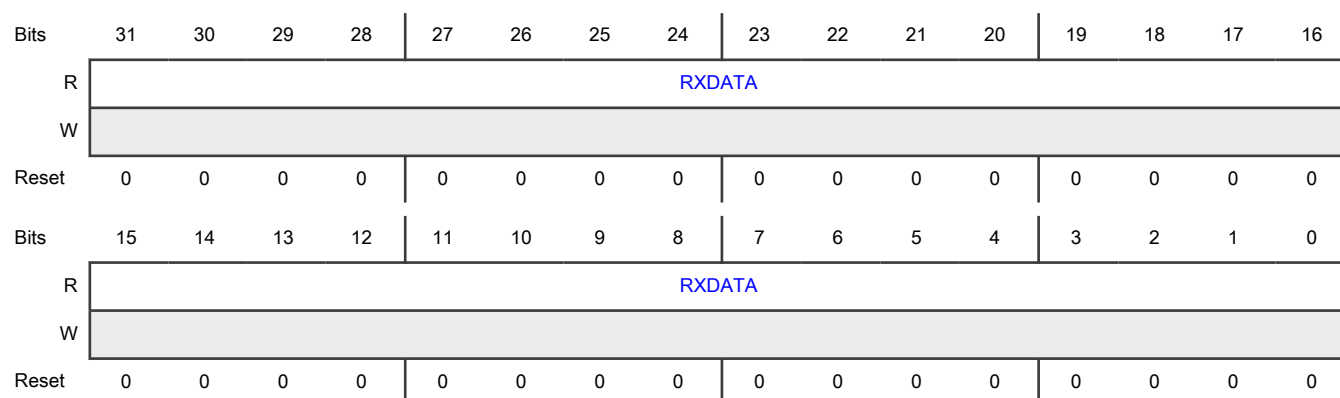
39.8.2.35 IP RX FIFO Data Register a (RFDR0 - RFDR31)**Offset**

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

Function

These registers provide read access to IP RX FIFO by IPS bus. The read value is unknown for read access to invalid entries in IP RX FIFO.

Diagram**Fields**

Field	Function
31-0 RXDATA	RX Data. For more details, see Reading data from IP RX FIFO

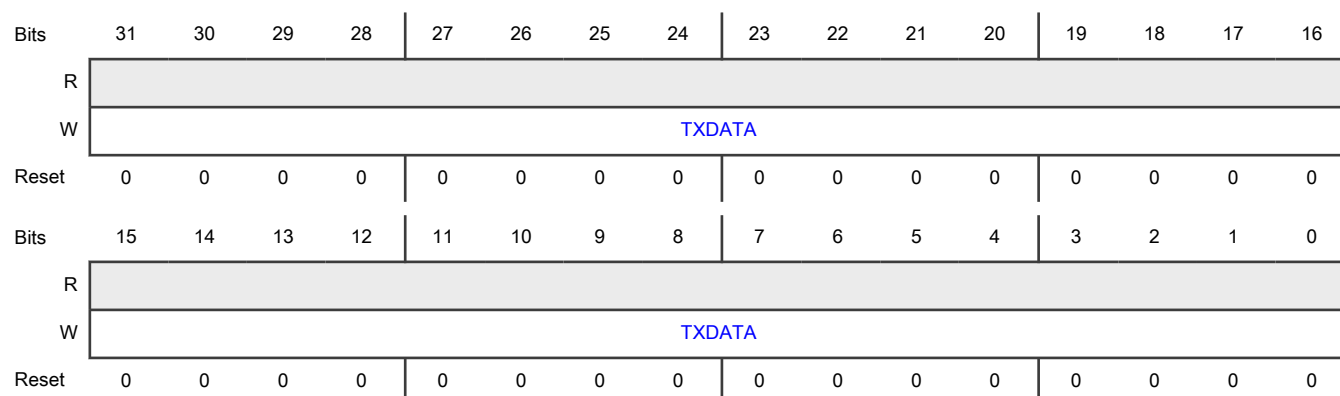
39.8.2.36 IP TX FIFO Data Register a (TFDR0 - TFDR31)**Offset**

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

Function

These registers provide write access to IP TX FIFO by IPS bus.

Diagram

Fields

Field	Function
31-0 TXDATA	TX Data For more details, see Filling data to IP TX FIFO

39.8.2.37 LUT a (LUT0 - LUT63)

Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

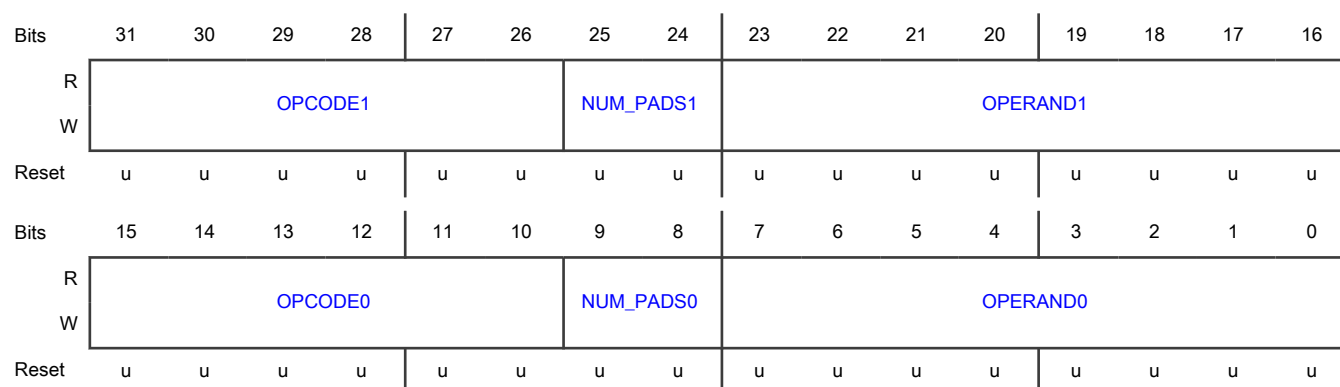
Function

The LUT is a look-up-table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI will fetch the command sequence from LUT when IP/AHB command triggered. There are 16 command sequences in LUT. See [LUT](#) and [Programmable sequence engine](#) for more details.

NOTE

LUT is implemented as memory, so the reset value is unknown.

Diagram



Fields

Field	Function
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 OPERAND1	OPERAND1
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0 OPERAND0	OPERAND0

39.8.2.38 HADDR REMAP START ADDR (HADDRSTART)

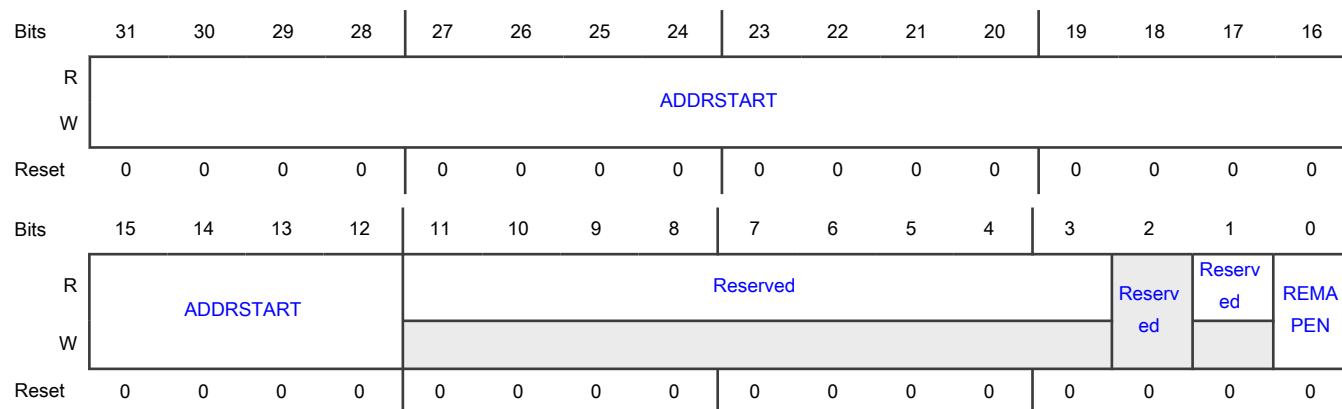
Offset

Register	Offset
HADDRSTART	420h

Function

The HADDRSTART register is used to configure options for the dual-image remap feature. See [Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers](#) section for more details.

Diagram



Fields

Field	Function
31-12	HADDR start address

Table continues on the next page...

Table continued from the previous page...

Field	Function
ADDRSTART	HADDR remap range's start address, 4K aligned NOTE When ADDRSTART setting is same as ASFM_BASE, and OTFAD keyblob function is enabled, keyblob will also be remapped. Remapping location is ASFM_BASE+HADDROFFSET.
11-3 —	Reserved
2 —	Reserved
1 —	Reserved
0 REMAPEN	AHB Bus address remap function enable 0b - HADDR REMAP Disabled 1b - HADDR REMAP Enabled

39.8.2.39 HADDR REMAP END ADDR (HADDREND)

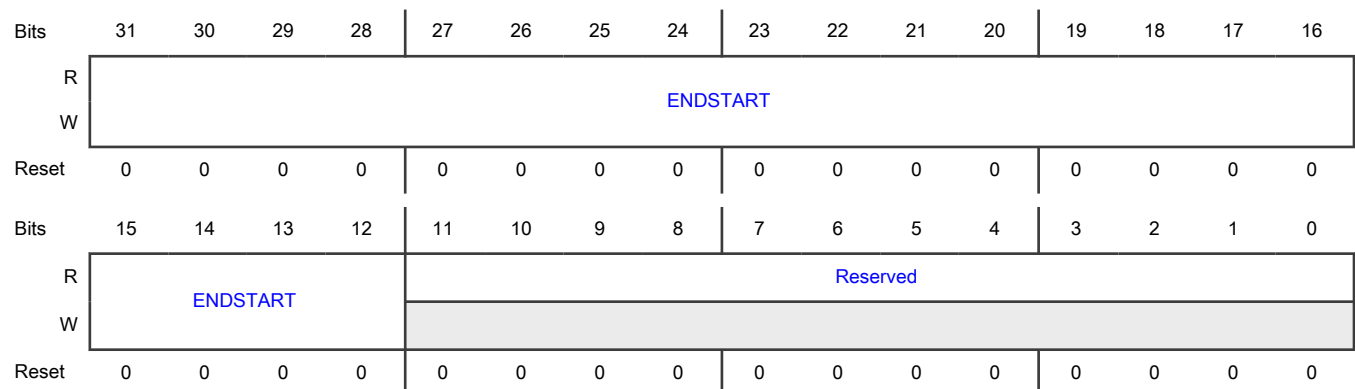
Offset

Register	Offset
HADDREND	424h

Function

The HADDREND register is used to configure options for the dual-image remap feature. See [Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers](#) section for more details.

Diagram



Fields

Field	Function
31-12 ENDSTART	HADDR remap range's end address, 4K aligned
11-0 —	Reserved

39.8.2.40 HADDR REMAP OFFSET (HADDROFFSET)

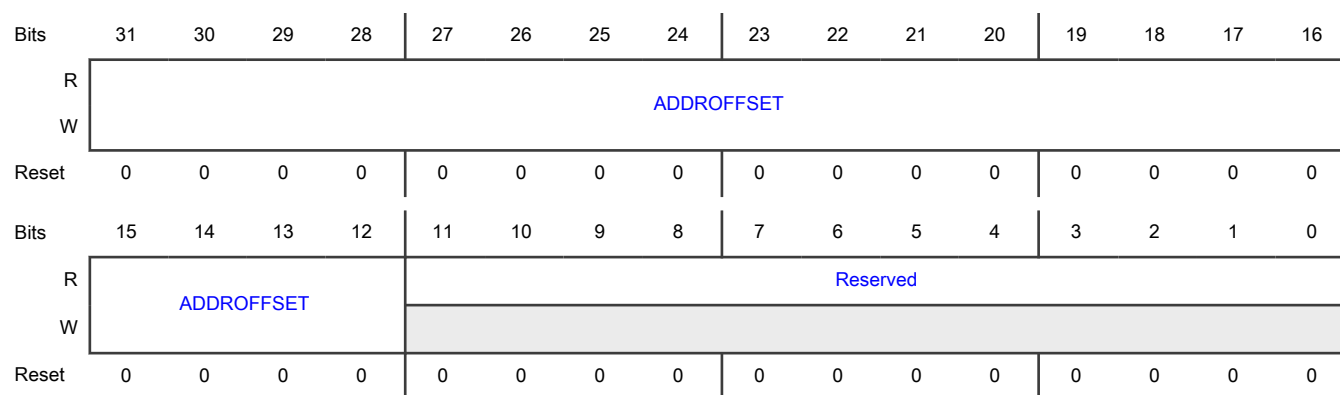
Offset

Register	Offset
HADDROFFSET	428h

Function

The HADDROFFSET register is used to configure options for the dual-image remap feature. See [Dual image use-case in using HADDRSTART, HADDREND, and HADDROFFSET registers](#) section for more details.

Diagram



Fields

Field	Function
31-12 ADDROFFSET	HADDR offset field, remapped address will be ADDR[31:12]=ADDR_original[31:12]+ADDROFFSET
11-0 —	Reserved

39.8.3 FlexSPI register descriptions

39.8.3.1 FlexSPI memory map

FLEXSPI1 base address: 4013_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register 0 (MCR0)	32	RW	FFFF_80C2h
4h	Module Control Register 1 (MCR1)	32	RW	FFFF_FFFFh
8h	Module Control Register 2 (MCR2)	32	RW	2000_81F7h
Ch	AHB Bus Control Register (AHBCR)	32	RW	0000_0018h
10h	Interrupt Enable Register (INTEN)	32	RW	0000_0000h
14h	Interrupt Register (INTR)	32	RW	0000_0000h
18h	LUT Key Register (LUTKEY)	32	RW	5AF0_5AF0h
1Ch	LUT Control Register (LUTCR)	32	RW	0000_0002h
20h	AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)	32	RW	8000_0020h
24h	AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)	32	RW	8001_0020h
28h	AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)	32	RW	8002_0020h
2Ch	AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)	32	RW	8003_0020h
30h	AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0)	32	RW	8004_0020h
34h	AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0)	32	RW	8005_0020h
38h	AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0)	32	RW	8006_0020h
3Ch	AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0)	32	RW	8007_0020h
60h	Flash Control Register 0 (FLSHA1CR0)	32	RW	0001_0000h
64h	Flash Control Register 0 (FLSHA2CR0)	32	RW	0001_0000h
68h	Flash Control Register 0 (FLSHB1CR0)	32	RW	0001_0000h
6Ch	Flash Control Register 0 (FLSHB2CR0)	32	RW	0001_0000h
70h	Flash Control Register 1 (FLSHA1CR1)	32	RW	0000_0063h
74h	Flash Control Register 1 (FLSHA2CR1)	32	RW	0000_0063h
78h	Flash Control Register 1 (FLSHB1CR1)	32	RW	0000_0063h
7Ch	Flash Control Register 1 (FLSHB2CR1)	32	RW	0000_0063h
80h	Flash Control Register 2 (FLSHA1CR2)	32	RW	0000_0000h
84h	Flash Control Register 2 (FLSHA2CR2)	32	RW	0000_0000h
88h	Flash Control Register 2 (FLSHB1CR2)	32	RW	0000_0000h
8Ch	Flash Control Register 2 (FLSHB2CR2)	32	RW	0000_0000h
94h	Flash Control Register 4 (FLSHCR4)	32	RW	0000_0000h
A0h	IP Control Register 0 (IPCR0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
A4h	IP Control Register 1 (IPCR1)	32	RW	0000_0000h
B0h	IP Command Register (IPCMD)	32	RW	0000_0000h
B4h	Data Learn Pattern Register (DLPR)	32	RW	0000_0000h
B8h	IP RX FIFO Control Register (IPRXFCR)	32	RW	0000_0000h
BCh	IP TX FIFO Control Register (IPTXFCR)	32	RW	0000_0000h
C0h	DLL Control Register 0 (DLLACR)	32	RW	0000_0100h
C4h	DLL Control Register 0 (DLLBCR)	32	RW	0000_0100h
E0h	Status Register 0 (STS0)	32	R	0000_0002h
E4h	Status Register 1 (STS1)	32	R	0000_0000h
E8h	Status Register 2 (STS2)	32	R	0100_0100h
ECh	AHB Suspend Status Register (AHBSPNDSTS)	32	R	0000_0000h
F0h	IP RX FIFO Status Register (IPRXFSTS)	32	R	0000_0000h
F4h	IP TX FIFO Status Register (IPTXFSTS)	32	R	0000_0000h
100h - 17Ch	IP RX FIFO Data Register a (RFDR0 - RFDR31)	32	R	0000_0000h
180h - 1FCh	IP TX FIFO Data Register a (TFDR0 - TFDR31)	32	W	0000_0000h
200h - 2FCh	LUT a (LUT0 - LUT63)	32	RW	See section

39.8.3.2 Module Control Register 0 (MCR0)

Offset

Register	Offset
MCR0	0h

Diagram



Fields

Field	Function
31-24 AHBGRANTWAIT	Timeout wait cycle for AHB command grant. If AHB Triggered Command is not granted by the arbitrator, it will timeout after AHBGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is IP triggered and the read/write data size is too large. When an AHB command grant timeout occurs, an interrupt is generated (INTR[AHBCMDGE]). When INTEN[AHBCMDGEEN] = 1, AHB command is ignored by the arbitrator. NOTE This field is for debug only, keep the default value! It is not allowed to set this field to value 0x0.
23-16 IPGRANTWAIT	Timeout wait cycle for IP command grant. If IP Triggered Command is not granted by the arbitrator, it will timeout after IPGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is AHB triggered and read/write data size is too large. When IP command grant timeout occurs, an interrupt is generated (INTR[IPCMDGE]). When INTEN[IPCMDGEEN] = 0x1, IP command is ignored by the arbitrator. NOTE This field is for debug only, keep the default value! It is not allowed to set this field to value 0x0.
15 LEARNEN	This bit is used to enable/disable the data learning feature. When data learning is disabled, the sampling clock phase 0 is always used for RX data sampling even if LEARN instruction is correctly executed. See Data learning feature for more details. 0b - Disable the data learning feature. 1b - Enable the data learning feature.
14 SCKFREERUNEN	This bit is used to enable SCLK output free-running. For FPGA applications, the external device may use SCLK as reference clock to its internal PLL. 0b - Disable SCLK output free-running. 1b - Enable SCLK output free-running.
13 —	Reserved
12 DOZEEN	Doze mode enable bit 0b - Doze mode support disabled. AHB clock and serial clock will not be gated off when there is doze mode request from system. 1b - Doze mode support enabled. AHB clock and serial clock will be gated off when there is doze mode request from system.
11 HSEN	Half Speed Serial Flash Access Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bit enables the divide by 2 of the clock to external serial flash devices (A_SCLK/B_SCLK) for all commands (for both SDR and DDR mode). FlexSPI need to be set into MDIS mode before changing value of HSEN. Otherwise, it is possible to cause issue on internal logic/state machine. 0b - Disable divide by 2 of serial flash clock for half clock frequency. 1b - Enable divide by 2 of serial flash clock for half clock frequency.
10-8 SERCLKDIV	Serial root clock The serial root clock could be divided inside FlexSPI . See Clocks chapter for more details on clocking. NOTE Do not modify this field while the FlexSPI is not active (MCR0[MDIS] = 0). 000b - Divided by 1 001b - Divided by 2 010b - Divided by 3 011b - Divided by 4 100b - Divided by 5 101b - Divided by 6 110b - Divided by 7 111b - Divided by 8
7 —	Reserved
6 —	Reserved
5-4 RXCLKSRC	Sample Clock source selection for Flash Reading See RX Clock Source Features for more details. 00b - Dummy Read strobe generated by FlexSPI Controller and loopback internally. 01b - Dummy Read strobe generated by FlexSPI Controller and loopback from DQS pad. 10b - Reserved 11b - Flash provided Read strobe and input from DQS pad
3-2 —	Reserved
1 MDIS	Module Disable When the module is disabled, AHB/serial clock will be gated off internally to save power. Only register access (except LUT/IP RX FIFO/IP TX FIFO) is allowed.

Table continues on the next page...

Table continued from the previous page...

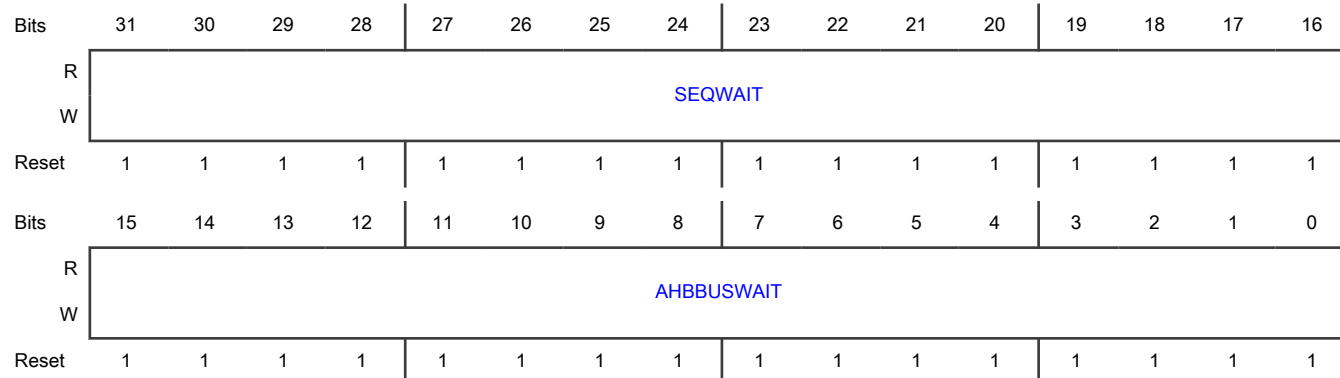
Field	Function
	0b - No impact 1b - Module disable
0 SWRESET	Software Reset This bit is auto-cleared by hardware after software reset is done. Configuration registers will not be reset. 0b - No impact 1b - Software reset

39.8.3.3 Module Control Register 1 (MCR1)

Offset

Register	Offset
MCR1	4h

Diagram



Fields

Field	Function
31-16 SEQWAIT	Command Sequence Execution will timeout and abort after SEQWAIT * 1024 Serial Root Clock cycles. When sequence execution timeout occurs, there will be an interrupt generated (INTR[SEQTIMEOUT]) if this interrupt is enabled (INTEN[SEQTIMEOUTEN] is set 0x1) and AHB command is ignored by arbitrator. NOTE It is not allowed to set this field to value 0x0.
15-0	AHB Bus wait

Table continues on the next page...

Table continued from the previous page...

Field	Function
AHBBUSWAIT	AHB Read/Write access to Serial Flash Memory space will timeout if data is not received from Flash or if data is not transmitted after AHBBUSWAIT * 1024 AHB clock cycles, AHB Bus will get an error response. When AHB bus timeout occurs, an interrupt is generated (INTR[AHBBUSTIMEOUT]). When (INTR[AHBBUSTIMEOUT]) = 0x1, AHB command is ignored by the arbitrator.
NOTE It is not allowed to set this field to value 0x0.	

39.8.3.4 Module Control Register 2 (MCR2)

Offset

Register	Offset
MCR2	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RESUMEWAIT								Reserved		Reserv ed	Reserv ed	Reserved	Reserved	Reserv ed	
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAME DEV...	0	Reserv ed	Reserv ed	CLRA HBB...	Reserved				Reserved						
W		CLRLE AR...														
Reset	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1

Fields

Field	Function
31-24 RESUMEWAIT	Wait cycle (in AHB clock cycle) for idle state before suspended command sequence resumed. See Command abort and suspend for more details.
23-21 —	Reserved
20 —	Reserved
19	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
18-17 —	Reserved
16 —	Reserved
15 SAMEDEVICEEN	All external devices are same devices (both in type and size) for A1/A2/B1/B2. 0b - In Individual mode, FLSHA1CRx/FLSHA2CRx/FLSHB1CRx/FLSHB2CRx register setting will be applied to Flash A1/A2/B1/B2 separately. In Parallel mode, FLSHA1CRx register setting will be applied to Flash A1 and B1, FLSHA2CRx register setting will be applied to Flash A2 and B2. FLSHB1CRx/FLSHB2CRx register setting will be ignored. 1b - FLSHA1CR0/FLSHA1CR1/FLSHA1CR2 register setting will be applied to Flash A1/A2/B1/B2. FLSHA2CRx/FLSHB1CRx/FLSHB2CRx will be ignored.
14 CLRLEARNPHASE	The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately. 0b - No impact 1b - The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately.
13 —	Reserved
12 —	Reserved
11 CLRAHBBUOPT	Clear AHB buffer This bit determines whether AHB RX Buffer and AHB TX Buffer will be cleared automatically when FlexSPI returns STOP mode ACK. Software should set this bit if AHB RX Buffer or AHB TX Buffer will be powered off in STOP mode. Otherwise AHB read access after exiting STOP mode may hit AHB RX Buffer or AHB TX Buffer but their data entries are invalid. 0b - AHB RX/TX Buffer will not be cleared automatically when FlexSPI returns Stop mode ACK. 1b - AHB RX/TX Buffer will be cleared automatically when FlexSPI returns Stop mode ACK.
10-9 —	Reserved
8-0 —	Reserved

39.8.3.5 AHB Bus Control Register (AHBCR)

Offset

Register	Offset
AHBCR	Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserved				Reserv	Reserv	Reserv	Reserv
W													ed	ed	ed	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	Reserved		Reserv	Reserv	READ	Reserv	Reserv	Reserv	READ	PREF	BUFF	CACH	CLRA	Reserv	APAR
W	ed			ed	ed	SZA...	ed	ed	ed	ADD...	ETC...	ERA...	ABL...	HBT...	ed	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Fields

Field	Function
31-27 —	Reserved
26-22 —	Reserved
21-20 —	Reserved
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14-13 —	Reserved
12 —	Reserved
11 —	Reserved
10 READSZALIGN	AHB Read Size Alignment 0b - AHB read size will be decided by other register setting like PREFETCH_EN 1b - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 READADDROP T	AHB Read Address option bit. This option bit is intended to remove AHB burst start address alignment limitation. When FlexSPI controller is used for FPGA application, it may be required for the FlexSPI to fetch exactly the byte number as AHB burst. In this case, FPGA device should be designed as non-word addressable and this option bit must be cleared. 0b - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is word-addressable. 1b - There is no AHB read burst start address alignment limitation. FlexSPI will fetch more data than AHB burst required to meet the alignment requirement.
5 PREFETCHEN	AHB Read Prefetch Enable. When AHB read prefetch is enabled, FlexSPI will fetch more flash read data than current AHB burst needed so that the read latency for next AHB read access will be reduced.
4 BUFFERABLEEN	Enable AHB bus bufferable write access support. This field affects the last beat of AHB write access, see AHB write access to Flash for more details. 0b - Disabled. For all AHB write accesses (bufferable or non-bufferable), FlexSPI will return AHB Bus ready after all data is transmitted to external device and AHB command finished.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Enabled. For AHB bufferable write access, FlexSPI will return AHB Bus ready when the AHB command is granted by arbitrator and will not wait for AHB command finished.
3 CACHABLEEN	Enable AHB bus cachable read access support. 0b - Disabled. When there is AHB bus cachable read access, FlexSPI will not check whether it hit AHB TX Buffer. 1b - Enabled. When there is AHB bus cachable read access, FlexSPI will check whether it hit AHB TX Buffer first.
2 CLRAHBTXBUF	Clear the status/pointers of AHB TX Buffer. Auto-cleared. 0b - No function. This bit will clear automatically when the clear operation completes. This bit always reads 0. 1b - Clear operation enable.
1 —	Reserved
0 APAREN	Parallel mode enabled for AHB triggered Command (both read and write). 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.

39.8.3.6 Interrupt Enable Register (INTEN)

Offset

Register	Offset
INTEN	10h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved		SEQTI ME...	AHBB UST...	SCKS TOP...	SCKS TOP...	DATA LEA...	IPTXW EEN	IPRX WAEN	AHBC MDE...	IPCM DER...	AHBC MDG...	IPCM DGE...	IPCM DDO...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUT EN	Sequence execution timeout interrupt enable. See Interrupts chapter for more details. 0b - Disable interrupt or no impact 1b - Enable interrupt
10 AHBBUSTIME OUTEN	AHB Bus timeout interrupt. See Interrupts chapter for more details. 0b - Disable interrupt or no impact 1b - Enable interrupt
9 SCKSTOPBYW REN	SCLK is stopped during command sequence because Async TX FIFO empty interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
8 SCKSTOPBYR DEN	SCLK is stopped during command sequence because Async RX FIFO full interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
7 DATALEARNF AILEN	Data Learning failed interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
6 IPTXWEEN	IP TX FIFO WaterMark empty interrupt enable. IP TX FIFO has more empty space than WaterMark level interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
5 IPRXWAEN	IP RX FIFO WaterMark available interrupt enable. IP RX FIFO has more valid data than WaterMark level interrupt enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disable interrupt or no impact 1b - Enable interrupt
4 AHBCMDERR EN	AHB triggered Command Sequences Error Detected interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
3 IPCMDDERREN	IP triggered Command Sequences Error Detected interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
2 AHBCMDGEEEN	AHB triggered Command Sequences Grant Timeout interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
1 IPCMDDGEEEN	IP triggered Command Sequences Grant Timeout interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt
0 IPCMDDONEEN	IP triggered Command Sequences Execution finished interrupt enable. 0b - Disable interrupt or no impact 1b - Enable interrupt

39.8.3.7 Interrupt Register (INTR)

Offset

Register	Offset
INTR	14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															Reserv ed
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved		SEQTI ME...	AHBB UST...	SCKS TOP...	SCKS TOP...	DATA LEA...	IPTXW E	IPRX WA	AHBC MDE...	IPCM DERR	AHBC MDGE	IPCM DGE	IPCM DDO...
W					W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUT	Sequence execution timeout interrupt.
10 AHBBUSTIMEOUT	AHB Bus timeout interrupt. See Interrupts chapter for more details.
9 SCKSTOPBYWR	SCLK is stopped during command sequence because Async TX FIFO empty interrupt.
8 SCKSTOPBYRD	SCLK is stopped during command sequence because Async RX FIFO full interrupt.
7 DATALEARNFAIL	Data Learning failed interrupt.
6 IPTXWE	IP TX FIFO watermark empty interrupt. IP TX FIFO has more empty space than WaterMark level interrupt.
5 IPRXWA	IP RX FIFO watermark available interrupt. IP RX FIFO has more valid data than WaterMark level interrupt.
4 AHBCMDERR	AHB triggered Command Sequences Error Detected interrupt. When an error detected for AHB command, this command will be ignored and not executed at all.
3 IPCMDERR	IP triggered Command Sequences Error Detected interrupt. When an error detected for IP command, this command will be ignored and not executed at all.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 AHBCMDGE	AHB triggered Command Sequences Grant Timeout interrupt.
1 IPCMDGE	IP triggered Command Sequences Grant Timeout interrupt.
0 IPCMDDONE	IP triggered Command Sequences Execution finished interrupt. This interrupt is also generated when there is IPCMDGE or IPCMDERR interrupt generated.

39.8.3.8 LUT Key Register (LUTKEY)

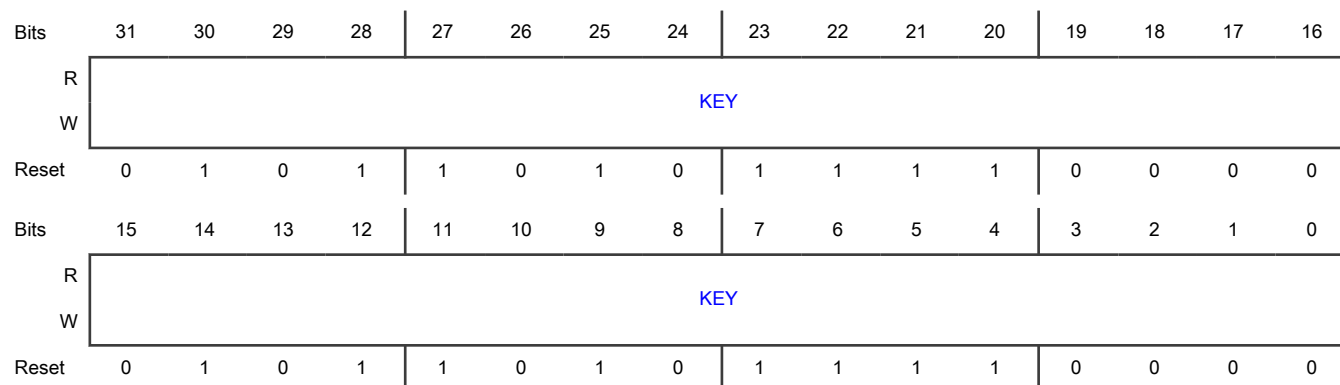
Offset

Register	Offset
LUTKEY	18h

Function

The LUT Key Register contains the key to lock and unlock LUT. See [LUT](#) for details.

Diagram



Fields

Field	Function
31-0 KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.

39.8.3.9 LUT Control Register (LUTCR)

Offset

Register	Offset
LUTCR	1Ch

Function

The LUT control register is used along with LUTKEY register to lock or unlock LUT. This register has to be written immediately after writing 0x5AF05AF0 to LUTKEY register for the lock or unlock operation to be successful. See [LUT](#) for details on locking/unlocking LUT. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												Reserv ed	UNLO CK	LOCK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-3 —	Reserved
2 —	Reserved
1 UNLOCK	Unlock LUT 0b - No impact 1b - Unlock LUT, the LUT can be written
0 LOCK	Lock LUT 0b - No impact 1b - Lock LUT, LUT will be locked and can't be written

39.8.3.10 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

Offset

Register	Offset
AHBRXBUF0CR0	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.

Table continues on the next page...

Table continued from the previous page...

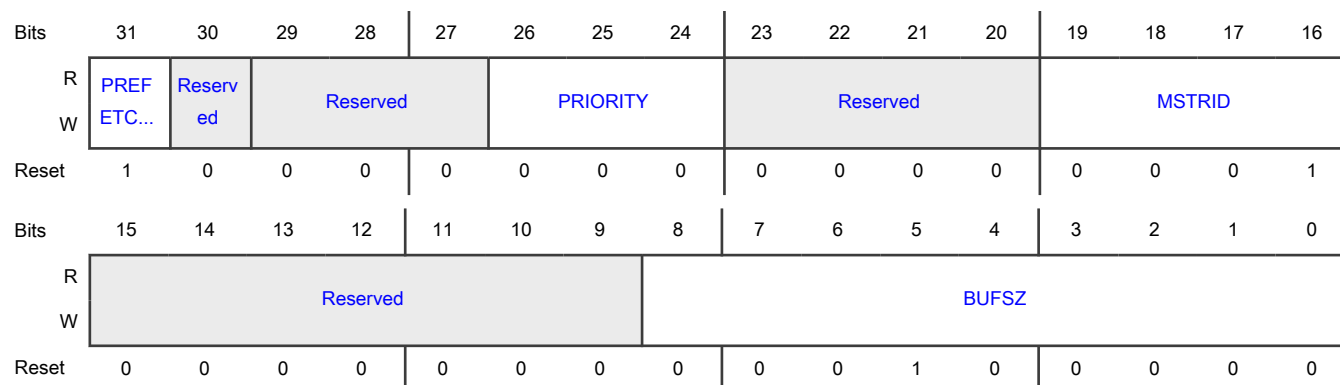
Field	Function
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.11 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

Offset

Register	Offset
AHBRXBUF1CR0	24h

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.12 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

Offset

Register	Offset
AHBRXBUF2CR0	28h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserved	Reserved		PRIORITY				Reserved				MSTRID			
W	ETC...															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

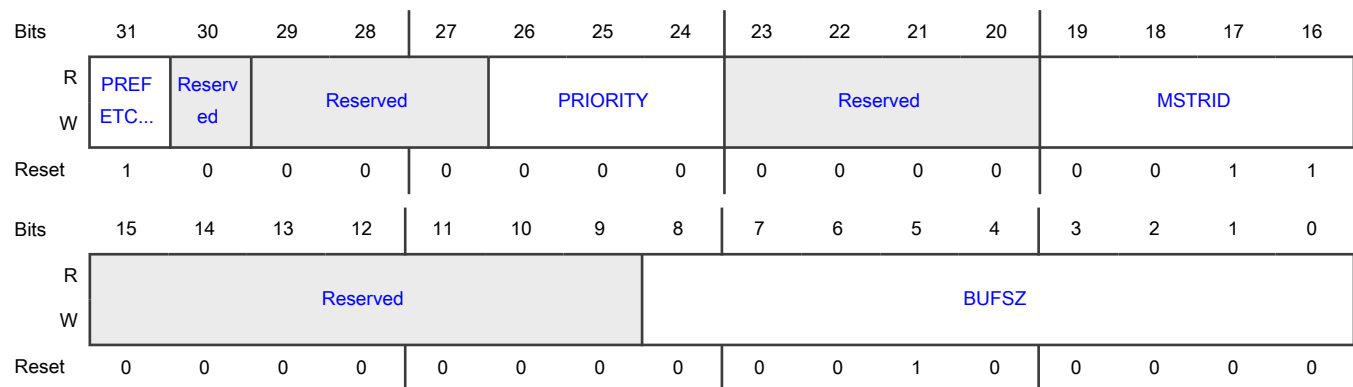
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.13 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

Offset

Register	Offset
AHBRXBUF3CR0	2Ch

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.14 AHB RX Buffer 4 Control Register 0 (AHBRXBUF4CR0)

Offset

Register	Offset
AHBRXBUF4CR0	30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PREF	Reserv	Reserved				PRIORITY				Reserved				MSTRID	
W	ETC...	ed														
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								BUFSZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.

Table continues on the next page...

Table continued from the previous page...

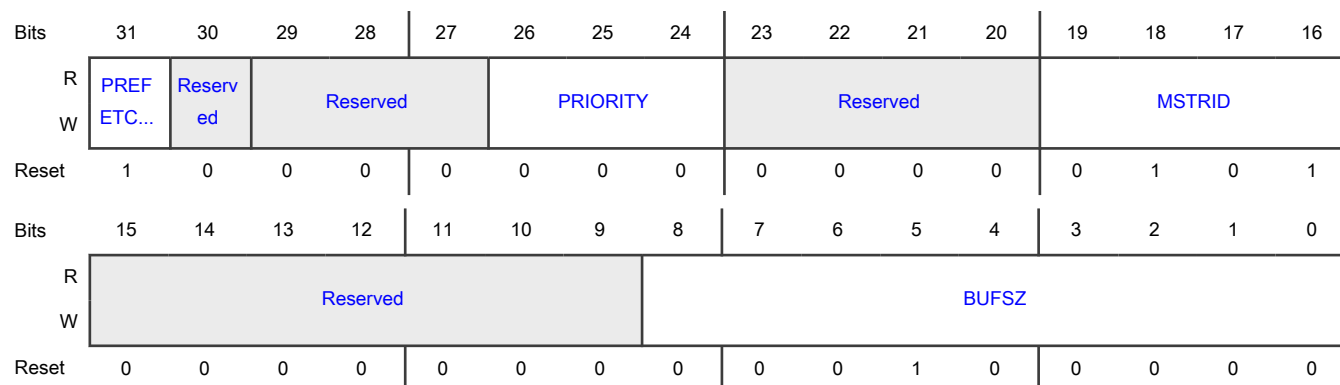
Field	Function
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.15 AHB RX Buffer 5 Control Register 0 (AHBRXBUF5CR0)

Offset

Register	Offset
AHBRXBUF5CR0	34h

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27	Reserved

Table continues on the next page...

Table continued from the previous page...

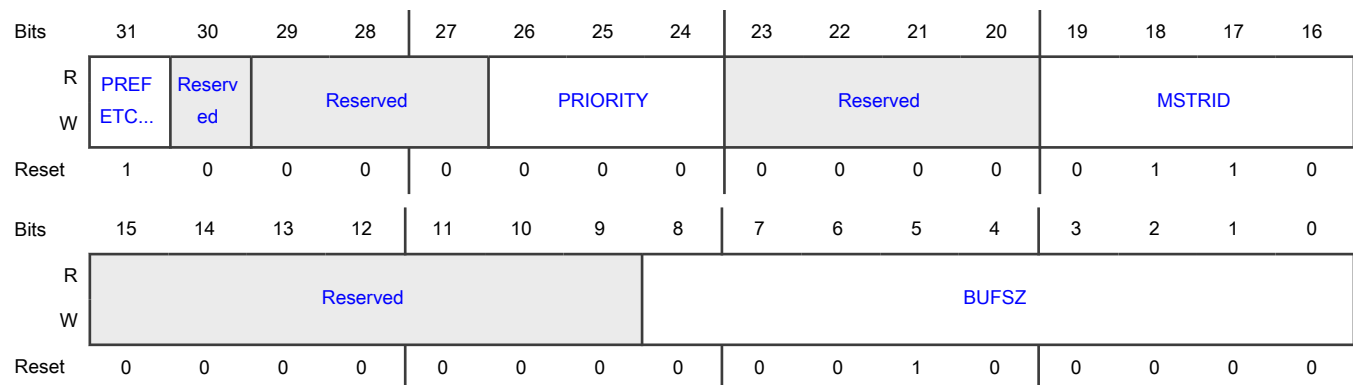
Field	Function
—	
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.16 AHB RX Buffer 6 Control Register 0 (AHBRXBUF6CR0)

Offset

Register	Offset
AHBRXBUF6CR0	38h

Diagram



Fields

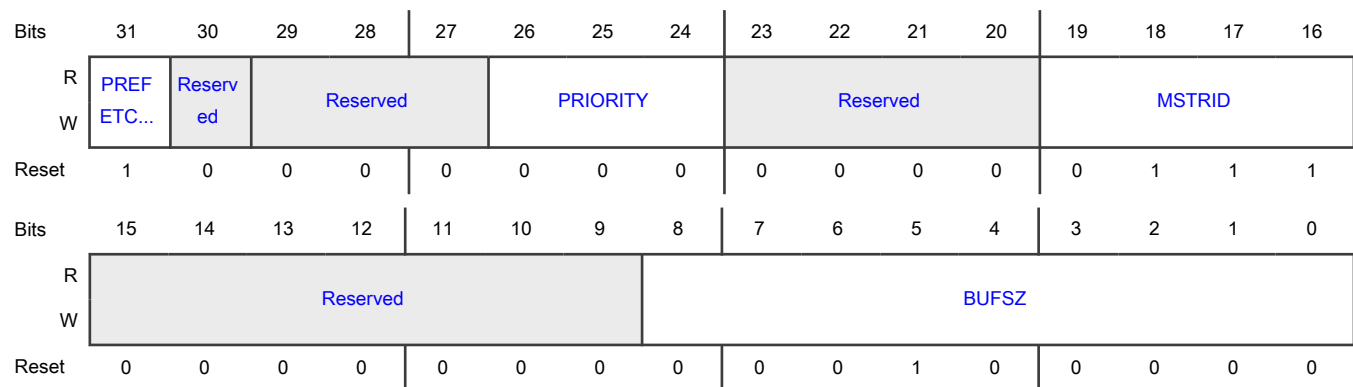
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.17 AHB RX Buffer 7 Control Register 0 (AHBRXBUF7CR0)

Offset

Register	Offset
AHBRXBUF7CR0	3Ch

Diagram



Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master. 0b - No prefetch 1b - Prefetch enable
30 —	Reserved
29-27 —	Reserved
26-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). See AHB RX Buffer Management for more details.
15-9 —	Reserved
8-0 BUFSZ	AHB RX Buffer Size in 64 bits. See AHB RX Buffer Management for more details.

39.8.3.18 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

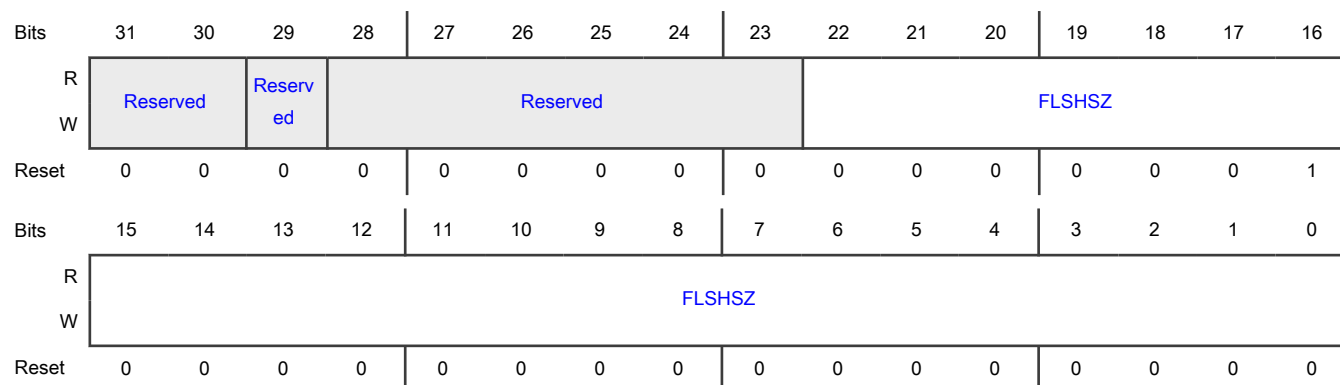
Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

Function

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Select).

Diagram



Fields

Field	Function
31-30 —	Reserved
29 —	Reserved
28-23 —	Reserved
22-0 FLSHSZ	Flash Size in KByte. Refer to the chip-specific section for the maximum flash size supported. When the FLSHSZ setting value is greater than the maximum size, the device flash size is interpreted as the maximum. The maximum flash size is the largest flash size supported for a single device, and also the maximum total flash size supported for all devices (up to 4). If the total flash size is larger than the maximum size, then only the maximum flash size address space is accessible.

39.8.3.19 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

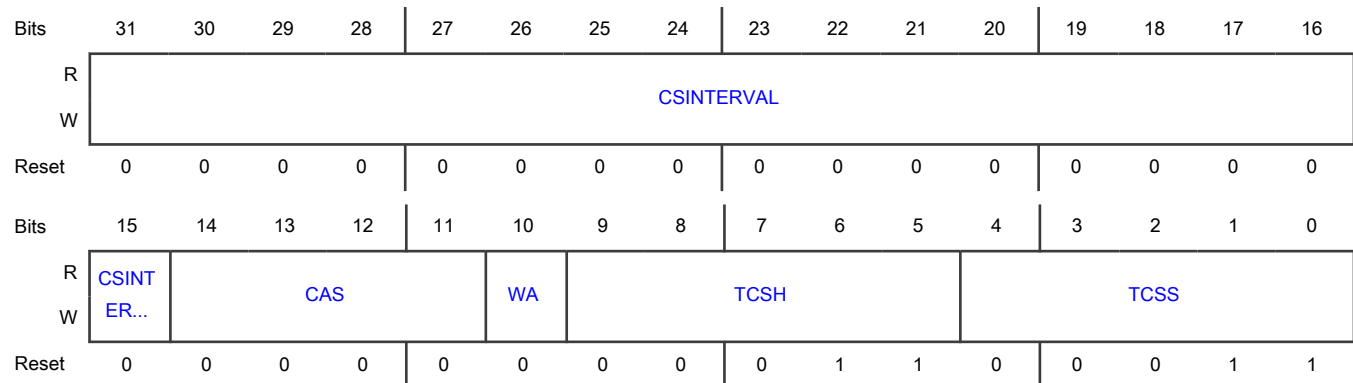
Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

Function

The Flash control register 1 contains the setting for FlexSPI to meet Flash device specific timings and Flash internal address space.

Diagram



Fields

Field	Function
31-16 CSINTERVAL	This field is used to set the minimum interval between flash device chip select deassertion and flash device chip select assertion. If external flash has a limitation on the interval between command sequences, this field should be set accordingly. If there is no limitation, set this field with value 0x0. When CSINTERVALUNIT is 0x0, the chip selecti invalid interval is: CSINTERVAL * 1 serial clock cycle; When CSINTERVALUNIT is 0x1, the chip Select invalid interval is: CSINTERVAL * 256 serial clock cycle. NOTE The chip Select interval is 2 cycle at least even if CSINTERVAL is less than 2.
15 CSINTERVALU NIT	CS interval unit 0b - The CS interval unit is 1 serial clock cycle 1b - The CS interval unit is 256 serial clock cycle
14-11	Column Address Size.

Table continues on the next page...

Table continued from the previous page...

Field	Function
CAS	When external flash has separate address field for row address and column address, this field should be set to flash column address bit width. FlexSPI will automatically split flash mapped address to Row address and Column address according to CAS field and WA field setting. This bit should be set to 0x0 when external Flash don't support column address. FlexSPI will transmit all flash address bits as Row address. For flash address mapping, see AHB access memory map for more detail.
10 WA	Word Addressable. This bit should be set when external Flash is word addressable. If Flash is word addressable, it should be access in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash. For flash address mapping, see AHB access memory map for more detail. 0b - This bit should be set as 0 when external Flash is byte addressable. 1b - This bit should be set as 1 when external Flash is word addressable. If Flash is word addressable, it should be accessed in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash.
9-5 TCSH	Serial Flash CS Hold time. This field is used to meet flash TCSH timing requirement. Serial flash CS Hold time promised by FlexSPI is: (TCSH + 1/2) serial clock cycles for DDR mode, and TCSH serial clock cycles for SDR mode. See Output timing between chip select and SCLK for more detail.
4-0 TCSS	Serial Flash CS setup time. This field is used to meet flash TCSS timing requirement. Serial flash CS Setup time promised by FlexSPI is: (TCSS + 1/2) serial root clock cycles (for both SDR and DDR mode). See Output timing between chip select and SCLK for more details.

39.8.3.20 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

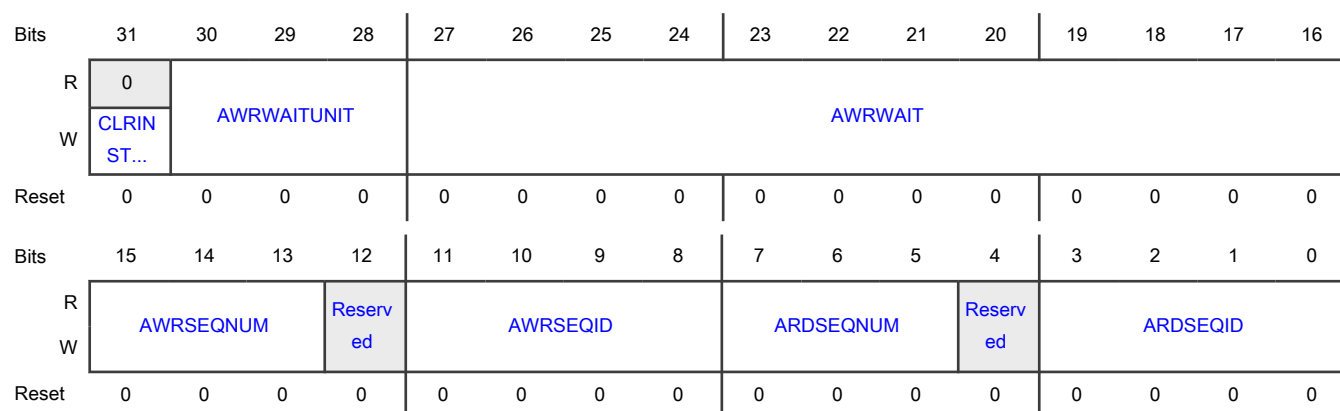
Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

Function

Flash Control Register 2 contains setting field for AHB Bus access configuration. If the 4 external device are in different types, AHB read/write command may use different command sequences and AHB bus ready wait time may be also different.

Diagram



Fields

Field	Function
31 CLRINSTRPTR	Clear the instruction pointer which is internally saved pointer by JMP_ON_CS. See Programmable Sequence Engine for details. This field is used for AHB Read access to external Flash supporting XIP (Execute-In-Place) mode.
30-28 AWRWAITUNIT	AWRWAIT unit 000b - The AWRWAIT unit is 2 AHB clock cycle 001b - The AWRWAIT unit is 8 AHB clock cycle 010b - The AWRWAIT unit is 32 AHB clock cycle 011b - The AWRWAIT unit is 128 AHB clock cycle 100b - The AWRWAIT unit is 512 AHB clock cycle 101b - The AWRWAIT unit is 2048 AHB clock cycle 110b - The AWRWAIT unit is 8192 AHB clock cycle 111b - The AWRWAIT unit is 32768 AHB clock cycle
27-16 AWRWAIT	For certain devices (such as FPGA), it need some time to write data into internal memory after the command sequences finished on FlexSPI interface. If another Read command sequence comes before previous programming finished internally, the read data may be wrong. This field is used to hold AHB Bus ready for AHB write access to wait the programming finished in external device. Then there will be no AHB read command triggered before the programming finished in external device. The Wait cycle between AHB triggered command sequences finished on FlexSPI interface and AHB return Bus ready: AWRWAIT * AWRWAITUNIT
15-13 AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices (e.g. HyperFlash/HyperRam/Serial NAND flash), a Flash programming access is done by several command sequences. AHB write Command will trigger (AWRSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE • Software should make sure the last sequence index never exceed LUT sequence numbers: $AWRSEQID + AWRSEQNUM < 16$ • Software needs to make sure that the fields AWRSEQNUM and LUT are configured correctly according to external device spec. FlexSPI doesn't check the sequence and just executes them one by one.
12 —	Reserved
11-8 AWRSEQID	Sequence Index for AHB Write triggered Command.
7-5 ARDSEQNUM	Sequence Number for AHB Read triggered Command in LUT. For certain flash devices (e.g. HyperFlash/HyperRam/Serial NAND flash), a Flash read access is done by several command sequences. AHB read Command will trigger (ARDSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. NOTE • Software should make sure the last sequence index never exceed LUT sequence numbers: $ARDSEQID + ARDSEQNUM \leq 16$ • Software needs to make sure that the fields ARDSEQNUM and LUT are configured correctly according to external device spec. FlexSPI does not check the sequence and just executes them one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read triggered Command in LUT.

39.8.3.21 Flash Control Register 4 (FLSHCR4)

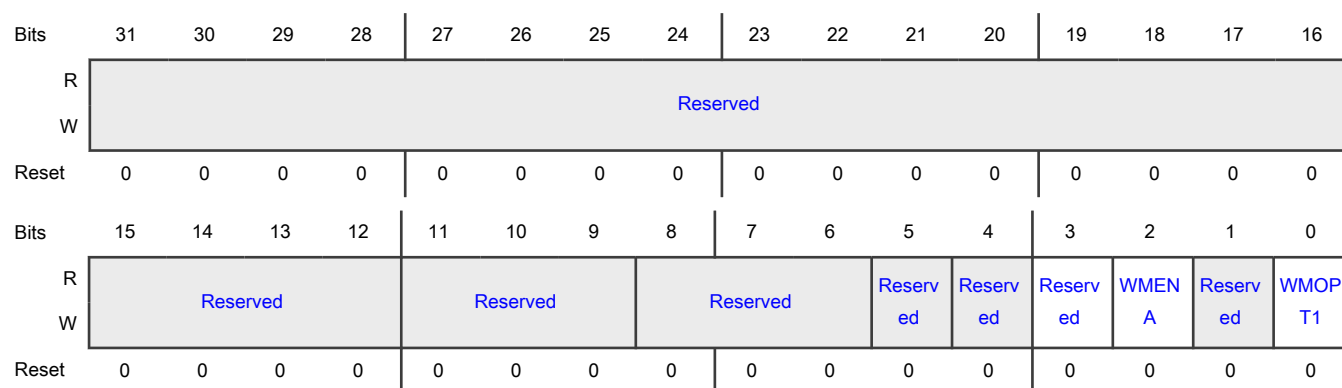
Offset

Register	Offset
FLSHCR4	94h

Function

The flash control register 4 provides the configuration for all external devices.

Diagram



Fields

Field	Function
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 WMENA	Write mask enable bit for flash device on port A. When write mask function is needed for memory device on port A, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will not be driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
1 —	Reserved
0 WMOPT1	Write mask option bit 1. This option bit could be used to remove AHB and IP write burst start address alignment limitation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - DQS pin will be used as Write Mask when writing to external device. There is no limitation on AHB/IP write burst start address alignment when flash is accessed in individual mode.
	1b - DQS pin will not be used as Write Mask when writing to external device. There is limitation on AHB/IP write burst start address alignment when flash is accessed in individual mode.

39.8.3.22 IP Control Register 0 (IPCR0)

Offset

Register	Offset
IPCR0	A0h

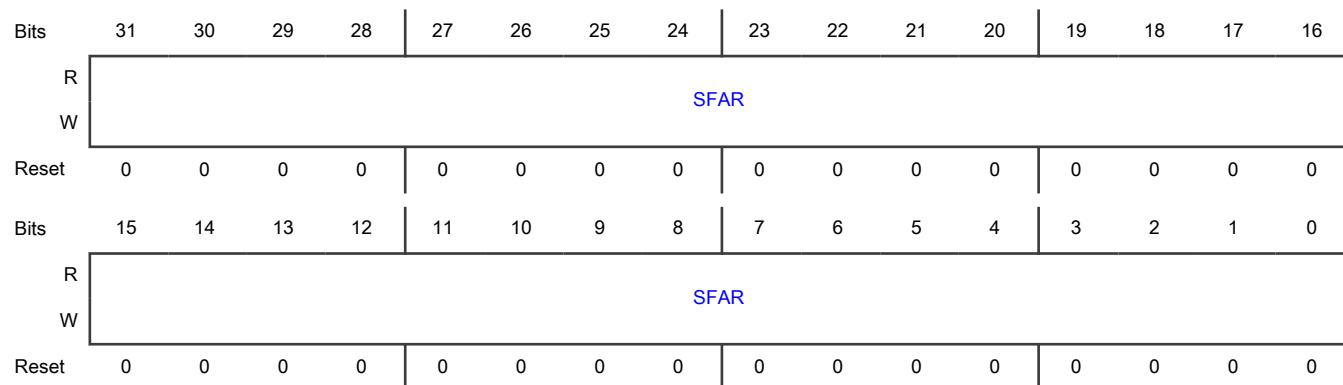
Function

The IP control registers provide all the configuration required for IP commands. This register provides the flash device's start address, instead of SoC address, to be accessed for IP command. FlexSPI will determine the chip select automatically according to this start address.

NOTE

- It's not allowed to issue IP command crossing Flash device boundaries. Otherwise there will be IPCMDERR interrupt generated.
- This register should be set before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

Diagram



Fields

Field	Function
31-0	Serial Flash Address for IP command.
SFAR	For IP command, the address in IPCR0[SFAR] should be the flash device's address without base address.

39.8.3.23 IP Control Register 1 (IPCR1)

Offset

Register	Offset
IPCR1	A4h

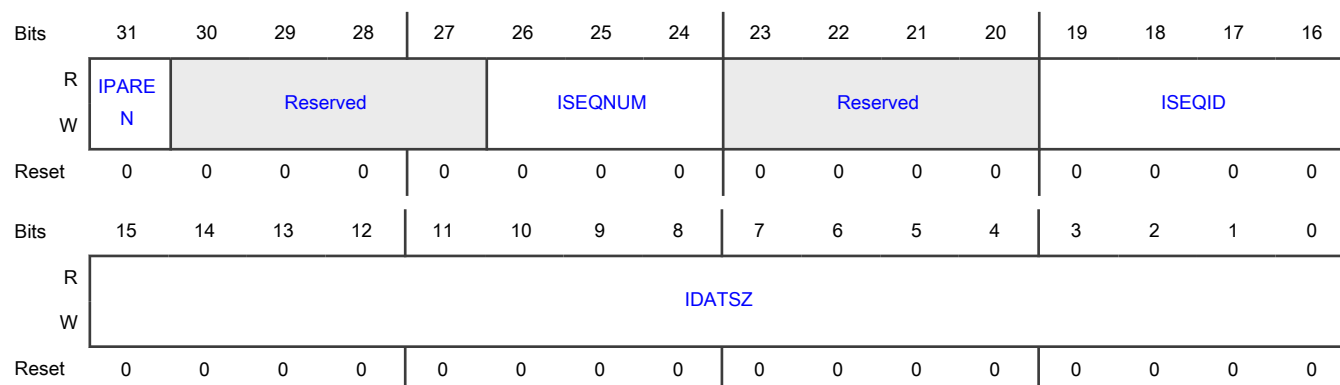
Function

The IP control registers provide all the configuration required for IP command. This register provides the flash read/program data size, sequence index in LUT, sequence number and individual/parallel mode setting for IP command.

NOTE

- This register should be written before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

Diagram



Fields

Field	Function
31 IPAREN	Parallel mode Enabled for IP command. 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.
30-27 —	Reserved
26-24 ISEQNUM	Sequence Number for IP command: ISEQNUM+1.
23-20 —	Reserved
19-16	Sequence Index in LUT for IP command.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ISEQID	
15-0 IDATSZ	Flash Read/Program Data Size (in Bytes) for IP command.

39.8.3.24 IP Command Register (IPCMD)

Offset

Register	Offset
IPCMD	B0h

Function

This register is used to trigger an IP command to access external flash device. IP command will be executed on FlexSPI interface after granted by arbitrator.

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 TRG	Setting this bit will trigger an IP Command. NOTE It's not allowed to trigger another IP command before previous IP command is finished on FlexSPI interface. Software need to poll register bit INTR[IPCMDDONE] or wait for this interrupt in order to wait for IP command finished.

39.8.3.25 Data Learn Pattern Register (DLPR)

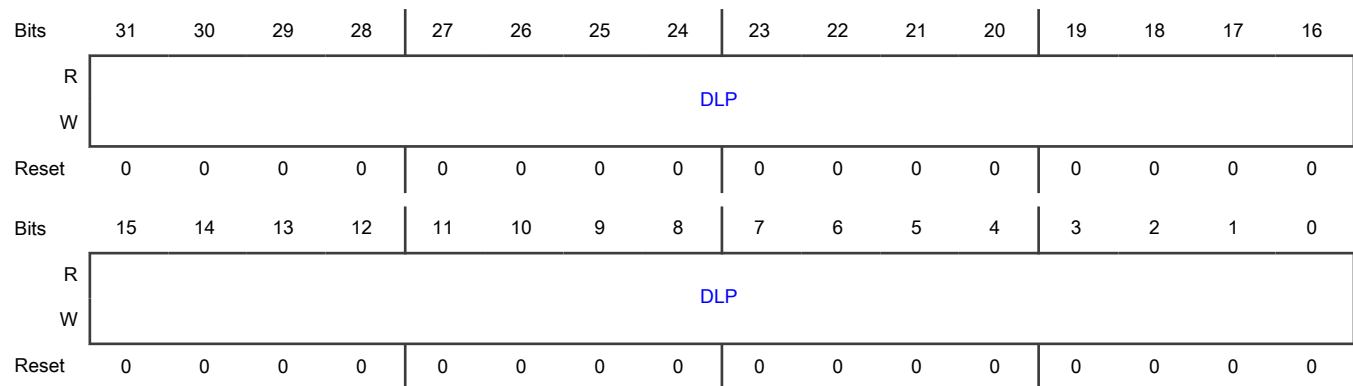
Offset

Register	Offset
DLPR	B4h

Function

This register provides the pattern to be used during Data Learning in FlexSPI.

Diagram



Fields

Field	Function
31-0	Data Learning Pattern.
DLP	The data learning pattern bit number is determined by operand in LEARN_SDR/LEARN_DDR Instruction Code. Data learning pattern bit number will be never more than 32 bits. If the operand in Instruction code is more than 32, 32 bits pattern will be used. If data learning pattern bit number is less than 32 bits, the lower pattern bits will be used. For more details, See Data learning feature .

39.8.3.26 IP RX FIFO Control Register (IPRXFCR)

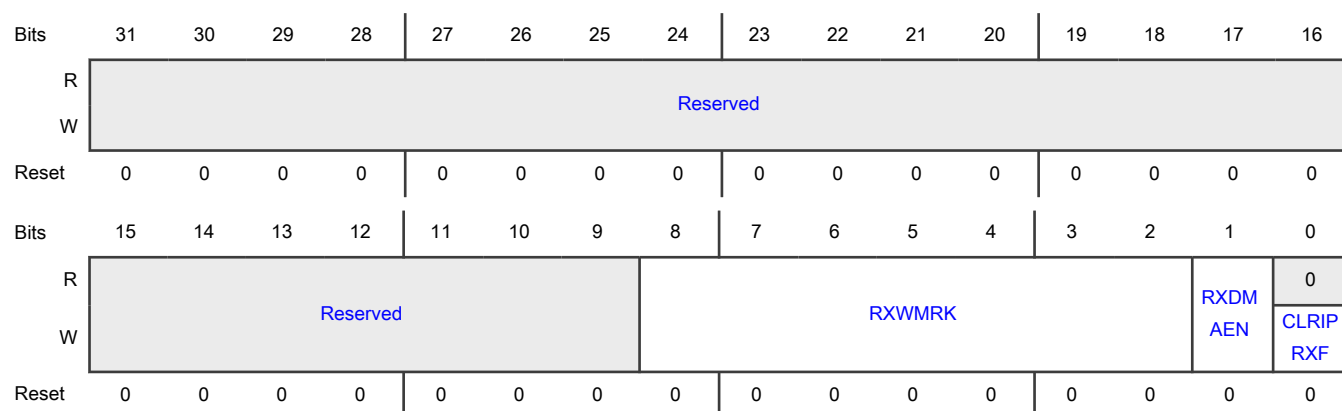
Offset

Register	Offset
IPRXFCR	B8h

Function

This register provides the configuration fields for IP RX FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 RXWMRK	Watermark level is $(RXWMRK+1)*64$ bits. INTR[IPRXWA] is set when filling level in IP RX FIFO is more than Watermark level by FlexSPI. There will be a DMA request when the filling level is more than Watermark level and DMA read is enabled (register bit RXDMAEN is set). There will be an IPRXWA (IP RX FIFO watermark available) interrupt generated when the filling level is more than Watermark level and IPRXWA interrupt is enabled (register bit INTEN[IPRXWAEN] is set). NOTE After write-1-clear to INTR[IPRXWA], read address should be rolled back to start address (memory mapped).
1 RXDMAEN	IP RX FIFO reading by DMA enabled. 0b - IP RX FIFO would be read by processor. 1b - IP RX FIFO would be read by DMA.
0 CLRIPRXF	Clear all valid data entries in IP RX FIFO. The read/write pointers in IP RX FIFO will be reset. 0b - No function. 1b - A clock cycle pulse to clear all valid data entries in IP RX FIFO.

39.8.3.27 IP TX FIFO Control Register (IPTXFCR)

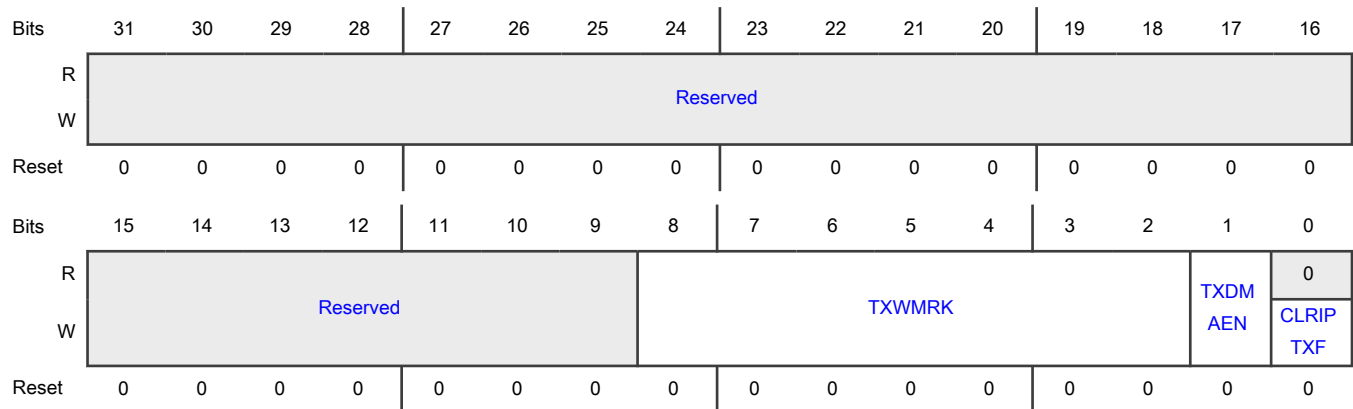
Offset

Register	Offset
IPTXFCR	BCh

Function

This register provides the configuration fields for IP TX FIFO management.

Diagram



Fields

Field	Function
31-9 —	Reserved
8-2 TXWMRK	Watermark level is $(TXWMRK+1)*64$ Bits. Interrupt register bit INTR[IPTXWE] is set when empty level in IP TX FIFO is more than Watermark level by FlexSPI. There will be a DMA request when empty level is more than Watermark level and DMA filling is enable (register bit TXDMAEN is set). There will be an IPTXWE (IP TX FIFO Watermark Empty) interrupt generated when empty level is more than Watermark level and IPTXWE interrupt is enable (register bit INTEN[IPTXWEEN] is set). NOTE - The watermark level should be no more than the write window. - The watermark level should be no more than IP TX FIFO size. - The write address to IP RX FIFO should roll back to the start address of write window after pushing IP TX FIFO by writing-one-clear to INTR[IPTXWE].
1 TXDMAEN	IP TX FIFO filling by DMA enabled. 0b - IP TX FIFO would be filled by processor. 1b - IP TX FIFO would be filled by DMA.
0 CLRIP TXF	Clear all valid data entries in IP TX FIFO. The read/write pointers in IP TX FIFO will be reset. 0b - No function. 1b - A clock cycle pulse to clear all valid data entries in IP TX FIFO.

39.8.3.28 DLL Control Register 0 (DLLACR - DLLBCR)

Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

Function

This register provides the configuration fields for Flash A/B sample clock DLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	OVRDVAL						OVRDEN	Reserved	SLVDLYTARGET				Reserved	DLLRESET	DLLLEN
W	Reserved							OVRDEN	Reserved					Reserved	DLLRESET	DLLLEN
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1. See DLL configuration for sampling for more details.
8 OVRDEN	Slave clock delay line delay cell number selection override enable. 0b - Slave clock delay line delay cell number selection override is disabled. 1b - Slave clock delay line delay cell number selection override is enabled.
7 —	Reserved
6-3 SLVDLYTARGET	The delay target for slave delay line is: $((\text{SLVDLYTARGET}+1) * 1/32 * \text{clock cycle of reference clock (serial root clock)})$. If serial root clock is ≥ 100 MHz, DLLLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0 is recommended.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	Reserved
1 DLLRESET	DLL reset Software could force a reset on DLL by setting this field to 0x1. This will cause the DLL to lose lock and re-calibrate to detect a ref_clock half period phase shift. The reset action is edge triggered, so the software needs to clear this bit after setting this bit (no delay limitation). 0b - No function. 1b - Software could force a reset on DLL by setting this field to 0x1.
0 DLLEN	DLL calibration enable. When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored. 0b - DLL calibration is disabled 1b - DLL calibration is enabled

39.8.3.29 Status Register 0 (STS0)

Offset

Register	Offset
STS0	E0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Reserved				DATALEARNPHASEA				ARBCMDSRC		ARBID LE	SEQID LE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Function
31-12 —	Reserved
11-8 —	Reserved
7-4 DATALEARNP HASEA	Indicate the sampling clock phase selection on Port A after Data Learning. There are 16 clock phases for sampling clock which is generated by delay cell line. When data learning feature is not enabled, the default clock phase 0 will be used to sample Flash read data. When data learning feature is enabled and LEARN_SDR/LEARN_DDR instruction executed correctly on FlexSPI Interface, FlexSPI will determine the correct clock phase to sample Flash read data. See Data Learning Feature for more details.
3-2 ARBCMDSRC	This status field indicates the trigger source of current command sequence granted by arbitrator. This field value is meaningless when ARB_CTL is not busy (STS0[ARBIDLE]=0x1). 00b - Triggered by AHB read command. 01b - Triggered by AHB write command. 10b - Triggered by IP command (triggered by setting register bit IPCMD[TRG]). 11b - Triggered by suspended command (resumed).
1 ARBIDLE	This status bit indicates the state machine in ARB_CTL is busy and there is command sequence granted by arbitrator and not finished yet on FlexSPI interface. When ARB_CTL state (ARBIDLE=0x1) is idle, there will be no transaction on FlexSPI interface also (SEQIDLE=0x1). So this bit should be polled to wait for FlexSPI controller become idle instead of SEQIDLE.
0 SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface. 0b - State machine in SEQ_CTL is not idle. 1b - State machine in SEQ_CTL is idle.

39.8.3.30 Status Register 1 (STS1)

Offset

Register	Offset
STS1	E4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 IPCMDERRCODE	Indicates the Error Code when IP command Error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - IP command with JMP_ON_CS instruction used in the sequence. 0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 0110b - Flash access start address exceed the whole flash address range (A1/A2/B1/B2). 1110b - Sequence execution timeout. 1111b - Flash boundary crossed.
23-20 —	Reserved
19-16 IPCMDERRID	Indicates the sequence Index when IP command error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c).
15-12 —	Reserved
11-8 AHBCMDERRCODE	Indicates the Error Code when AHB command Error detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - AHB Write command with JMP_ON_CS instruction used in the sequence.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 1110b - Sequence execution timeout.
7-4 —	Reserved
3-0 AHBCMDERRI D	Indicates the sequence index when an AHB command error is detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c).

39.8.3.31 Status Register 2 (STS2)

Offset

Register	Offset
STS2	E8h

Function

This register indicates the status of Flash A and B sample clock DLLs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				BREFSEL				BSLVSEL				BREFL OCK		BSLVL OCK	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AREFSEL				ASLVSEL				AREFL OCK		ASLVL OCK	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
29-24 BREFSEL	Flash B sample clock reference delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
23-18 BSLVSEL	Flash B sample clock slave delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
17 BREFLOCK	Flash B sample clock reference delay line locked. 0b - Flash B sample clock reference delay line is not locked. 1b - Flash B sample clock reference delay line is locked.
16 BSLVLOCK	Flash B sample clock slave delay line locked. 0b - Flash B sample clock slave delay line is not locked. 1b - Flash B sample clock slave delay line is locked.
15-14 —	Reserved
13-8 AREFSEL	Flash A sample clock reference delay line delay cell number selection. There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
7-2 ASLVSEL	Flash A sample clock slave delay line delay cell number selection . There are 0-63 phases. • 6'b000001 - Indicates lock phase 0. • 6'b100000 - Indicates lock phase 63.
1 AREFLOCK	Flash A sample clock reference delay line locked. 0b - Flash A sample clock reference delay line is not locked 1b - Flash A sample clock reference delay line is locked
0 ASLVLOCK	Flash A sample clock slave delay line locked. 0b - Flash A sample clock slave delay line is not locked 1b - Flash A sample clock slave delay line is locked

39.8.3.32 AHB Suspend Status Register (AHBSPNDSTS)

Offset

Register	Offset
AHBSPNDSTS	ECh

Function

Indicates the status of Suspended AHB Read Prefetch command sequence. When there is IP/AHB command triggered and arbitrator is processing an AHB Read sequence (prefetching more data not for current AHB burst), the prefetch sequence will be suspended and may be resumed when there is no transaction on FlexSPI any more. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence suspended already, previous suspended sequence will be removed and never resumed. See [Command abort and suspend](#) for more details.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATLFT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												BUFID		ACTIVE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 DATLFT	The Data size left for suspended command sequence (in byte).
15-4 —	Reserved
3-1 BUFID	AHB RX BUF ID for suspended command sequence.
0 ACTIVE	Indicates if an AHB read prefetch command sequence has been suspended. 0b - No suspended AHB read prefetch command. 1b - An AHB read prefetch command sequence has been suspended.

39.8.3.33 IP RX FIFO Status Register (IPRXFSTS)

Offset

Register	Offset
IPRXFSTS	F0h

Function

This status register indicates the status of IP RX FIFO.

Diagram

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

Fields

Field	Function
31-16 RDCNTR	Total Read Data Counter: RDCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP RX FIFO. Valid Data entries in IP RX FIFO is: FILL * 64 Bits.

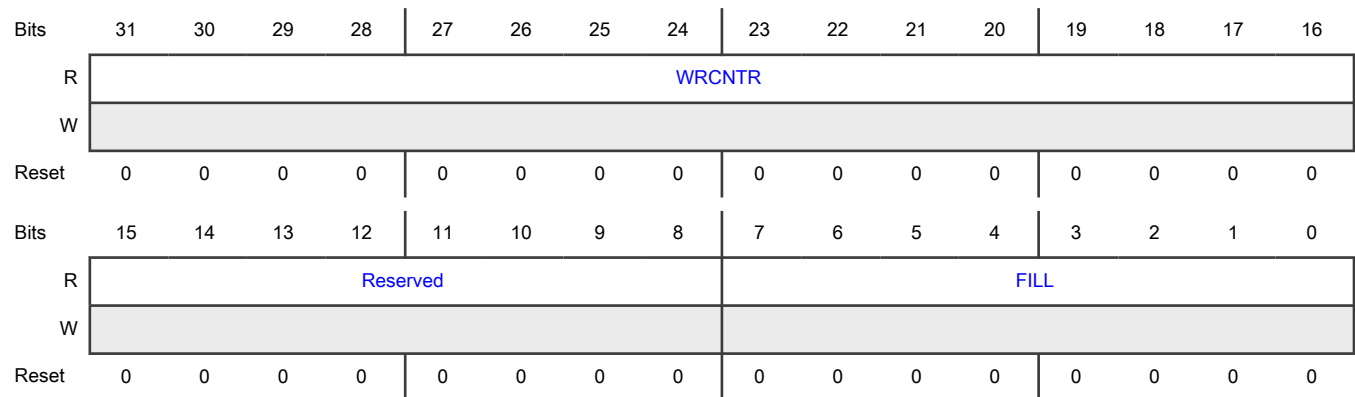
39.8.3.34 IP TX FIFO Status Register (IPTXFSTS)

Offset

Register	Offset
IPTXFSTS	F4h

Function

This status register indicates the status of IP TX FIFO.

Diagram**Fields**

Field	Function
31-16 WRCNTR	Total Write Data Counter: WRCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP TX FIFO. Valid Data entries in IP TX FIFO is: FILL * 64 Bits.

39.8.3.35 IP RX FIFO Data Register a (RFDR0 - RFDR31)**Offset**

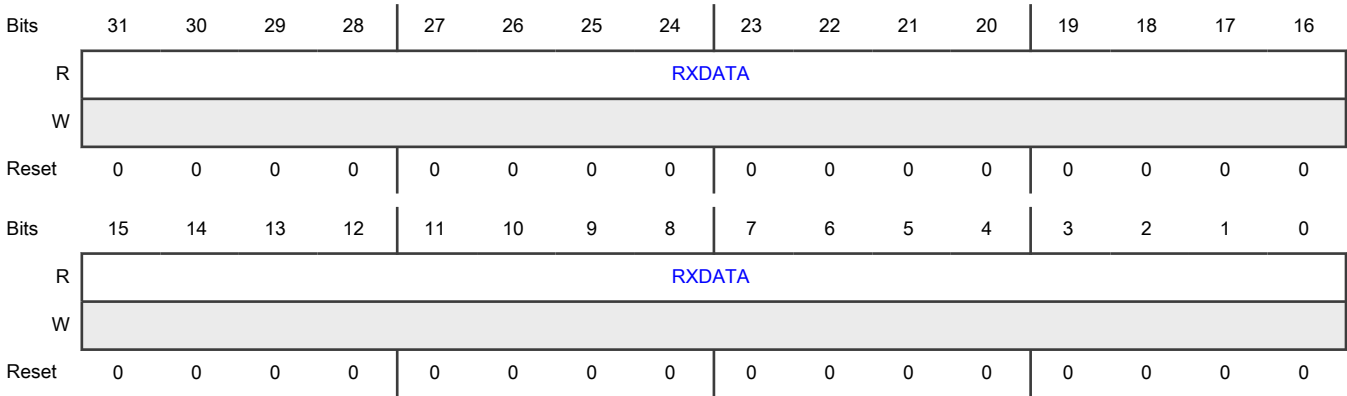
For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

Function

These registers provide read access to IP RX FIFO by IPS bus. The read value is unknown for read access to invalid entries in IP RX FIFO.

Diagram



Fields

Field	Function
31-0 RXDATA	RX Data. For more details, see Reading data from IP RX FIFO

39.8.3.36 IP TX FIFO Data Register a (TFDR0 - TFDR31)

Offset

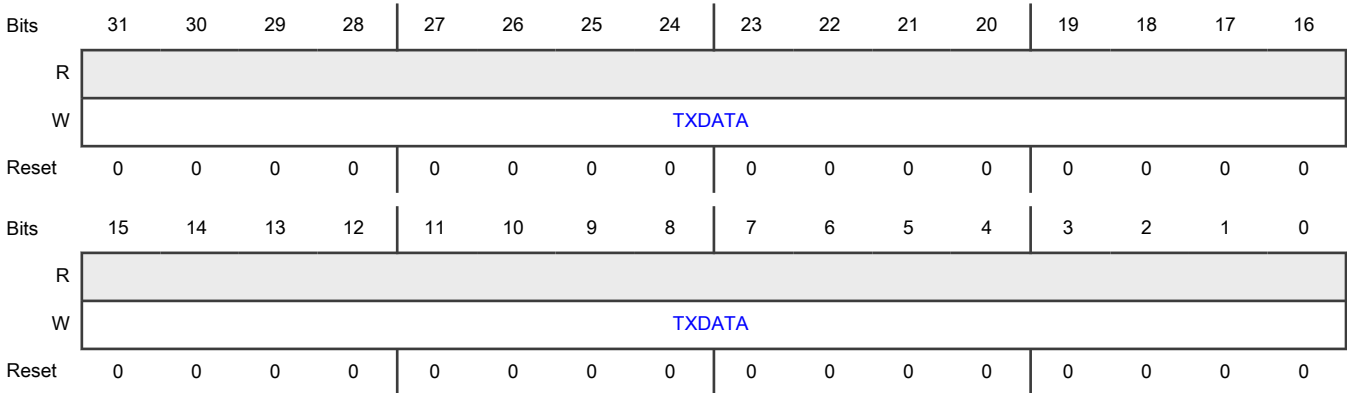
For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

Function

These registers provide write access to IP TX FIFO by IPS bus.

Diagram



Fields

Field	Function
31-0 TXDATA	TX Data For more details, see Filling data to IP TX FIFO

39.8.3.37 LUT a (LUT0 - LUT63)

Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

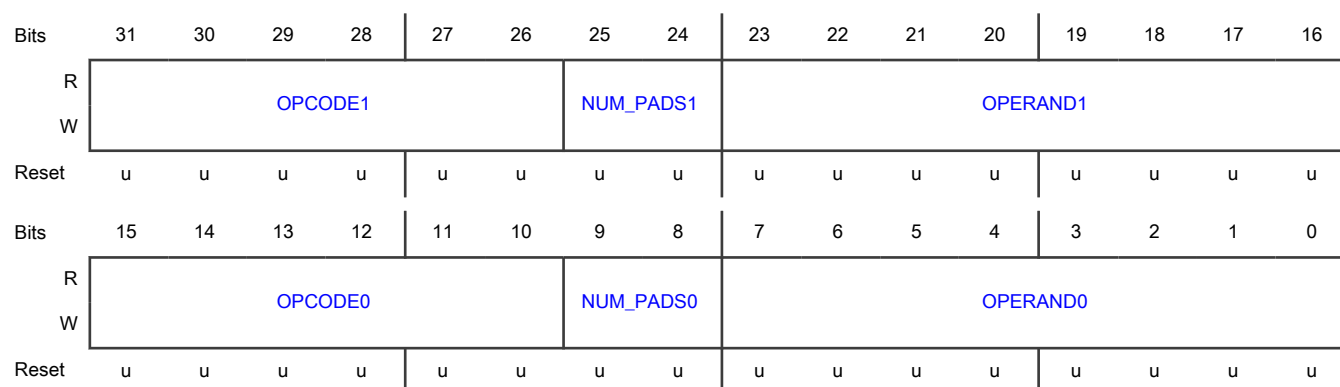
Function

The LUT is a look-up-table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI will fetch the command sequence from LUT when IP/AHB command triggered. There are 16 command sequences in LUT. See [LUT](#) and [Programmable sequence engine](#) for more details.

NOTE

LUT is implemented as memory, so the reset value is unknown.

Diagram



Fields

Field	Function
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 OPERAND1	OPERAND1
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0 OPERAND0	OPERAND0

39.9 AHB Memory Map definition

This section describes FlexSPI module AHB memory map in detail.

39.9.1 AHB Memory Map for Serial Flash memory access

AHB read/write access for serial flash memory is mapped to a specific address range. See the system memory map for specific address ranges supported.

AHB Bus feature supported for Serial Flash memory reading:

- Cachable and Non-Cachable access
- Prefetch Enable/Disable
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

AHB Bus feature for Serial Flash memory writing:

- Bufferable and Non-Bufferable access
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

Refer [Flash access by AHB Command](#) for more details about AHB access to Serial Flash memory.

Chapter 40

On-the-Fly AES Decryption Module (OTFAD)

40.1 Chip-specific OTFAD information

Table 449. Reference links to related information

Topic	Related module	Reference
Full description	OTFAD	OTFAD
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

40.1.1 Module instances

This device has one instance of the OTFAD module, OTFAD0.

40.2 Overview

One unmistakable trend in embedded processor design is the increasing need for hardware support of cryptographic calculations required for system security. In particular, there are emerging customer requirements to protect application code and data stored in external memories in an encrypted form and have the embedded processor provide hardware support for "on-the-fly" decryption. As the name suggests, the external memory image of the code and data is always in an encrypted format, and, in response to processor references of the memory space, the memory image is decrypted on the fly, returning the original values to the requesting bus master.

In these applications, the standard Advanced Encryption Standard (AES) [1] is the preferred cryptographic symmetric key block cipher algorithm. The AES function operates on 128-bit (16 byte) data blocks with either 128-, 192- or 256-bit secret keys. For this form of code and data encryption, the AES-128 mode, that is, the use of a 128-bit key is deemed sufficient.

The AES algorithm specifies a variable number of "rounds" be performed in the cryptographic calculation, based on the size of the key. For a 128-bit key, the data is processed through a series of calculations requiring 10 rounds to complete. Each round performs four different data transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the state array by different offsets, 3) mixing the data within each column of the state array, and 4) adding a round key to the state.

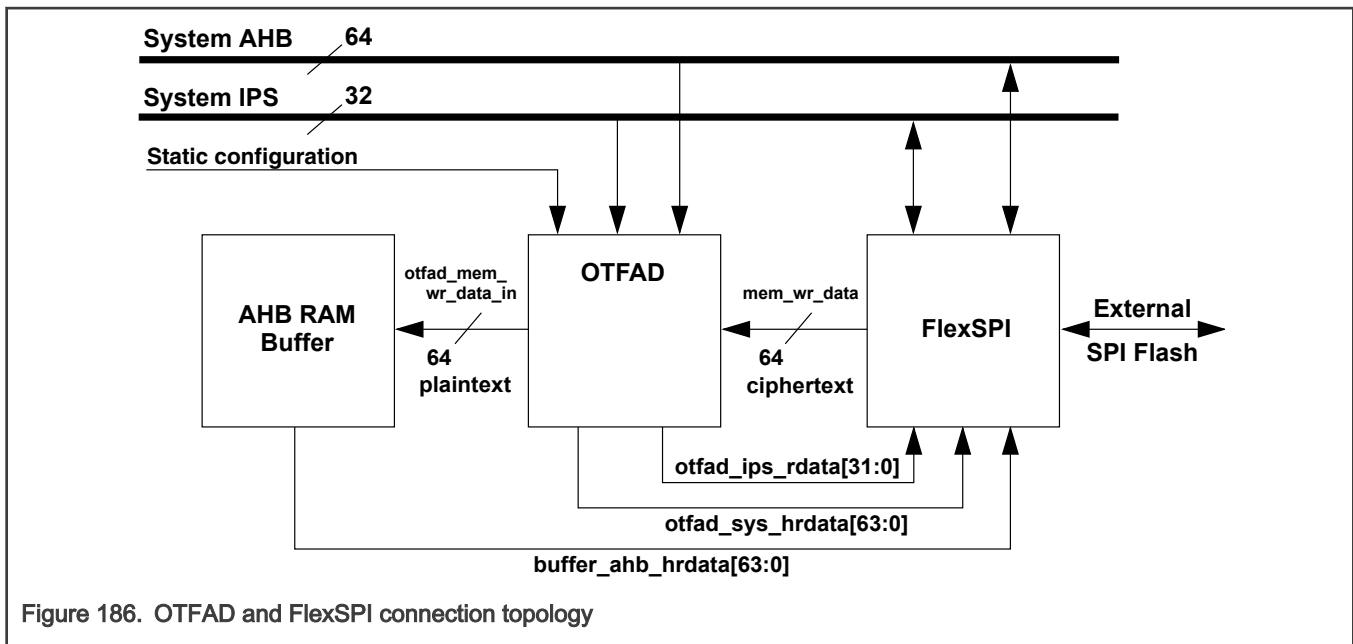
Many typical AES hardware implementations perform a single round operation in 1-2 machine cycles, but the performance requirements associated with on-the-fly decryption dictate a far more aggressive implementation.

The on-the-fly decryption engine is combined with the FlexSPI external flash memory controller. The FlexSPI memory controller supports glueless external connections to a wide variety of industry standard external "SPI" flashes. These memories were initially developed as very low pin count alternatives to traditional NOR flash memories with their wide parallel data interfaces. Over time, the complexity of the SPI flash memories has increased (and this trend is continuing) and now, two bank, double data rate interfaces can supply 16 bits of flash data every cycle once the address and command have been sent and the basic access has been performed. For a 100 MHz DDR Quad SPI external interface, the access time for a 4-beat, 64-bit, 32-byte burst fetch might generate an 18-4-4-4 response at the microprocessor pins. Given these access times, the latest generation of Quad SPI external flash devices provides a cost effective non-volatile memory solution that supports eXecute-in-Place (XiP) capabilities. Stated differently, the combined access speed of these external flash memories, coupled with internal processor-local cache memories, allow the application code to be executed directly from the external memory without the need to copy the code into another (faster) memory. This capability to execute directly from external flash without copying the code to another memory allows an easy migration from MCU to MPU configurations.

The resulting OTFAD engine adds ***zero cycles of additional latency*** to decrypt the code and data fetched from the external flash memory. The combination of the support for the CTR-AES128 mode plus an aggressive, pipelined implementation of the OTFAD performing up to 3 rounds in a single machine cycle allowing the engine to keep pace with the fastest data arrival rates. As a result, the OTFAD engine provides superior cryptographic decryption capabilities without compromising system performance for those embedded processor applications requiring this class of enhanced security.

40.2.1 Block diagram

Figure 186 shows the connection topology of the OTFAD and its relationship to the FlexSPI and its AHB RAM buffer.



40.2.2 Features

The key features of the OTFAD include:

- AES-128 Counter Mode On-the-Fly Decryption
 - 128-bit key and 128-bit data block sizes
 - 128-bit counter includes 64 bits of initialization vector + the 32-bit system address
 - Adds zero cycles of incremental latency for decryption when used with FlexSPI
 - Receives 64-bit encrypted data from FlexSPI, calculates decrypted data which is sent to AHB RAM buffer and bypassed back to system AHB read data bus
- Hardware support for 4 independent decryption segments, known as memory “contexts”
 - Each context has a unique 128-bit key, 64-bit counter and 64-bit memory region descriptor
- Functionally acts as a slave submodule to the FlexSPI
 - Logically connected between the FlexSPI and its AHB RAM buffer
 - Private 64-bit data buses for encrypted (ciphertext) and decrypted (plaintext) data
- Hardware microarchitecture
 - Heavily pipelined AES engine, optimized for encryption, performing 3 rounds per cycle
 - 64-bit AHB connections for easy integration to system bus fabric and FlexSPI

- Data storage for two 128-bit encrypted counters and three 64-bit decrypted data buffers
- Optimized for {32,64}-bit WRAP4 bursts (CPU cache miss fetch size, typical DMA fetch size)

40.2.3 References and terminology

These documents provide specific references to the cryptographic functions supported by the OTFAD:

1. FIPS Publication 197, "Advanced Encryption Standard (AES)", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), November 26, 2001.
2. NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), December 2001.

There are numerous names and phrases that are commonly used in describing cryptographic functions. The following table provides a summary of the terms used in describing the OTFAD. It was mostly extracted from Section 4.1, "Definitions and Abbreviations" of [2].

Table 450. Cryptographic terms

Term	Definition
Block Cipher	A family of functions and their inverse functions that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.
Block Size	The number of bits in an input (or output) block of the block cipher. For OTFAD, this is 128 bits (16 bytes).
Ciphertext	Encrypted data.
CTR	Counter.
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher operation and the inverse cipher operation.
Data Block (Block)	A sequence of bits whose length is the block size of the block cipher.
Decryption (Deciphering)	The process of a confidentiality mode that transforms encrypted data into the original usable data.
ECB	Electronic Codebook.
Encrypted Counter (ECTR)	An output in the CTR-AES decryption process, generated by encrypting the unique counter value. Exclusive-OR'ed with the ciphertext to produce plaintext.
Encryption (Enciphering)	The process of a confidentiality mode that transforms usable data into an unreadable form.
Forward Cipher Function (Forward Cipher Operation)	One of the two functions of the block cipher algorithm that is selected by the cryptographic key.
Initialization Vector (IV)	A data block that some modes of operation require as an additional initial input.

Table continues on the next page...

Table 450. Cryptographic terms (continued)

Term	Definition
Input Block	A data block that is an input to either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Inverse Cipher Function (Inverse Cipher Operation)	The function that reverses the transformation of the forward cipher function when the same cryptographic key is used.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.
Most Significant Bit(s)	The left-most bit(s) of a bit string.
Nonce	A value that is used only once.
OTP key, otp_key	One-time programmable key
Output Block	A data block that is an output of either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Plaintext	Usable data that is formatted as input to a mode.

40.3 Functional description

This section provides more details on the operation and implementation of the OTFAD.

40.3.1 Modes of operation

The OTFAD supports modes of operation as defined by input configuration (see [Static configuration signals](#)) and control signals. The operating mode is signaled in the SR and includes:

1. Logically Disabled Mode (LDM)

This mode is entered immediately after reset by the assertion of a static configuration input signal, which is treated as the highest priority configuration signal. While in this mode, the OTFAD is logically disabled. Input data from the FlexSPI is simply passed through to the AHB RAM buffer and no data decryptions are performed. The OTFAD's programming model is inaccessible and all attempted accesses are error terminated. If the static configuration input signal is asserted, the OTFAD is always in this mode.

2. Normal Mode (NM)

This mode is defined when the OTFAD is not in the logically disabled mode. In this mode, the OTFAD functions normally and is capable of performing data decryptions and/or data bypasses as it responds strictly to the memory transactions on its connected system buses. It should be noted the OTFAD operates in this mode regardless of the state of CR[GE]. The assertion of the global enable is required when the OTFAD is to perform data decryptions; if CR[GE] = 0, then the OTFAD simply bypasses all data fetched by the FlexSPI.

The OTFAD's operating mode is encoded and visible in the SR[MODE] field. For more information, see [OTFAD operating modes](#).

In normal mode, as a memory-mapped device located in the core platform's clock domain, it responds based on the memory addresses of its connected system buses. The slave peripheral bus may be used to access the OTFAD's programming model to configure the module's operation.

40.3.2 CTR-AES128 mode basics

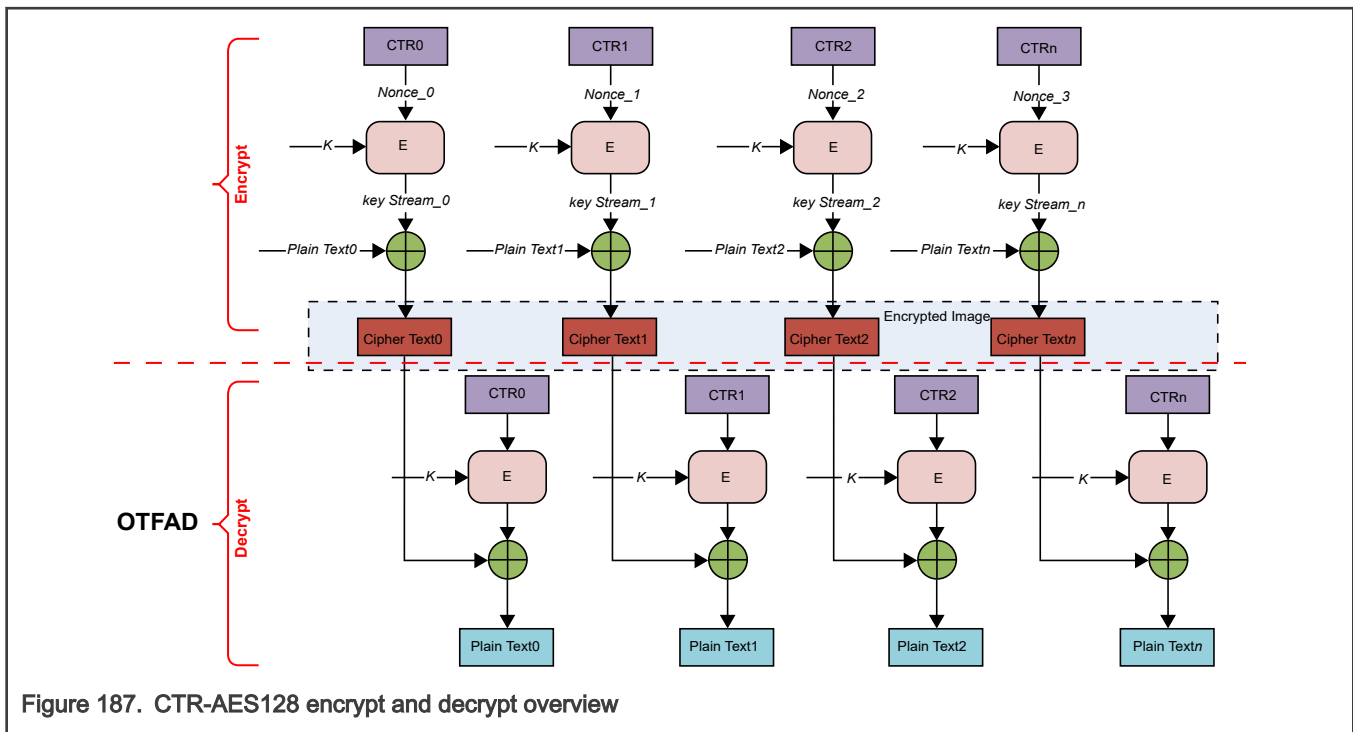
When processing data fetched from an external memory, the OTFAD implements the CTR-AES128 mode for on-the-fly decryption.

The OTFAD engine implements a block cipher mode of operation, specifically supporting counter mode (CTR). The CTR mode "provides a confidentiality mode that features the application of the forward cipher (encryption) to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa." [2]

The CTR-AES128 mode was selected for two reasons: 1) it produces a cryptographically stronger solution than the simple Electronic Cookbook (ECB) mode since each block in the sequence is different than every other block, even for the same input data, and 2) the definition of CTR mode allows the 128-bit counter values to include the system memory address of the referenced flash block and to be computed in advance of their actual usage to convert the ciphertext back into plaintext for the requesting bus master device in the microprocessor - typically the CPU itself.

As a result, the OTFAD module adds zero cycles of additional latency to decrypt the code and data fetched from the external flash memory. The combined support for the CTR-AES128 mode plus an aggressive, pipelined implementation of the OTFAD performing up to 3 rounds in a single machine cycle allows the engine to keep pace with the fastest data arrival rates.

The OTFAD module performs the decryption operation within the overall CTR-AES128 flow, as shown in [Figure 187](#).



As shown in [Figure 187](#), CTR-AES128 scheme combines a unique 128-bit counter (CTR_n) with a 128-bit key (K) to encrypt (Ciphertext $_n$) or decrypt (Plaintext $_n$) each 128 bit data block.

Authorized software executes the "encrypt" portion to create the ciphertext image stored in the external memory, while the OTFAD module, in conjunction with the FlexSPI external memory controller, executes the "decrypt" portion.

Consider these two basic operations in more detail.

For encryption, the 0-modulo-16 byte system address is combined with 96 bits of counter information defined in the memory context registers to produce a unique 128-bit CTR_n value for each data block. Specifically, the OTFAD's implementation defines the 128-bit CTR_n for memory context "x" as the concatenation of 4 values:

```
CTRn_x[127:0] = {CTR_W0_x[C0...C3], // 32 bits of pre-programmed CTR
                 CTR_W1_x[C4...C7], // another 32 bits of CTR
```

```

CTR_W0_x[C0...C3] ^ CTR_W1_x[C4...C7], // exclusive-OR of CTR values

systemAddress[31-4], 0000b} // 0-modulo-16 system address

```

NOTE

The addresses used by both (software) encryption and OTFAD decryption are the physical system addresses associated with the FlexSPI, never the code alias addresses.

Software uses this CTR_n value as the data input to the AES128 encryption process (E) to produce the Key_Stream_n output. The final step in encryption is to exclusive-OR the Key_Stream_n value with the data block's plaintext to create the 128-bit block of Ciphertext_n which is stored in the external memory. This process is then repeated for each 16-byte data block of the code and data image needing encryption.

The OTFAD's decryption process begins by inputting the same CTR_n value and same key K to the AES128 encryption process (E) to generate a 128-bit encrypted counter (ECTR_n, not explicitly labeled in [Figure 187](#)). The ECTR_n is exclusive-OR'ed with the Ciphertext_n to produce the Plaintext_n. The 128-bit key (K) is defined in the KEY_W[0-3]_x registers for context "x".

Since the encrypted counter value is *not* dependent on the ciphertext, it can be pre-calculated in response to a system bus access request such that the OTFAD only needs to perform the exclusive-OR once the ciphertext has been fetched from the external memory. This is the key concept in allowing the OTFAD to add zero incremental cycles of fetch latency.

For additional details on the OTFAD's processing during a 32-byte burst read, see [Typical OTFAD CTR-AES128 decryption operation](#).

40.3.3 Microarchitecture overview

This section provides additional details on the hardware implementation. The following subsections provide details on the hardware decryption operations. Refer to the block diagram of [Figure 186](#).

40.3.3.1 Context determination

The first step in a CTR-AES128 decryption is the determination of the external memory context.

The OTFAD supports four memory contexts where each has its own 128-bit key, 64-bits of counter and a 64-bit memory region descriptor. The region descriptor includes start and end addresses plus three bits of control and configuration. See [AES Region Descriptor Word0 \(CTX0_RGD_W0 - CTX3_RGD_W0\)](#) and [AES Region Descriptor Word1 \(CTX0_RGD_W1 - CTX3_RGD_W1\)](#) for details.

NOTE

Always use an address range within the physical address space assigned to the FlexSPI module when defining an external memory context in a region descriptor, even if the system application uses the aliased area of the FlexSPI module.

The system address (SYSADDR) is compared to the start and end addresses of each context to determine a region "hit". To determine if the current reference hits in a given context, two magnitude comparators are used in conjunction with the region's start and end addresses. The boolean equation for this portion of the hit determination is defined as:

```

context_hit_n
= ((addr[31:10] >= rgd_w0_n[srtaddr]) & (addr[31:10] <= rgd_w1_n[endaddr]))
& rgd_w1_n[vld]

```

where `addr[*]` is the current system reference address, `rgd_w0_n[srtaddr]` and `rgd_w1_n[endaddr]` are the start and end addresses, and `rgd_w1_n[vld]` is the valid bit, all from context region descriptor n.

There are no hardware checks to verify that `rgd_w1_n[endaddr] ≥ rgd_w0_n[srtaddr]`, and it is software's responsibility to load appropriate values into these fields of the context region descriptor.

Recall the context memory regions are defined as modulo-1024 byte values to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. The result is that each AHB transfer, regardless of access size or length, is limited to a single context for purposes of AES decryption.

Since each context has unique key and counter values, the OTFAD does not support any form of memory region overlap. For system accesses that hit in multiple contexts or no contexts, the fetched data is simply bypassed.

For accesses that hit in a single context, there is a separate register bit (RGD_W1_n[ADE]) that enables AES decryption or a simple bypass of the fetched data. Based on the hit determination and the state of the ADE indicator, the appropriate 128-bit key and 64 bit counter are selected from the context registers.

40.3.3.2 AES engine and encrypted counter generation

Once the appropriate context has been determined and the corresponding key and counter values selected, the OTFAD proceeds with the generation of the encrypted counters. Since the dominant FlexSPI system bus reference is a 64-bit WRAP4 burst transfer in response to a cache miss, the OTFAD is optimized for this transaction type. The subsequent discussion is described in terms of the 64-bit WRAP4 transfer.

As the 64-bit WRAP4 accesses 32 bytes of data, the OTFAD calculates two 128-bit encrypted counters since there are two AES128 data blocks referenced by this transaction. Since external flash SPI memories typically do not support any type of address wrapping, the FlexSPI forces the external fetches to begin at the corresponding 0-modulo-size address. Accordingly, the OTFAD calculates the (first) encrypted counter associated with system address 0-modulo-32, and then the (second) address 16-modulo-32. The two encrypted counter calculations are performed in the AES Engine as the FlexSPI is making the external flash fetches.

As each encrypted counter is generated, the 128-bit value is stored in one of the two data registers connected to the AES Engine output.

Recall the OTFAD's AES Engine is heavily pipelined, performing up to 3 rounds per cycle, so the encryption speed (4 cycles total) matches the fastest data arrival rate.

Each AES round performs 4 different data transformations:

- Byte substitution
- State array row shift
- State array column mix
- Round key addition

Recall the FlexSPI initiated the external fetch at a 0-modulo-32 address, so as the encrypted data is fetched and presented to the OTFAD, it uses the first encrypted counter to perform the required exclusive-OR operation to generate the corresponding plaintext. The generated plaintext is output to the FlexSPI's AHB RAM buffer to be written into that memory (which operates as a small cache for FlexSPI accesses). See [Figure 186](#). This process continues as each of the four fetches of the 64-bit WRAP4 burst is presented to the OTFAD, decrypted and then sent to the AHB RAM buffer.

40.3.3.3 Decrypted data buffer operation

Although the FlexSPI initiated its external fetches from a 0-modulo-32 address, the system bus transaction does not have any restrictions about its starting address and can begin at any of the 64-bit addresses. So, as the sequential fetched data is decrypted by the OTFAD, the resulting plaintext is also loaded into three 64-bit decrypted data buffers.

Depending on the original address of the system bus reference, the OTFAD either decrypts and outputs data to *both* the AHB RAM buffer and the system bus, or registers the decrypted data so it can be sourced later after the system bus address "wraps". Consider the following two examples: first, examine the sequencing on a 64-bit WRAP4 with an original address of 0x00, and a second example with a starting address of 0x18.

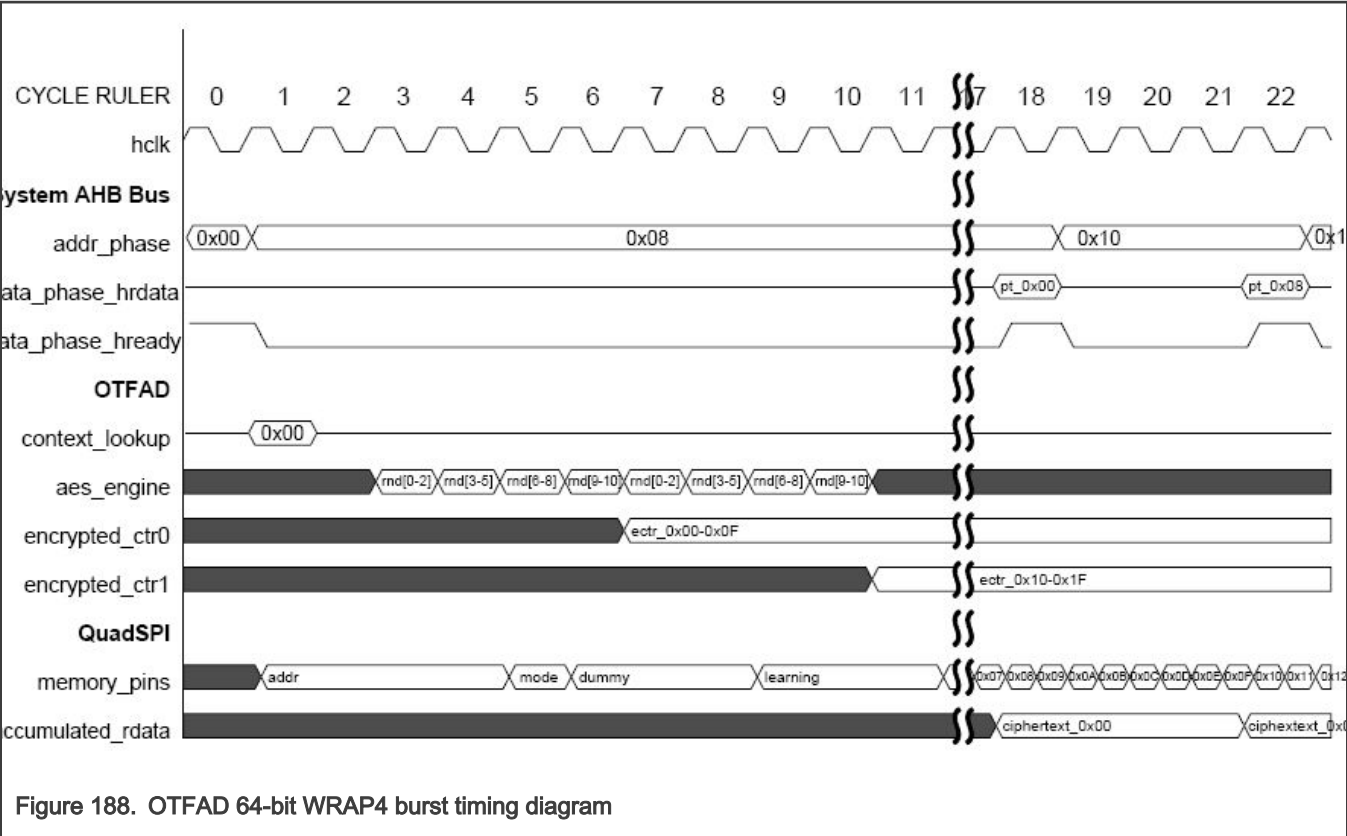
In both examples, the FlexSPI's fetch addresses are {0x00, 0x08, 0x10, 0x18}. In the first example, the system bus addresses are {0x00, 0x08, 0x10, 0x18}, while in the second example, the sequence is {0x18, 0x00, 0x08, 0x10}. See [Table 451](#) for a description of the operations for these two examples.

Table 451. OTFAD and decrypted data buffer operation

FlexSPI fetch	SystemBus fetch	Description
Example 1 64-bit WRAP4, miss @ 0x00		
0x00	0x00	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x08	0x08	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x10	0x10	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
0x18	0x18	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
Example 2 64-bit WRAP4, miss @ 0x18		
0x00	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 0, SystemBus read is stalled
0x08	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 1, SystemBus read is stalled
0x10	0x18	OTFAD decrypts data, sends to AHB RAM Buffer and loads into Decrypted Data Buffer 2, SystemBus read is stalled
0x18	0x18	OTFAD decrypts data and simultaneously sends to AHB RAM Buffer and SystemBus
	0x00	OTFAD sources data from decrypted data buffer 0
	0x08	OTFAD sources data from decrypted data buffer 1
	0x10	OTFAD sources data from decrypted data buffer 2

The resulting behavior from the combined FlexSPI + OTFAD provides the best system performance. For longer burst sizes, for example, 64-bit WRAP8 and 64-bit WRAP16, the implemented number of decrypted data buffers is not sufficient to temporary store the entire burst. In these cases, the three OTFAD buffers are used in conjunction with "hits" from the AHB RAM Buffer to source the decrypted data back to the system bus.

A simplified (partial) timing diagram for the 64-bit WRAP4 burst of Example 1 is shown in [Figure 188](#).



In Figure 188, the system AHB read data bus "pt_<burst_offset>" is meant to refer to decrypted plaintext data. The zero cycle incremental delay to perform the required CTR-AES128 decryption is visible in cycles 18 (the doubleword at burst offset 0x00) and 22 (doubleword at burst offset 0x08).

40.3.4 Clocking

This module has no special clocking considerations.

40.3.5 Interrupts

This module has no interrupts.

40.4 External signals

The OTFAD module does not directly support any external interfaces.

The internal interfaces include the address and attribute input signals from a standard 64-bit AMBA-AHB system bus and a 32-bit IPS slave peripheral bus for all programming model accesses, as shown in Figure 186.

The OTFAD receives the IPS input signals and generates a read data output which is sent to the FlexSPI where it is combined with local signals to form an IPS read data output as well as bus termination control signals. There are also three private 64-bit data buses: one input from the FlexSPI typically providing ciphertext and two outputs containing plaintext data after decryption - one sent to the AHB RAM buffer and another sent back to the FlexSPI for forwarding onto the system AHB read data bus. There is also a 64-bit read data bus from the AHB RAM buffer back to the FlexSPI for servicing system bus accesses which "hit" in the buffer.

40.4.1 Static configuration signals

The specific value settings of the static configuration signals are shown in Table 452.

Table 452. Static configuration signal settings

Configuration Signal	Description	Value
otfad_enable	Enable OTFAD decryption	connected
restrict_otfad_ips	Restrict access to CR, SR	0

40.5 Initialization

40.5.1 OTFAD configuration details

The operating configuration of the OTFAD is controlled by a number of static input signals (see [Static configuration signals](#)) plus programmable bits in the Control Register. All of the basic modes are discussed in [Modes of operation](#) and [OTFAD operating modes](#). This section focuses on the configuration capabilities associated with Normal mode (SR[MODE] = 00).

There is a static input signal and an associated CR bit that define various configuration features. In all cases, the static input signal and the associated CR bit are logically summed (OR'd) together to form an “effective” configuration bit. The combined control is visible via a slave peripheral bus CR read. See [Table 453](#).

Table 453. OTFAD normal mode configurations

otfad_enable CR[GE]	Description
0	1) OTFAD operates with all data decryption disabled; FlexSPI fetched data is simply bypassed.
1	2) Basic OTFAD processing is enabled; decryptions are enabled based on context n “hit” and RGD_W1_n[ADE, VLD].

In addition to the static configuration inputs and associated CR bits shown in [Table 453](#), there is one additional set: a static configuration input and CR[RRAE]. When either the configuration input signal or CR[RRAE] is asserted, only the CR and SR registers can be accessed; attempted accesses of all other registers are treated as read-as-zero, write-ignore.

40.6 Application information

40.6.1 OTFAD operating modes

As discussed in [Modes of operation](#), the OTFAD supports two modes of operation as defined by input configuration and control signals. The mode states and their transitions are shown in [Figure 189](#). The states include Normal Mode (00) and Logically Disabled Mode (11). The operating mode state information is visible in the SR[MODE] field.

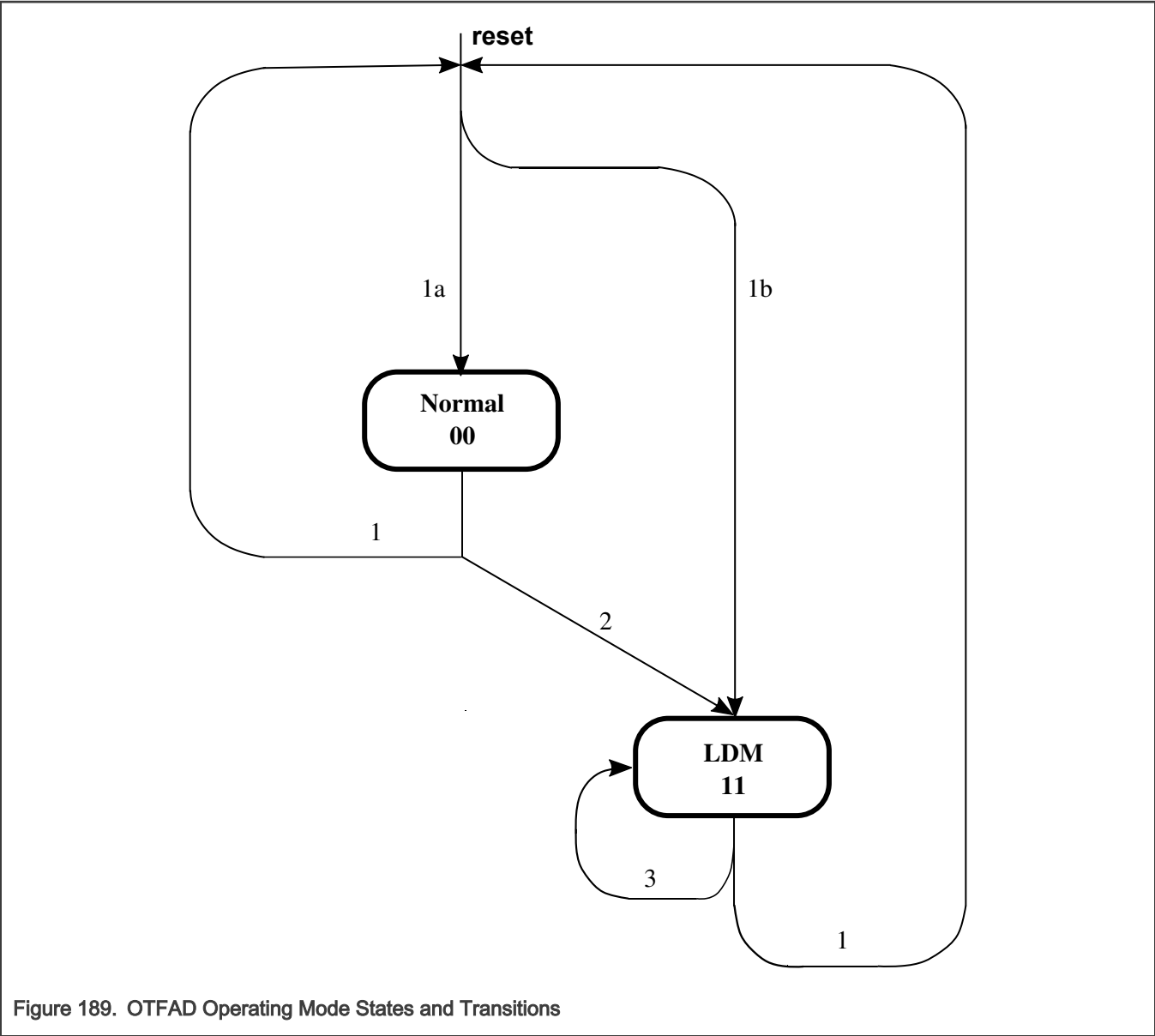


Figure 189. OTFAD Operating Mode States and Transitions

The state transitions are defined in [Table 454](#).

Table 454. OTFAD mode state transitions

Current state	Next state	Transition	Description
–	Normal, LDM	1 -> 1a, 1b	Any reset event initializes the OTFAD mode state.
–	Normal	1a	Any reset, combined with a negated <code>otfad_disable_override</code> configuration input signal, unconditionally forces OTFAD into the Normal state.
–	LDM	1b	Any reset, combined with an asserted <code>otfad_disable_override</code> configuration input signal, unconditionally forces OTFAD into the LDM state.

Table continues on the next page...

Table 454. OTFAD mode state transitions (continued)

Current state	Next state	Transition	Description
Normal	LDM	2	A transition from Normal to LDM is made when the boolean expression (cr[fldm] & ~reset) is true.
LDM	LDM	3	The LDM is retained while reset is not asserted.

40.6.2 Typical OTFAD CTR-AES128 decryption operation

The dominant use case in normal mode is expected to be CTR-AES128 decryption operations performed in response to a 64-bit WRAP4 burst read in response to a processor cache miss. The following C code is a simplified representation of the OTFAD processing associated with this use case. The ciphertext fetched by the FlexSPI is the `spi_ciphertext[32]` byte array, and the resulting decrypted data is `otfad_plaintext[32]`. See [Figure 190](#).

```

void KeyExpansion (unsigned char ckey[], int nbits, unsigned int w[]);
void Cipher (unsigned char cin[], unsigned int w[], int nr, unsigned char cout[]);

unsigned char          spi_ciphertext[32];
unsigned char          otfad_plaintext[32];

union context_regs {
    volatile unsigned char    key[16];
    volatile unsigned char    ctr[8];
    volatile unsigned int     rgd[2];
    volatile unsigned int     unused_filler[8]
} otfad_context[4];

void otfad_32byte_processing (unsigned int sys_addr,
                             unsigned char spi_ciphertext[],
                             unsigned char otfad_plaintext[])
{
    unsigned int    i;
    unsigned int    ctx_hit;
    unsigned int    combined_hits;
    unsigned int    enable_decryption;
    unsigned int    w[44];          // expanded key
    unsigned char    encrypted_ctr[2][16];
    unsigned char    temp_ctr[16];
    unsigned int    local_addr;
    unsigned int    hit_addr;

    local_addr &= 0xffffffff;        // force byte addr to 0-mod-16 (aes size)
    hit_addr = local_addr & 0xfffffc00;    // force hit addr to 0-mod-1K

    // context determination
    combined_hits = 0;                // 4-bit hit indicator: ctx{3,2,1,0}
    for (i = 0; i < 4; i++) {
        ctx_hit = ((hit_addr >= otfad_context[i].rgd[0])    // srtaddr
                   & (hit_addr <= otfad_context[i].rgd[1])) // endaddr
                   & (otfad_context[i].rgd[i] & 1);        // vld
        combined_hits += (ctx_hit << i);
    }
    // check combined_hits to determine if decryption is enabled
    enable_decryption = 0;

```

```

switch (combined_hits) {
    case 1:                // ctx0 hit
        ctx_hit = 0;
        if ((otfad_context[0].rgd[1] & 2) != 0) {        // ade
            enable_decryption = 1;
        }
        break;
    case 2:                // ctx1 hit
        ctx_hit = 1;
        if ((otfad_context[1].rgd[1] & 2) != 0) {        // ade
            enable_decryption = 1;
        }
        break;
    case 4:                // ctx2 hit
        ctx_hit = 2;
        if ((otfad_context[2].rgd[1] & 2) != 0) {        // ade
            enable_decryption = 1;
        }
        break;
    case 8:                // ctx3 hit
        ctx_hit = 3;
        if ((otfad_context[3].rgd[1] & 2) != 0) {        // ade
            enable_decryption = 1;
        }
        break;
    default:                // none or multiple ctx hits
        ctx_hit = 0;
        enable_decryption = 0;                // no decryption, bypass data
        break;
}

// clear encrypted counters
for (i = 0; i < 16; i++) {
    encrypted_ctr[0][i] = 0;
    encrypted_ctr[1][i] = 0;
}

// if decryption is enabled, calculate two encrypted counter values
if (enable_decryption == 1) {
    KeyExpansion (&otfad_context[ctx_hit].key[0], 128, w);
    // encrypt the counter for the 1st 16 bytes
    // step 1: generate the 128-bit counter
    // step 1a: first 8 bytes are taken directly from context.ctr[]
    for (i = 0; i < 8; i++) {
        temp_ctr[i] = otfad_context[ctx_hit].ctr[i];
    }
    // step 1b: next 4 bytes are xor of ctr[0-3] ^ ctr[4-7]
    temp_ctr[8] = temp_ctr[0] ^ temp_ctr[4];
    temp_ctr[9] = temp_ctr[1] ^ temp_ctr[5];
    temp_ctr[10] = temp_ctr[2] ^ temp_ctr[6];
    temp_ctr[11] = temp_ctr[3] ^ temp_ctr[7];
    // step 1c: final 4 bytes are local_addr
    temp_ctr[12] = (local_addr & 0xff000000) >> 24;
    temp_ctr[13] = (local_addr & 0x00ff0000) >> 16;
    temp_ctr[14] = (local_addr & 0x0000ff00) >> 8;
    temp_ctr[15] = (local_addr & 0x000000ff);

    // step 2: perform the 1st counter encryption
    Cipher (&temp_ctr[0], w, 10, &encrypted_ctr[0][0]);
}

```

```

// encrypt the counter for the 2nd 16 bytes
// step 1a: increment the local_addr by 16
local_addr += 16;
// step 1b: final 4 bytes of counter are new local_addr
temp_ctr[12] = (local_addr & 0xff000000) >> 24;
temp_ctr[13] = (local_addr & 0x00ff0000) >> 16;
temp_ctr[14] = (local_addr & 0x0000ff00) >> 8;
temp_ctr[15] = (local_addr & 0x000000ff);

// step 2: perform the 2nd counter encryption
Cipher (&temp_ctr[0], w, 10, &encrypted_ctr[1][0]);
}

// generate plaintext data = f(encrypted_ctr[], spi_ciphertext[])
// for bypass operations, the encrypted_ctr[] = 0, so plain = cipher
// since spi_ciphertext arrives 8 bytes at a time, code does the same...
for (i = 0; i < 8; i++)
    otfad_plaintext[i] = encrypted_ctr[0][i] ^ spi_ciphertext[i];
for (i = 8; i < 16; i++)
    otfad_plaintext[i] = encrypted_ctr[0][i] ^ spi_ciphertext[i];
for (i = 16; i < 24; i++)
    otfad_plaintext[i] = encrypted_ctr[1][i-16] ^ spi_ciphertext[i];
for (i = 24; i < 32; i++)
    otfad_plaintext[i] = encrypted_ctr[1][i-16] ^ spi_ciphertext[i];
}

```

Figure 190. Basic OTFAD 32-byte decryption processing

40.7 Register descriptions and data organization

The registers of the OTFAD programming model can only be accessed while in secure privileged mode. Unless noted otherwise, the programming model registers can be accessed via 8-, 16- or 32-bit reads and 32-bit write references. Attempted accesses in a different operating mode, using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error unless noted otherwise. See also [Data organization and endianness considerations](#).

In the registers named *CTXn_<regname>*, the "n" is the context number.

Software access in this environment may be considerably reduced based on the state of the CR[RRAE] and static chip configuration inputs (see [Static configuration signals](#)). For the affected *CTXn_<regname>* registers, their access values are shown as "RW" in this section; however, these registers are treated as read-as-zero/write-ignore (RAZ/WI) if operating in the restricted register access mode (CR[RRAE] = 1 and SR[RRAM] = 1). See the individual register descriptions for more information.

40.7.1 OTFAD register descriptions

40.7.1.1 OTFAD memory map

OTFAD0 base address: 4013_4000h

Offset	Register	Width (In bits)	Access	Reset value
C00h	Control Register (CR)	32	RW	0000_0000h
C04h	Status Register (SR)	32	R	0000_0040h
D00h	AES Key Word (CTX0_KEY0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
D04h	AES Key Word (CTX0_KEY1)	32	RW	0000_0000h
D08h	AES Key Word (CTX0_KEY2)	32	RW	0000_0000h
D0Ch	AES Key Word (CTX0_KEY3)	32	RW	0000_0000h
D10h	AES Counter Word (CTX0_CTR0)	32	RW	0000_0000h
D14h	AES Counter Word (CTX0_CTR1)	32	RW	0000_0000h
D18h	AES Region Descriptor Word0 (CTX0_RGD_W0)	32	RW	0000_0000h
D1Ch	AES Region Descriptor Word1 (CTX0_RGD_W1)	32	RW	0000_03F8h
D40h	AES Key Word (CTX1_KEY0)	32	RW	0000_0000h
D44h	AES Key Word (CTX1_KEY1)	32	RW	0000_0000h
D48h	AES Key Word (CTX1_KEY2)	32	RW	0000_0000h
D4Ch	AES Key Word (CTX1_KEY3)	32	RW	0000_0000h
D50h	AES Counter Word (CTX1_CTR0)	32	RW	0000_0000h
D54h	AES Counter Word (CTX1_CTR1)	32	RW	0000_0000h
D58h	AES Region Descriptor Word0 (CTX1_RGD_W0)	32	RW	0000_0000h
D5Ch	AES Region Descriptor Word1 (CTX1_RGD_W1)	32	RW	0000_03F8h
D80h	AES Key Word (CTX2_KEY0)	32	RW	0000_0000h
D84h	AES Key Word (CTX2_KEY1)	32	RW	0000_0000h
D88h	AES Key Word (CTX2_KEY2)	32	RW	0000_0000h
D8Ch	AES Key Word (CTX2_KEY3)	32	RW	0000_0000h
D90h	AES Counter Word (CTX2_CTR0)	32	RW	0000_0000h
D94h	AES Counter Word (CTX2_CTR1)	32	RW	0000_0000h
D98h	AES Region Descriptor Word0 (CTX2_RGD_W0)	32	RW	0000_0000h
D9Ch	AES Region Descriptor Word1 (CTX2_RGD_W1)	32	RW	0000_03F8h
DC0h	AES Key Word (CTX3_KEY0)	32	RW	0000_0000h
DC4h	AES Key Word (CTX3_KEY1)	32	RW	0000_0000h
DC8h	AES Key Word (CTX3_KEY2)	32	RW	0000_0000h
DCCh	AES Key Word (CTX3_KEY3)	32	RW	0000_0000h
DD0h	AES Counter Word (CTX3_CTR0)	32	RW	0000_0000h
DD4h	AES Counter Word (CTX3_CTR1)	32	RW	0000_0000h
DD8h	AES Region Descriptor Word0 (CTX3_RGD_W0)	32	RW	0000_0000h
DDCh	AES Region Descriptor Word1 (CTX3_RGD_W1)	32	RW	0000_03F8h

40.7.1.2 Control Register (CR)

Offset

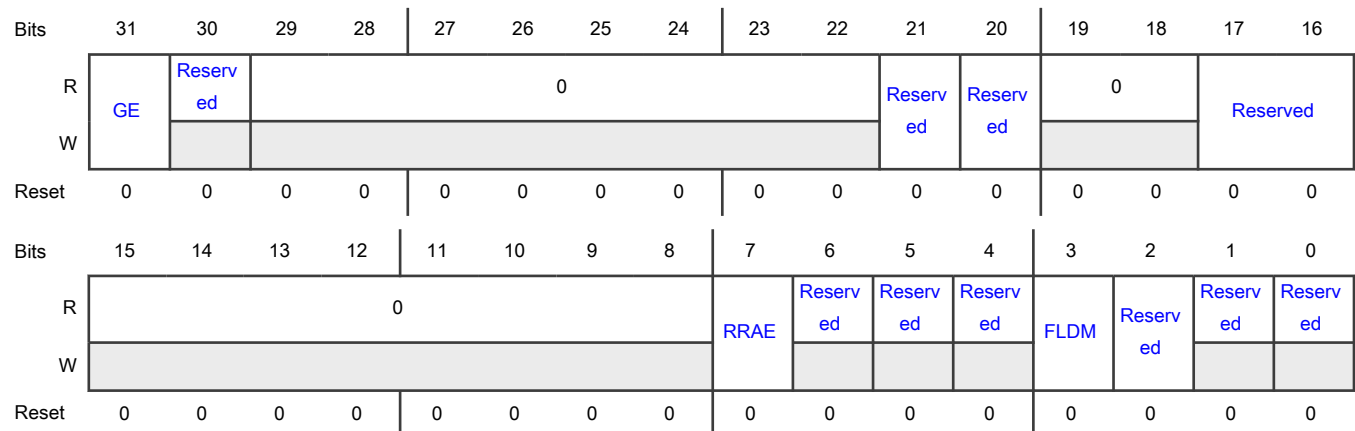
Register	Offset
CR	C00h

Function

This register defines the operating configuration of the OTFAD including a global enable indicator.

There are 2 register fields in the CR that control the OTFAD configuration. These fields (GE, RRAE) are generated based on both the contents of the CR plus static input configuration signals. For both bits, the contents of the CR flag is logically summed (OR'd) with the static input configuration signals to form the actual control used by the OTFAD. Reads of the CR return the logical summation of these configuration controls. Entry into the LDM mode clears both indicators. For more details, see [OTFAD configuration details](#).

Diagram



Fields

Field	Function
31	Global OTFAD Enable
GE	This field enables the OTFAD operation.
	NOTE The global enabled/disabled state of the OTFAD module is based on the logical OR of the value written to GE and the value of the static input <i>otfad_enable</i>. For example, if <i>otfad_enable</i> = 1, the OTFAD module is always enabled. Reading GE returns the global enabled/disabled state of the OTFAD module. It is cleared by entry into the LDM mode. 0b - OTFAD has decryption disabled. All data fetched by the FlexSPI bypasses OTFAD processing.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - OTFAD has decryption enabled, and processes data fetched by the FlexSPI as defined by the hardware configuration.
30 —	Reserved
29-22 —	Reserved
21 —	Reserved NOTE When writing to this register, software must write 0 to this field. This field always reads as zero.
20 —	Reserved NOTE When accessing this register, software must write 0 to this field and ignore read values.
19-18 —	Reserved
17-16 —	Reserved NOTE When accessing this register, software must write 0s to this field and ignore read values.
15-8 —	Reserved
7 RRAE	Restricted Register Access Enable If this bit is asserted, only the CR, SR and optional MDPC registers can be accessed; attempted accesses of all other registers are treated as RAZ/WI (read-as-zero/write-ignore). This indicator is sticky; once set, it remains asserted until the next system reset, when it is cleared. It is also asserted by entry into the LDM mode. The value of this field is reflected in the SR[RRAM] flag. 0b - Register access is fully enabled. The OTFAD programming model registers can be accessed “normally”. 1b - Register access is restricted and only the CR, SR and optional MDPC registers can be accessed; others are treated as RAZ/WI.
6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 —	Reserved
4 —	Reserved
3 FLDM	Force Logically Disabled Mode This field is intended to provide a mechanism for software testing of entry into the Logically Disabled Mode (LDM). This indicator is sticky; once set, it remains asserted until the next system reset, when it is cleared. 0b - No effect on the operating mode. 1b - Force entry into LDM after a write with this data bit set. SR[MODE] signals the operating mode.
2 —	Reserved Reserved. Writes to this field should be '0' only.
1 —	Reserved
0 —	Reserved

40.7.1.3 Status Register (SR)

Offset

Register	Offset
SR	C04h

Function

This register provides OTFAD status information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed	Reserv ed	GEM	RRAM	HRL				0				Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				Reserved				NCTX				MODE		MDPC P	Reserv ed
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 GEM	Global Enable Mode This indicator signals the global enabled/disabled state of the OTFAD. This flag is identical to the read value of CR[GE]. 0b - OTFAD is disabled. All data fetched by the FlexSPI bypasses OTFAD processing. 1b - OTFAD is enabled, and processes data fetched by the FlexSPI as defined by the hardware configuration.
28 RRAM	Restricted Register Access Mode This indicator signals that accesses to the slave peripheral bus are operating in a restricted mode, where only the CR, SR and optional MDPC registers can be referenced. This access mode can be defined by the assertion of CR[RRAE] = 1. Attempted accesses to other registers are treated as RAZ/WI. This flag is identical to the read value of CR[RRAE]. 0b - Register access is fully enabled. The OTFAD programming model registers can be accessed “normally”. 1b - Register access is restricted and only the CR, SR and optional MDPC registers can be accessed; others are treated as RAZ/WI.
27-24 HRL	Hardware Revision Level This field reads as zero.
23-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-16 —	Reserved
15-12 —	Reserved
11-8 —	Reserved
7-4 NCTX	Number of Contexts This field signals the number of implemented hardware contexts. It reads as 4.
3-2 MODE	Operating Mode This field specifies the OTFAD's operating mode. Input configuration and control signals force entry from normal mode (NRM) into a special operating mode (LDM) . See Modes of operation for more details. 00b - Operating in Normal mode (NRM) 01b - Unused (reserved) 10b - Unused (reserved) 11b - Operating in Logically Disabled Mode (LDM)
1 MDPCP	MDPC Present This field signals the presence of the Multi-Dimensional Parity Checker. For this device, it always has the value 0.
0 —	Reserved

40.7.1.4 AES Key Word (CTX0_KEY0 - CTX3_KEY3)

Offset

For n = 0 to 3; m = 0 to 3:

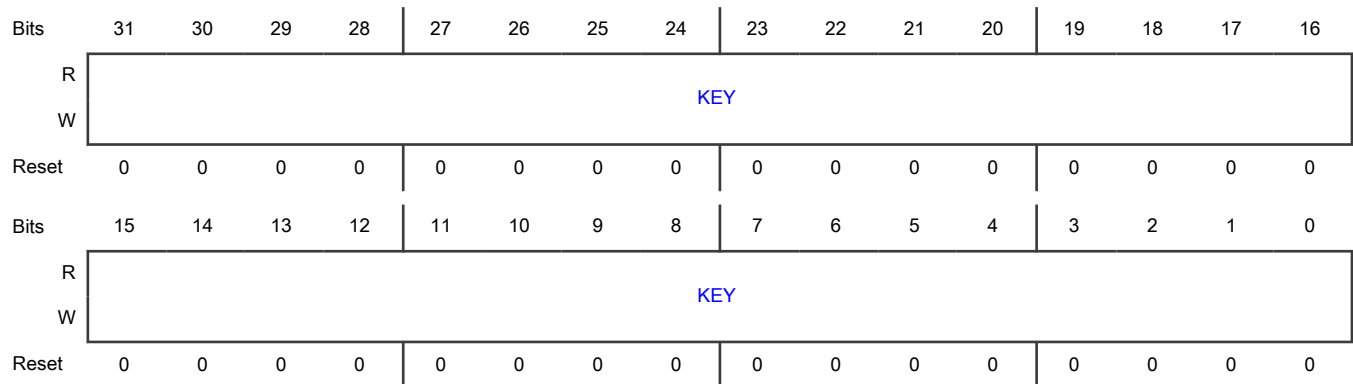
Register	Offset
CTXn_KEYm	D00h + (n × 40h) + (m × 4h)

Function

The CTXn_KEYm registers provide a 2-dimensional data structure for local OTFAD storage of the 128-bit key for context “n”. There are four consecutive memory-mapped register words containing the key used for the AES calculations associated with the given context. The programming model view of the CTXn_KEYm registers is a little-endian 16-element byte data array, while the 128-bit key is defined as the concatenation of {A0, A1, A2,..., A14, A15}.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Diagram



Fields

Field	Function
31-0 KEY	AES Key The key can be written using the slave peripheral bus. The four consecutive little-endian memory-mapped registers provide 128 bits of key storage. Word0: KEY[31:0][A03, A02, A01, A00] Word1: KEY[31:0][A07, A06, A05, A04] Word2: KEY[31:0][A11, A10, A09, A08] Word3: KEY[31:0][A15, A14, A13, A12]

40.7.1.5 AES Counter Word (CTX0_CTR0 - CTX3_CTR1)

Offset

For n = 0 to 3; m = 0 to 1:

Register	Offset
CTXn_CTRm	D10h + (n × 40h) + (m × 4h)

Function

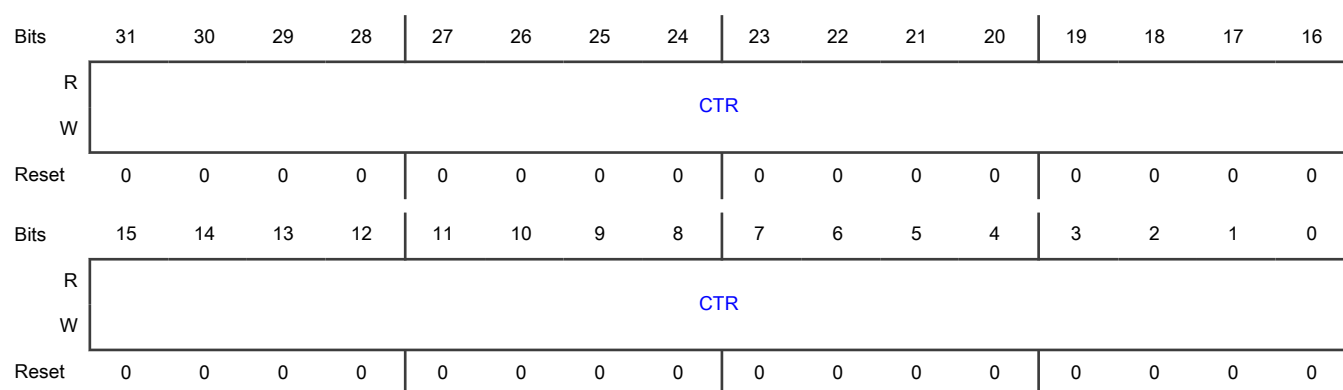
The CTXn_CTRm registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of the counter value for context “n”. There are two consecutive memory-mapped register words defining the upper 96 bits of the counter used for the AES calculations associated with the given context.

The programming model view of the CTXn_CTRm registers is a little-endian 8-element byte data array.

The entire 128-bit counter value is defined as the concatenation of four 32-bit values:

$CTR[127:0] = \{CTR0[C0...C3], CTR1[C4...C7], CTR0[C0...C3] \wedge CTR1[C4...C7], \text{systemAddress}[31-4], 0h\}$

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Diagram**Fields**

Field	Function
31-0	AES Counter
CTR	The upper 64 bits of the counter can be written using the slave peripheral bus. The two consecutive memory-mapped registers directly provide the upper 64 bits of counter storage. Word0: CTR[31:0][C3, C2, C1, C0] Word1: CTR[31:0][C7, C6, C5, C4] The third 32-bit portion of the CTR is formed by exclusive-or'ing the upper 64 bits of the counter as two 32-bit values, while the least-significant portion of the counter is the 32-bit 0-modulo-16 byte system address of the external flash memory. $CTR[C0...C15] = \{CTR[C0...C7], CTR[C0...C3] \oplus CTR[C4...C7], \text{systemAddress}[31-4], 0h\}$

40.7.1.6 AES Region Descriptor Word0 (CTX0_RGD_W0 - CTX3_RGD_W0)**Offset**

Register	Offset
CTX0_RGD_W0	D18h
CTX1_RGD_W0	D58h
CTX2_RGD_W0	D98h
CTX3_RGD_W0	DD8h

Function

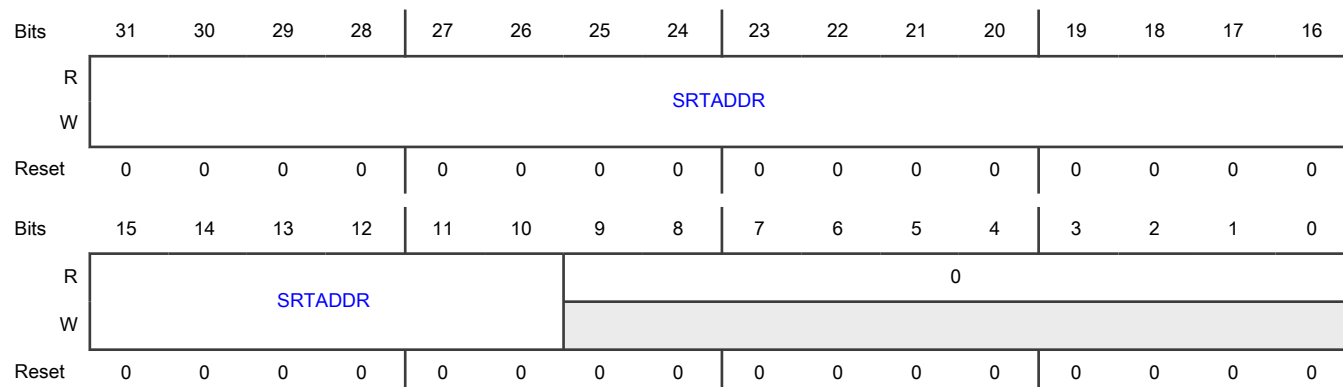
The CTXn_RGD_W0 registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of memory region descriptor for context “n”. There are two consecutive memory-mapped register words defining the starting and ending addresses for the external flash memory region associated with the given context.

Each memory region context defines a specified section of external flash memory associated with the given context and its associated 128-bit key and 128-bit counter values.

The context memory regions are defined as modulo-1024 bytes to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. As a result, no AHB command, either single transfer, or any type of wrapping or incrementing burst, can cross any 1 KB boundary. This means that any AHB transfer is limited to a single context for purposes of AES decryption.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Diagram



Fields

Field	Function
31-10 SRTADDR	Start Address This field defines the most significant bits of the 0-modulo-1024 byte start address of the memory region for context n.
9-0 —	Reserved

40.7.1.7 AES Region Descriptor Word1 (CTX0_RGD_W1 - CTX3_RGD_W1)

Offset

Register	Offset
CTX0_RGD_W1	D1Ch
CTX1_RGD_W1	D5Ch
CTX2_RGD_W1	D9Ch
CTX3_RGD_W1	DDCh

Function

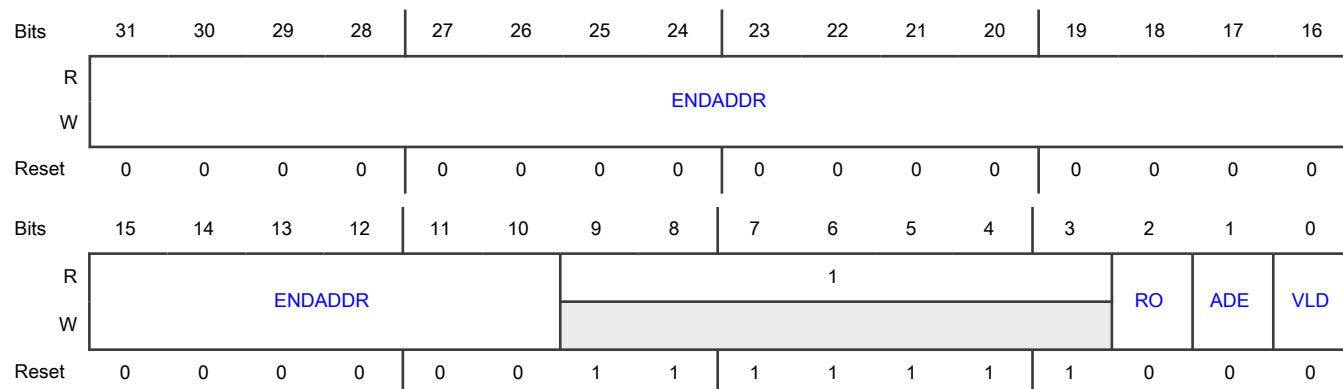
The CTXn_RGD_W1 registers provide a 2-dimensional data structure for local OTFAD storage of 64 bits of memory region descriptor for context “n”. There are two consecutive memory-mapped register words defining the starting and ending addresses for the external flash memory region associated with the given context.

Each memory region context defines a specified section of external flash memory associated with the given context and its associated 128-bit key and 128-bit counter values.

The context memory regions are defined as modulo-1024 bytes to match the AMBA-AHB protocol requirement that no bursting transfer cross that boundary. As a result, no AHB command, either single transfer, or any type of wrapping or incrementing burst, can cross any 1 KB boundary. This means that any AHB transfer is limited to a single context for purposes of AES decryption.

If SR[RRAM] = 1 indicating restricted register access mode, access to these registers is treated as read-as-zero, write-ignored (RAZ/WI).

Diagram



Fields

Field	Function
31-10 ENDADDR	End Address This field defines the most significant bits of the 1023-modulo-1024 byte end address of the memory region for context n.
9-3 —	Reserved
2 RO	Read-Only This field signals that the entire set of context registers (CTXn_KEY[0-3], CTXn_CTR[0-1], CTXn_RGD_W[0-1] are read-only and cannot be modified. This field is sticky and remains asserted until the next system reset. SR[RRAM] provides another level of register access control and is independent of the RO indicator. 0b - The context registers can be accessed normally (as defined by SR[RRAM]). 1b - The context registers are read-only and accesses may be further restricted based on SR[RRAM].
1 ADE	AES Decryption Enable. For accesses hitting in a valid context, this bit indicates if the fetched data is to be decrypted or simply bypassed. 0b - Bypass the fetched data. 1b - Perform the CTR-AES128 mode decryption on the fetched data.
0	Valid

Table continues on the next page...

Table continued from the previous page...

Field	Function
VLD	This field signals if the context is valid or not. 0b - Context is invalid. 1b - Context is valid.

40.7.2 Data organization and endianness considerations

OTFAD implements a little-endian memory organization. This affects the data organization of its programming model registers as well as how it handles memory data.

The OTFAD module follows the standard AES definition on data organization and mapping into the bit-oriented cryptographic algorithms. To review these definitions, recall the following:

"The basic unit for processing the AES algorithm is a byte, a sequence of eight bits treated as single entity. The input, output and Cipher Key bit sequences... are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes" [1, pp. 8-9].

Further, if an input, output or key is denoted by the AES standard convention as "a", the bytes in the resulting array can be described as $a[n]$, where $0 < n < 16$. The mapping of this byte array data into the required 128-bit sequence produces:

```
data[127:0] = {a[0], a[1], a[2], ..., a[14], a[15]}
```

That is, the 128-bit data is formed as the concatenation of the 0th through the 15th byte array element. The OTFAD hardware performs all the required byte "swapping" to convert the programming model view of the keys and counters, etc. from the little endian 32-bit view into the organization required by the AES standard. As an example, consider the mapping of the 128-bit key from its 32-bit programming model registers for the OTFAD AES engine. Let the little-endian memory image of the 128-bit key be defined as a byte-sized, sixteen element array, $a[16]$. Note $A[16]$ is intended to be an alternate, but equivalent, description:

```
AES Key Word 0[31:0] = {a[ 3], a[ 2], a[ 1], a[ 0]}
```

```
AES Key Word 1[31:0] = {a[ 7], a[ 6], a[ 5], a[ 4]}
```

```
AES Key Word 2[31:0] = {a[11], a[10], a[ 9], a[ 8]}
```

```
AES Key Word 3[31:0] = {a[15], a[14], a[13], a[12]}
```

This data is converted by the OTFAD into a 128-bit key of the form:

```
AES Key[127:0] = {a[0], a[1], a[2], a[3], ..., a[12], a[13], a[14], a[15]}
```

Next, consider the 64-bit counter as an eight element byte array, $c[8]$:

```
AES Counter Word 0[31:0] = {c[ 3], c[ 2], c[ 1], c[ 0]}
```

```
AES Counter Word 1[31:0] = {c[ 7], c[ 6], c[ 5], c[ 4]}
```

Before being used, this data is reorganized into the required 128-bit counter value of the form:

```
AES Counter[127:0] = {c[ 0], c[ 1], c[ 2], c[ 3], // bits[127:96]
                      c[ 4], c[ 5], c[ 6], c[ 7], // bits[ 95:64]
                      {c[ 0], c[ 1], c[ 2], c[ 3]}}
```

```
^ {c[ 4], c[ 5], c[ 6], c[ 7]}, // bits[ 63:32]

sysAddr[31:4], 4'b0000} // bits[ 31: 0]
```

To summarize, the OTFAD programming model registers are organized as little-endian byte-size data arrays and the hardware automatically reorganizes the register data into the 128-bit formats required by AES.

Note that since OTFAD is a little-endian design, it processes each 64 bits of external memory data fetched by the FlexSPI using the following mapping for data byte array:

```
mem_wr_data[63:0]

= {data[i+7],data[i+6],data[i+5],data[i+4],data[i+3],data[i+2],data[i+1],data[i+0]}
```

Chapter 41

Peripheral Bridge (AIPS_Lite)

41.1 Chip-specific AIPS-Lite information

Table 455. Reference links to related information

Topic	Related module	Reference
Full description	AIPS-Lite	AIPS-Lite
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

41.1.1 Module instances

This device has two instances of the AIPS module, AIPS0 and AIPS1.

41.2 Overview

AIPS_Lite converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

This peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 4 KB each. All the peripherals may not be used. See the memory map chapter for details on slot assignments. The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

41.2.1 Features

Following are the key features of the peripheral bridge:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

41.2.2 General operation

The slave devices connected to the peripheral bridge are modules that contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 4-KB boundaries. Each peripheral is allocated one or more 4-KB block(s) of the memory map.

41.3 Functional description

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

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

41.3.1 Access support

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

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

41.3.2 Clocking

This module has no clocking considerations.

41.3.3 Interrupts

This module has no interrupts.

41.4 External signals

This module has no external signals.

41.5 Memory map and register definition

The AIPS module(s) on this chip do(es) not contain any user-programmable registers.

Chapter 42

Messaging Unit (MU)

42.1 Chip-specific MU information

Table 456. Reference links to related information

Topic	Related module	Reference
Full description	MU	MU
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

42.1.1 Module instances

This device has one instance of the MU module.

42.2 Overview

The Messaging Unit module (MU) enables two processors within the SoC to communicate and coordinate by passing messages (e.g. data, status and control) through the MU interface. The MU also provides the ability for one processor to signal the other processor using interrupts.

Because the MU manages the messaging between processors potentially using different clocks, the MU must synchronize the accesses from one side to the other. The MU accomplishes synchronization using two sets of matching registers (Processor A-facing, Processor B-facing).

42.2.1 Block Diagram

The following is the MU Block Diagram:

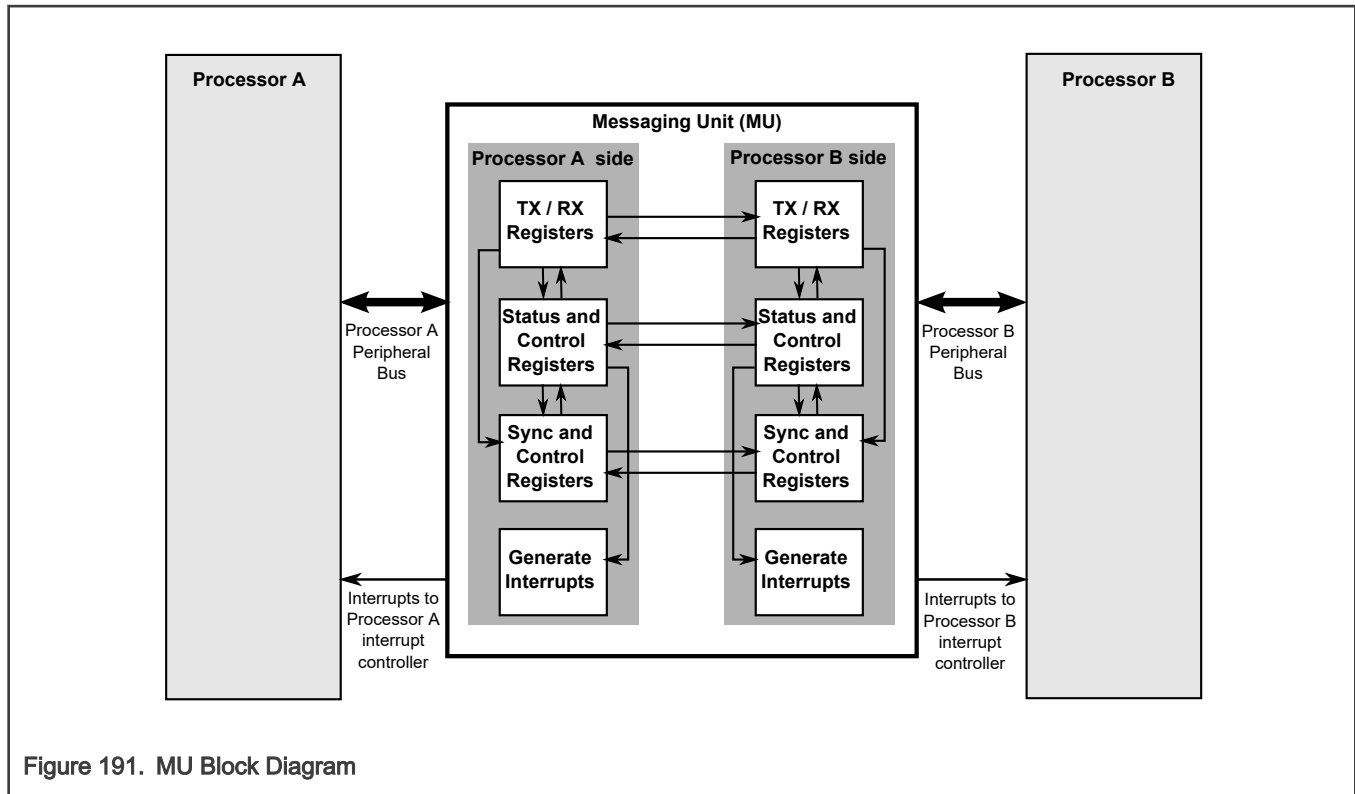


Figure 191. MU Block Diagram

42.2.2 Features

The MU includes the following features:

- Messaging control by interrupts or by polling
- Symmetrical processor interfaces with each side supporting the following:
 - Three general-purpose flags reflected to the other side
 - Four general-purpose interrupt requests reflected to the other side
 - Four receive registers with maskable interrupt
 - Four transmit registers with maskable interrupt
- The Processor B can take the Processor A out of low-power modes by asserting one of the interrupts to the Processor A and vice versa

NOTE

MU chapter references to Processor A may also correspond to Core 0 and references to Processor B may also correspond to Core 1 in other parts of this document.

42.3 Functional Description

Table 457. Major Features of the MU

Major Feature	Description
Interprocessor Interrupts	The MU has 12 interrupt sources on each side (Processor A-side, Processor B-side) that are used for signaling the other processor. The interrupts can be used for notification of RX/TX events and general-purpose signaling between the processors.
MU Reset	- The Processor A can issue a reset to the entire MU, using a control bit (MUR) in the Processor A Control Register (CR). - The MUR bit is a self-clearing bit.
Status and Control Communications between Cores	- The MU provides a way for the two cores to communicate using the status and control registers present on both the Processor B and Processor A sides of the MU. - The status register of one MU side reflects the status of the other MU side. - The control register is used for control operations, such as enabling an interrupt and sending an interrupt to the other processor.
Synchronized Message Transfers between Cores	- The transfer of data messages between cores uses transmit empty and receive full flags provided on both sides of the MU. - The update of these transmit and receive flags is accomplished using a synchronization mechanism. There is inherent latency between updating the flag on one side and reflecting its status on other side. For more about latency, see Event Update Timing
Accessing Shared Memory Directly and Avoiding Collisions	- For sending data or messages from one MU-side to the other MU-side, the MU provides 4 transmit registers and 4 receive registers on each side of the MU. - The Processor A or Processor B can access shared memory resources of the SoC directly. However, to avoid simultaneous access to shared memory by both cores, the MU provides a method (to prevent simultaneous access) using interrupts and transmit-receive registers for both processors.
Memory-Mapped Registers	The MU is connected as a peripheral under the Peripheral bus on both sides—on the Processor A-side, the Processor A Peripheral Bus, and on the Processor B-side, the Processor B Peripheral Bus.

42.3.1 Modes of Operation

The MU supports the modes described in the indicated sections:

- [Low Power Modes](#)

42.3.2 Processor A Side Memory-Mapping

The messaging, control, and status registers of the Processor A-side for the MU are mapped to the Processor A memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

42.3.3 Processor B Side Memory-Mapping

The messaging, control, and status registers of the Processor B-side for the MU are mapped to the Processor B memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

42.3.4 MU Messaging

The MU provides 32-bit status and control registers to the Processor B and Processor A sides for control operations (such as interrupts and reset), and for status checking of the other MU-side.

For messaging, the MU has four, 32-bit write-only transmit registers and four, 32-bit read-only receive registers on the Processor B and Processor A-sides. These registers are used for sending messages to each other. These messages can also be controlled using the 3 general purpose flags provided in the control and status registers of either MU-side.

42.3.4.1 Programmer Model

The messaging logic is used in conjunction with external memory. There are various messaging methods, which can be used to implement a messaging protocol. Some of these messages could mean "A message of N words has been written, starting at offset X in the memory," or "the previous data block that was sent has finished being read." Having the messaging logic independent from the memory array is not restricted to a predefined hardware protocol. On the other hand, the software needed to manage the messaging is short and straightforward.

Most of the messaging mechanisms are symmetric; they are duplicated and are available on both the Processor B-side and the Processor A-side. The messaging mechanisms are:

- Four, 32-bit write-only transmit registers, which are each reflected in four, read-only receive registers in the other processor's side. These registers can be used to transfer 32-bit word messages or frame information of messages written to the shared memory (number of words, initial address, and message type code).
- A write to a transmit register on the transmitter side clears a "transmitter empty" bit in the Status Register on the transmitter side, and sets a "receiver full" bit in the Status Register on the receiver side. The setting of the bit at the receiver side can optionally trigger an interrupt at the receiver side (maskable receive interrupt).
- A read of one of the receive registers at the receiver side clears the "receiver full" bit in the Status Register at the receiver side, and sets the "transmitter empty" bit in the Status Register on the transmitter side. The setting of the "transmitter empty" bit can optionally trigger an interrupt at the transmitter side (maskable transmit interrupt).
- Four general purpose flags are reflected in the Status Register on the receiver side
- A read/write access to any reserved location and a write to a read-only register on the Processor A-side of the MU generates a module transfer error acknowledge to the Processor A.
- A read/write access to any reserved location and write to a read-only register on the Processor B-side of the MU generates a module transfer error acknowledge to the Processor B.

42.3.5 Low Power Modes

This section describes the low power operating modes of the MU module.

42.3.5.1 Processor Low Power Modes

The Processors have four power modes:

- Run
- WAIT
- STOP/VLPS
- LLS/VLLS

The Processor can be awakened from a low-power mode by any enabled Processor side MU interrupt, as reflected in the xSR "status" register (RF0–3, TE0–3, GIP0–3 bits are set) and enabled in the xCR control register. Using these bits, the Processor can actively control when to wake the other Processor.

While the Processor is in STOP/VLPS mode (such that the xSR register bits cannot be updated with events), special logic drives the enabled Processor interrupts directly from the other Processor-side (instead of from the xSR register).

While the Processor is in STOP/VLPS mode, the asynchronous Processor interrupt asserts to wake the Processor:

- If any transmit data register of the other Processor-side is full, because of a write to it (transmit data register); that is, its “empty” bit in the xSR register is cleared while its corresponding receive interrupt is enabled on the Processor-side.
- If any receive data register of the other Processor-side is empty, because of a read on the other Processor -side; that is, its “full” bit in the xSR register is cleared while its corresponding transmit interrupt is enabled on the Processor-side.
- If any general purpose interrupt is set in the xCR register while the corresponding interrupt is enabled on the Processor-side.
- If the other Processor issues a non-maskable interrupt to the Processor.

The logic enables the other Processor to operate independently while the Processor is in any power mode (including STOP/VLPS). However, the Processor power mode change protocol should be handled with care regarding:

- The interrupts that are enabled on the Processor-side
- The events that could be triggered by the other Processor-side
- The compatibility with the other Processor protocol of entering STOP/VLPS mode

If the Processor is in STOP/VLPS mode and an event on the other Processor is triggered, the EP bit (in the xSR register) remains high until the Processor wakes up.

Before entering STOP/VLPS mode, the Processor programmer should verify that the EP bit (in the xSR register) is cleared. This check is needed to ensure that all pending updates from the Processor, including the power mode change when STOP/VLPS or WAIT is executed, updates in the xSR register.

- If the other Processor is in STOP/VLPS mode or LLS/VLLS mode, the EP bit (in the xSR register) may be stuck high; in this case, the Processor need not check the EP bit before entering STOP/VLPS mode.

42.3.6 Event Update Timing

Each processor’s MU messaging side (Processor A or Processor B) has a hardware mechanism to send “event update requests” to the other processor’s side. An “event” is considered when any information change should be reflected at the Status Register of the receiving processor. The event update latency is the delay between the event being ready at one processor and the resulting update at the Status Register of the other processor.

- The minimum event latency is “1 clock of the sending side” + “2 1/2 clocks of the receiving side”. The minimum case is if there is no event pending when the new event occurs.
- The maximum event latency is “6 clocks of the sending side” + “6 1/2 clocks of the receiving side.” The maximum case is if the event occurred just after a previous event was sent to the other side.

The event update latency varies between the above-mentioned minimum and maximum latencies, depending on the time at which the subsequent event is triggered.

42.3.7 Interrupts

The MU controls the Processor B interrupt requests to the Processor A, and the Processor A interrupt requests to the Processor B.

This section describes all the interrupts that the module generates.

42.3.7.1 Interrupts to the Processors

There are 12 interrupt sources from the MU to the Processors:

- Four receive interrupts (asserted when the Processors receive full bits are set and enabled in the xCR register) for each of the receive registers
- Four transmit interrupts (asserted when the Processor transmit empty bits are set and enabled in the xCR register) for each of the transmit registers
- Four general purpose interrupts (asserted when the GIP bits are set and enabled in the xCR register)

All the interrupts are maskable in the Processor Control Register (xCR). The MU does not assume any internal priority of these interrupts. Multiple interrupts (for example, Receive 0 and Receive 1 interrupts or any of the transmit and general purpose

interrupts) can be asserted at one time. The priority of these interrupts should be resolved by the interrupt controller at the chip level.

The General Purpose Interrupt Pending bits (GIP0, GIP1, GIP2, and GIP3) should be cleared by the software (as part of the interrupt service routine) to de-assert the request to the interrupt controller.

42.3.7.2 General-purpose Interrupt Clearing Sequence

When a Processor writes to the general interrupt bit (GIR), the write event is synchronized to the other Processor clock to set the general interrupt request pending bit (GIP). When the GIP bit is set, and if the general purpose interrupt is enabled on the transmitting Processor side (GIE bit is set), then the receiving Processor general purpose interrupt is issued to the transmitting Processor. The transmitting Processor clears this interrupt by writing a “1” on the GIP bit. The interrupt is de-asserted as soon as the GIP bit is written. The write event of the GIP bit is synchronized to the other Processor clock. The synchronized signal clears the GIR bit. The software should not write the GIR bit again until the GIR bit is cleared.

42.3.8 Resets

- The MU has two sources of reset, and each reset has a different function from the MU or system perspective.
- One asynchronous system that is connected to both sides of the MU interface.
 - One programmable hardware reset (MUR bit) in the CR register (on the Processor A-side).

Table 458. MU programmable resets

Reset	Description
Processor A MU reset	• Processor A MU Reset bit (MUR) of the CR register • The MUR reset affects the messaging section on both the Processor A and the Processor B sides. The MUR reset causes all control and status registers to return to their default values and all internal states to be cleared. • It is up to the Processor A software to decide whether to use the MUR reset or not. • The instruction immediately following assertion of the MUR bit should not write to MU registers. Such a write may be overwritten by the reset sequence and the register remains with the reset value. A delay of at least one instruction (after assertion of the MUR bit) should occur before attempting a write to MU registers.

After issuing MUR bit reset events, the Processor A programmer can verify that the reset sequence on the Processor B-side has ended, by checking the RS bit in the SR register.

NOTE

MUR bit assertion is a delicate operation because it affects the other side’s registers asynchronously. MUR bit assertion may cause unpredictable behavior if, for example, the Processor B is concurrently testing an MU register bit (TE bit in Processor B SR register). Before asserting the MUR bit, you should verify that the Processor B is not presently engaged in an MU signalling activity.

42.4 External Signals

There are no Messaging Unit signals directly available externally.

42.5 Application Information

42.5.1 Messaging Examples

The following are messaging examples:

- **Passing short messages:** Transmit register(s) can be used to pass short messages from one to four words in length. For example, when a four-word message is desired, only one of the registers needs to have its corresponding interrupt enable bit set at the receiver side; the message's first three words are written to the registers whose interrupt is masked, and the fourth word is written to the other register (which triggers an interrupt at the receiver side).
- **Passing frame information:** Transmit registers can be used to pass frame information for long messages written to the shared system memory. Such frame information normally includes a start address, number of words, and perhaps a message type code.
- **Passing event notices and requests:** Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the general interrupts, such as acknowledging that a long message was read from the shared system memory.
- **Passing fixed length data:** Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general interrupt (Processor A or Processor B) to signal the other processor that the data is ready.
- **Passing announcements:** The three flags can be used by a processor to announce its current program state or other billboard messages to the other processor.

Figure 194 shows the MU registers schematic.

42.5.1.1 Messaging Protocols using Interrupts

The example below describes a four-word messaging sequence sent between the processors.

The registers are written sequentially for $n = 0, 1, 2, 3$. For $n = 0, 1, 2$, the interrupts are disabled, therefore no interrupt goes to the processor (although interrupt conditions occur). For $n = 3$, the interrupt is enabled, and the last Receive Interrupt request is generated.

1. Write Sequence

- The Transmitting Processor writes the message information sequentially to its Transmit Registers 0, 1, 2
- When the write to the Transmit Register 3 occurs, the SR[RF3] bit is set after synchronization, and it immediately trigger the Receive Full 3 interrupt to the Receiving Processor

2. Read Sequence

- The Receiving Processor receives the Receive Full 3 interrupt and starts reading the message transferred from the receive registers
- After Receive Register 3 is read, the interrupt SR[RF3] bit is cleared

The following table and diagram describe the messaging model using transmit/receive registers and interrupt messaging protocol.

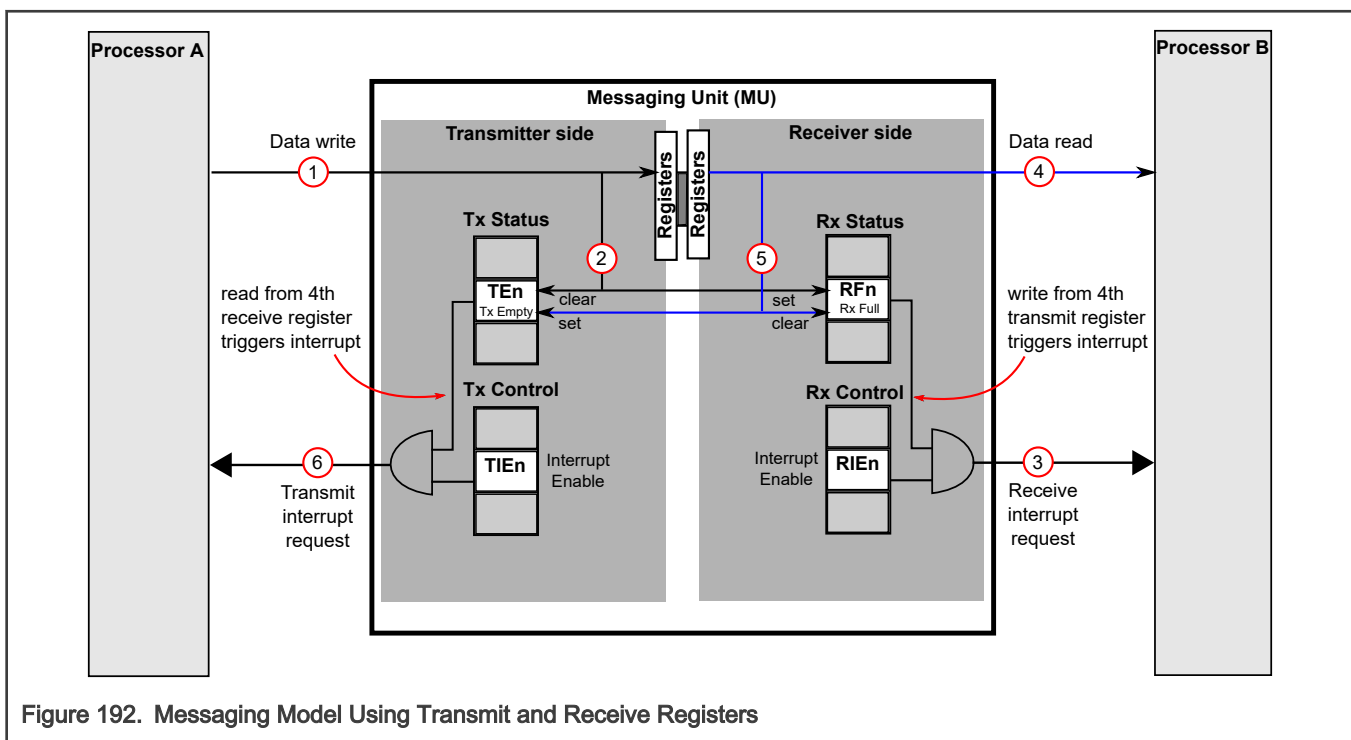
Table 459. Interrupt Messaging Protocol (Generalized)

Step	Action	Description
1	Processor A Data write	A data write to the TRn register by Processor A is reflected in the Processor B RRn register.
2	Clear Tx Empty bit and Set Rx Full bit	The data write to the TRn register • Clears the transmitter empty bit (TEn) in the Processor A Transmit Status Register • Sets the receiver full bit (RFn) in the Processor B Receive Status Register
3	Generate Receive Interrupt request	The setting of the receiver full bit (RFn) in the Receive Status Register generates a Receive Interrupt request to Processor B.

Table continues on the next page...

Table 459. Interrupt Messaging Protocol (Generalized) (continued)

Step	Action	Description
4	Processor B Data read	After receiving the Receive Interrupt request, Processor B performs a data read of the RRn register.
5	Clear Rx Full bit and Set Tx Empty bit	Reading the data out of the RRn register - Clears the receiver full bit (RFn) in the Processor B Receive Status Register - Sets the transmitter empty bit (TEn) in the Processor A Transmit Status Register
6	Generate Transmit Interrupt request	The setting of the transmitter empty bit (TEn) in the Transmit Status Register generates a Transmit Interrupt request to Processor A.



The messaging hardware can be used by software to implement messaging protocols for a wide array of message types. Full support is given for both interrupt and polling management schemes.

42.5.1.2 Messaging Protocols using Event Interrupts

Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the two general interrupts.

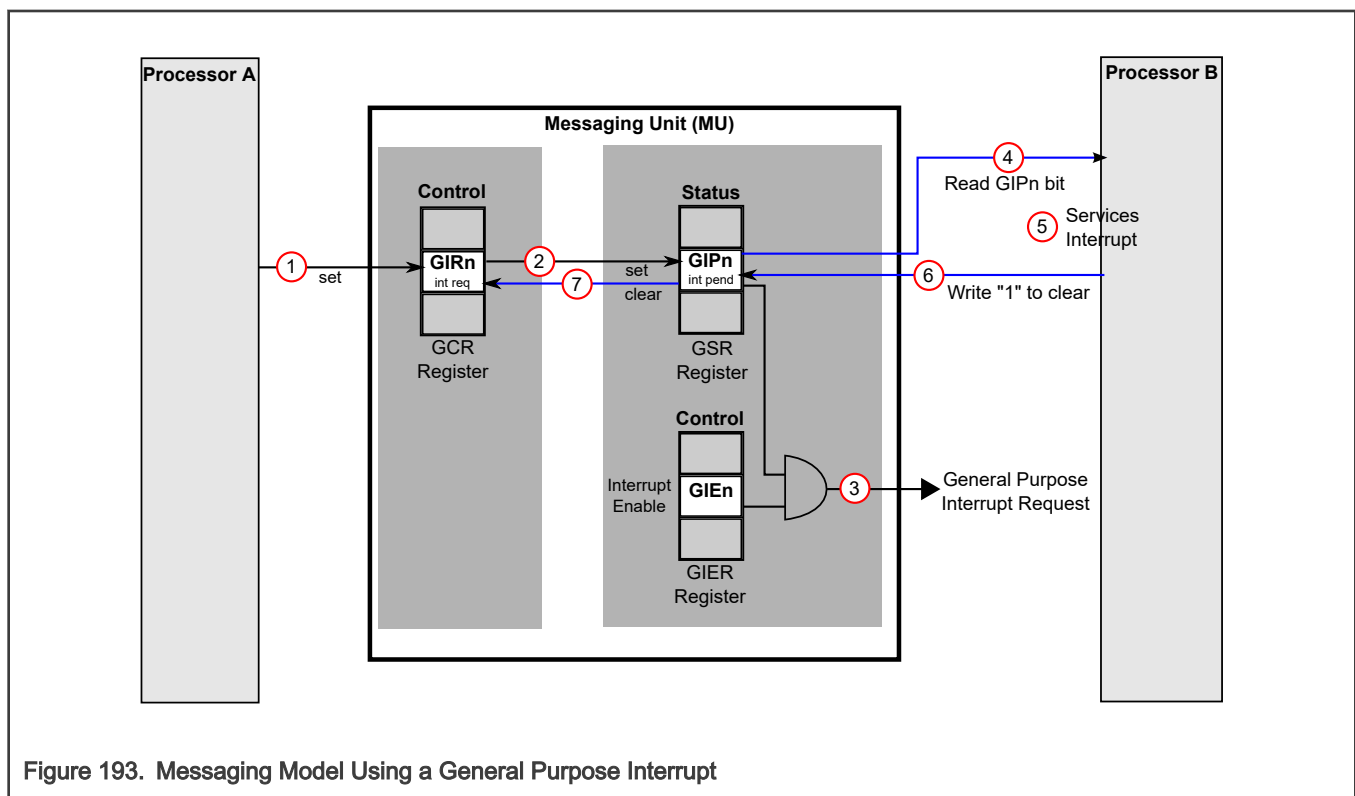
Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general purpose interrupt to signal the other processor that the data is ready.

The three flags can be used by a processor to announce to the other processor the program state it is currently in, or to announce similar messages.

Table 460 and Figure 193 describe the event steps when the Processor triggers an interrupt.

Table 460. Interrupt Messaging Protocol (Generalized)

Step	Action	Description
1	Processor A sets General Interrupt request bit	Processor A sets its associated General Interrupt request bit (GIRn = 1) in the control register (ACR).
2	General Interrupt Request Pending status bit is set	The General Interrupt Request Pending status bit (GIPn) in the status register (BSR) is set to "1"
3	General Interrupt request to Processor B is generated	Setting the GIPn bit generates the General Interrupt request to Processor B (Interrupt Request Enable bit, GIEn, must be set for Processor B)
4	Processor B reads status register	The Processor B reads the GIPn bit in the BSR register.
5	Processor B services the interrupt	-
6	Processor B sets GIPn bit to clear interrupt	The Processor B writes "1" to the corresponding GIPn bit to clear the interrupt
7	GIRn bit is cleared	Setting the GIPn bit to "1" clears the General Interrupt request bit (GIRn) in the Processor A control register (ACR).



42.5.1.3 Exclusive Access to Shared Memory

MU can be used to signal one processor about its current access to the shared memory, so that the data is not overwritten by the other processor during the exclusive memory access period.

The following tables describe the signaling protocol that the Processor A uses to inform the Processor B about its current access (write) to the shared memory, assuming that the set of bits and registers (GIRn bit, RRn register, TRn register) are reserved to support exclusive access to the shared memory protocol.

Table 461. How the Processor A Performs an Exclusive Access to Shared Memory

Step	Action	Description
1	Processor A sends GIRn request to Processor B using Processor A control register	When the Processor A wants to perform an exclusive access to the shared memory, the Processor A sends an GIR0 request to the Processor B.
2	Processor A sends an exclusive-access request using a transmit data register (TRn)	The Processor A sends an exclusive-access request (command, location, and length of target access) to Processor B using a selected transmit data register (TR0).
3	Processor A waits for a dedicated interrupt from Processor B	The Processor A waits for a dedicated interrupt (as an acknowledgement) triggered by the Processor B before proceeding.
4	Processor A accesses shared memory	After receiving a dedicated interrupt from the Processor B, Processor A proceeds.

Table 462. How the Processor B Scans for Transaction Information

Step	Action	Description
1	Processor B receives an interrupt from a receive data register (RRn)	-
2	Processor B reads the receive data register (RRn)	-
3	Processor B scans the receive data register contents	For transaction information (whether Processor A has requested an exclusive-access).

Table 463. How the Processor B Accepts Exclusive Access by Processor A

Step	Action	Description
1	Processor B triggers a dedicated interrupt	Processor B acknowledges the Processor A request by triggering a dedicated interrupt (ack) to the Processor A.
2	Processor B sends a code message to Processor A	Along with the acknowledge interrupt, the Processor B sends a code message to the Processor A through the selected transmit register (TRn). The message informs the Processor A that it can exclusively access the shared memory.

Table 464. How the Processor B Rejects Exclusive Access by Processor A

Step	Action	Description
1	Processor B ignores Processor A request for exclusive access	If the Processor B does not want to give go-ahead permission to the Processor A, Processor B ignores the exclusive access request.

42.5.1.4 Packet Data Transfers

The following example describes the packet transfer sequence between the Processor B and Processor A subsystems:

Table 465. Packet Data Transfer Sequence

Step	Action	Description
1	Processor B requests DMA	The Processor B sends a DMA request to initiate the packet data transfer.
2	DMA data transfer	DMA acknowledges.
3		DMA starts transferring data from the specified Processor B memory location to the specified shared memory
4		DMA interrupts the Processor B to signal that the packet transfer has finished.
5	Processor B informs Processor A that data is in shared memory	Using an MU Processor B-side transmit register, the Processor B sends a packet information message to the Processor A to inform the Processor A of the arrival of new packet data that is stored in shared memory. The message contains the command, location, and length of packet data information.
6	Processor A receives interrupt	The Processor A receives an interrupt (assuming its corresponding Processor A MU-side receive interrupt is enabled), and the pending processing task becomes active and processes packet data from memory.
7	Processor A reads data, writes data	The Processor A reads or processes packet data from shared memory.
8		The Processor A writes the result from packet processing to a separate buffer.
9	Processor A informs Processor B that transfer is finished	After the processing of the packet data finishes, the Processor A informs the Processor B (using the MU Processor A-side transmit register, ATRn).
10	Processor A sends interrupt to Processor B (request for more data)	The Processor B receives the next interrupt from the Processor A, in which the Processor A requests more packet data.

42.5.2 Software Restrictions

This section describes certain software restrictions when accessing the MU.

42.5.2.1 General Restrictions

This section lists the restrictions that apply to both the sides (Processor A, Processor B) of the MU.

42.5.2.1.1 Write-After-Write to a Transmit Register

A write to a transmit register signals the receiver side that data is ready for retrieval.

- Writing to the transmit register again without verifying that the data was retrieved is prohibited, because the transmitter side has no way of knowing the exact time that the receiver attempts to retrieve the data.
- Before attempting to write the transmit register again, the transmitter side should wait for a “Transmitter Empty” interrupt, or should poll the “Transmitter Empty” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being read on the receiver side of the MU.

42.5.2.1.2 Read-After-Read from a Receive Register

A read of a receive register signals the transmitter side that data can be written to that register. In the same way, the receiver processor should not read a receive register before receiving a “Receiver Full” interrupt or polling the “Receiver Full” bit in the Status Register.

- Reading the receive register again without verifying that the data was written is prohibited, because the receiver side has no way of knowing the exact time that the transmitter attempts to write the data.
- Before attempting to read the receive register again, the receiver side should wait for a “Receiver Full” interrupt, or should poll the “Receiver Full” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being written on the transmitter side of the MU.

42.5.2.2 Processor Restrictions

This section lists the restrictions that apply each side of the processor in the MU.

42.5.2.2.1 Before Entering Low Power Mode

Before entering Low Power mode, the Processor should verify that the Processor Event Pending (EP) bit in the Status Register is cleared.

- If the Event Pending bit (EP) is still set to “1”, then the Processor should wait and poll the EP bit until it is cleared, before executing the LPM instruction.
- Note that if the other Processor is in Low Power mode, the EP bit may be stuck high. In this case, the other Processor clock must be turned ON to get the EP bit cleared before the Processor can enter Low Power mode.
- To discover which power mode the other Processor is in, the Processor can check the PM bits in the xSR register.

42.5.2.2.2 Before Setting a General Interrupt Request Bit (GIR0–3)

Before setting a General Interrupt Request bit (GIR0–3), the user must verify that the GIRn bit is cleared, which means that a general interrupt is not pending. Generally, setting the GIRn bit while the bit is set to “1” is ignored, but in some cases it may issue a second interrupt. This restriction is meant to prevent this indeterministic behavior.

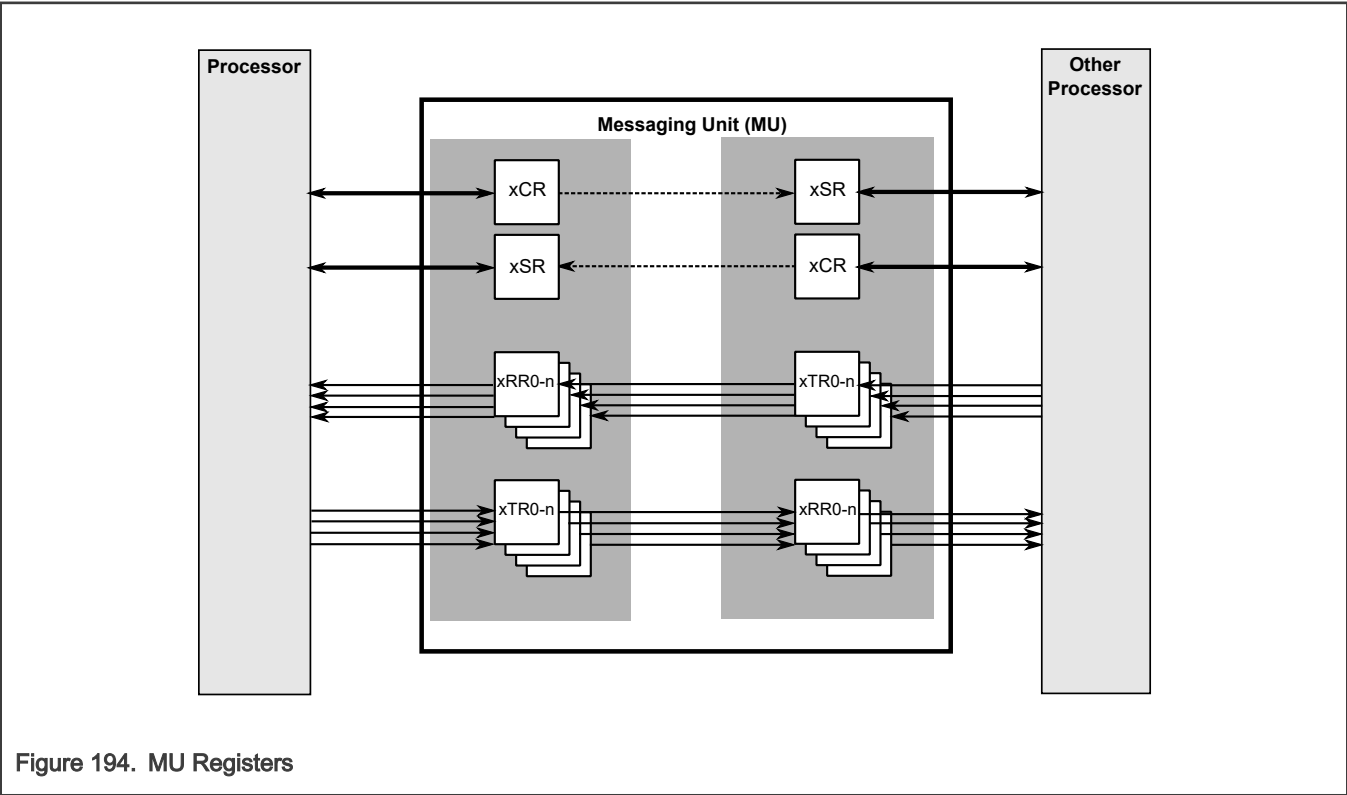
42.5.2.2.3 Reset Bit Restrictions

The reset bit (MUR, HR) restrictions are:

- Before asserting the MUR bit in the CR register, verify that the Processor B-side is not engaged in some MU activity.
- Do not write to an MU register in the instruction immediately after the assertion of the MUR bit in the CR register, because the written data can be overridden by the reset value.

42.6 Register Definition

The MU provides transmit and receive data registers (xTR0-n, xRR0-n) for the communication between Processor A and Processor B. Some control and status registers (xCR, xSR) to the Processor A and Processor B sides for control operations (such as interrupts and reset), and for status checking of the other MU-side. [Figure 194](#) shows the MU registers schematic.



The detailed MU Register Definition can be found below.

NOTE
Writing to an RRn register can cause a transfer error.

42.6.1 MUA register descriptions

This section contains the detailed register descriptions for the MUA registers.

42.6.1.1 MU memory map

MU0.MUA base address: 4011_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VER)	32	R	0100_0001h
4h	Parameter Register (PAR)	32	R	0000_0000h
20h - 2Ch	Transmit Register (TR0 - TR3)	32	RW	0000_0000h
40h - 4Ch	Receive Register (RR0 - RR3)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
60h	Status Register (SR)	32	RW	00F0_0080h
64h	Control Register (CR)	32	RW	0000_0000h

42.6.1.2 Version ID Register (VER)

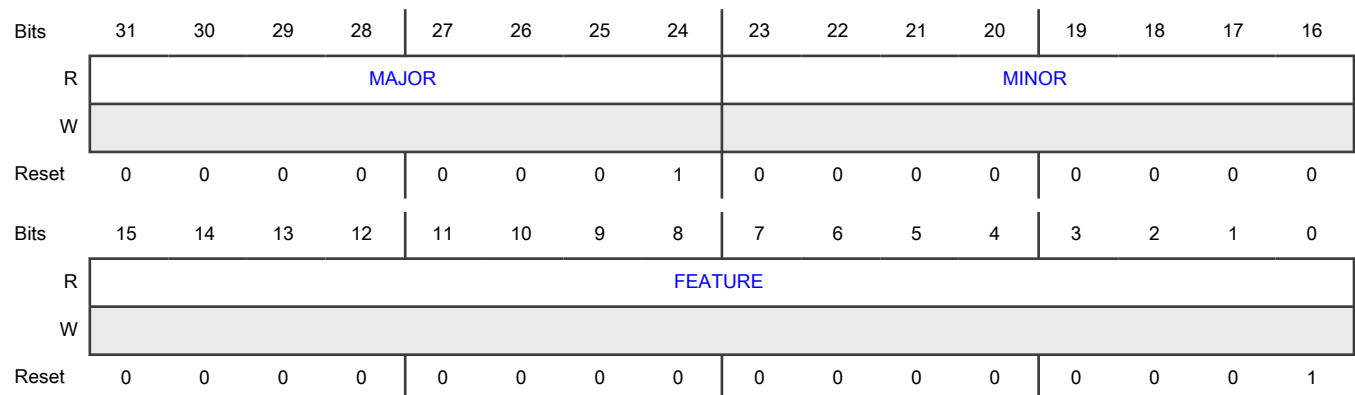
Offset

Register	Offset
VER	0h

Function

Use Version ID register to determine the version ID and feature set number of MUA.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number This read only field returns the major version number for MUA.
23-16 MINOR	Minor Version Number This read only field returns the minor version number for MUA.
15-0 FEATURE	Feature Specification Number This read only field returns the feature set number for MUA.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000_0000_0000_x1xxb - Core Control and Status Registers are implemented in both MUA and MUB. 0000_0000_0000_xx1xb - RAIP/RAIE register bits are implemented. 0000_0000_0000_xxx0b - Standard features implemented

42.6.1.3 Parameter Register (PAR)

Offset

Register	Offset
PAR	4h

Function

Use Parameter register to determine the parameter settings of MUA.

Diagram

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

Fields

Field	Function
31-0 PARAMETER	This bitfield contains the parameter settings of MUA.

42.6.1.4 Transmit Register (TR0 - TR3)

Offset

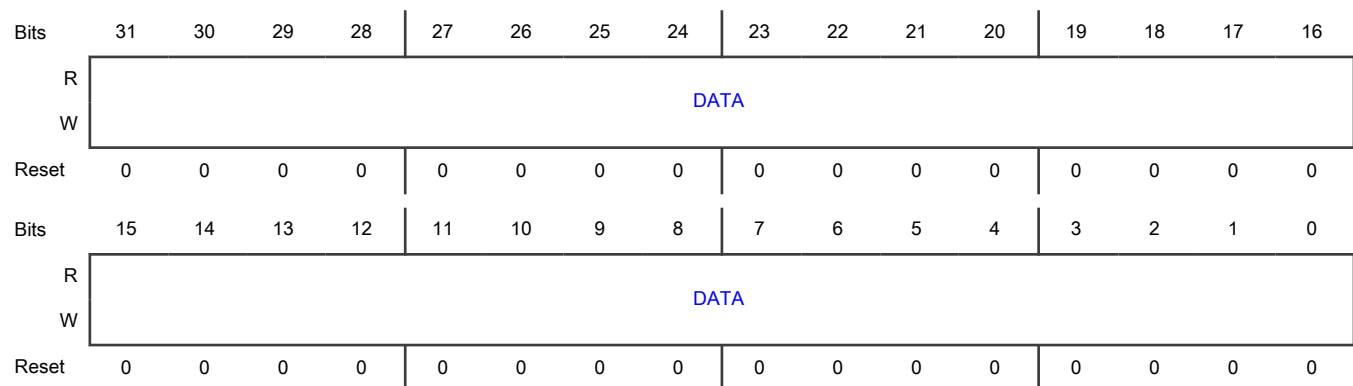
Register	Offset
TR0	20h
TR1	24h
TR2	28h
TR3	2Ch

Function

Use MUA Transmit Register (TRn, 32-bit, write-only) to transmit a message or data to MUB.

- You can only write to the TRn register when the TEn bit in SR register is set to "1".
- Reading the TRn register returns all zeros.

Diagram



Fields

Field	Function
31-0	DATA
DATA	MUA Transmit Register Data. (Write-only) • Data written to the TRn register is reflected in the MUB Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register. • A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUA Status Register (SR) on the transmitter side, and sets a "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUA side). • Any write to the TRn register updates all status information.

42.6.1.5 Receive Register (RR0 - RR3)

Offset

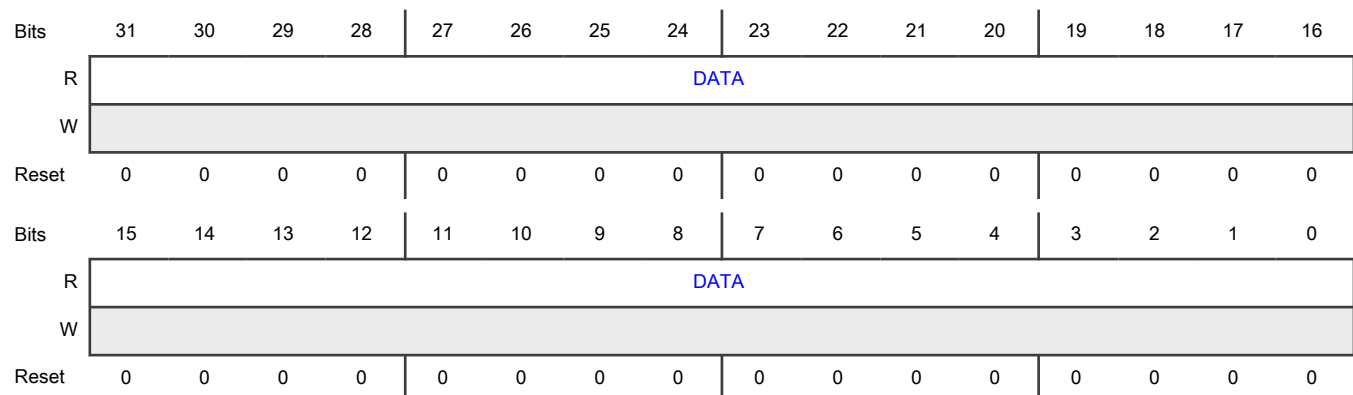
Register	Offset
RR0	40h
RR1	44h
RR2	48h
RR3	4Ch

Function

Use MUA Receive Register (RRn, 32-bit, read-only) to receive a message or data from the MUB.

- Data written to the MUB TRn register is immediately reflected in the MUA RRn register.
- You can only read the RRn register when the RFn bit in the SR register is set to "1".
- Writing to the RRn register generates an error response to the MUA.

Diagram



Fields

Field	Function
31-0	DATA
DATA	MUA Receive Register. (Read-only) • Reflects the data written to MUB Transmit Register 0 (TRn). • Reading the RRn register clears the "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUB Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUB side). • Any read of the RRn register updates all status information.

42.6.1.6 Status Register (SR)

Offset

Register	Offset
SR	60h

Function

Use the Processor A Status Register (SR, 32-bit, read-write) to show interrupt status from the Processor B, general purpose flags, the Processor B power mode, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor A-side programmer
- Other dual-purpose bits are set by the Processor A-side programmer, and cleared by the MU logic.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GIPn				RFn				TEn				Reserved			
W	W1C															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	Reserved			Reserv ed	RAIP	RDIP	FUP	RS	PM		EP	Reserv ed	Fn		
W					W1C	W1C										
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Fields

Field	Function
31-28 GIPn	GIPn For n = {0, 1, 2, 3} MUA General Interrupt Request n Pending. (Read-Write) • GIPn bit signals the MUA that the GIRn bit in the CR register on the MUB side was set from "0" to "1". If the GIEn bit in the MUA CR register is set to "1", a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as "1". Writing "0", or writing "1" when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUA register, set the desired bit, and write it to the MUA SR register, thus clearing the GIPn bit. • GIPn bit is cleared when the MU is reset. 0000b - MUA general purpose interrupt n is not pending. (default) 0001b - MUA general purpose interrupt n is pending.
27-24 RFn	RFn

Table continues on the next page...

Table continued from the previous page...

Field	Function
	For n = {0, 1, 2, 3} MUA Receive Register n Full. (Read-only) - The RFn bit is set to "1" when the MUB TRn register is written on the MUB side. - After the RFn bit is set to "1", the RFn bit signals the MUA side that new data is ready to be read by the MUA in the MUA RRn register, and a Receive n interrupt is issued on the MUA side (if the RIEn bit in the MUA CR register has been set to "1"). - RFn bit is cleared when the MUA RRn register is read, and when the MU is reset. 0000b - MUA RRn register is not full (default). 0001b - MUA RRn register has received data from MUB TRn register and is ready to be read by the MUA.
23-20 TEn	TEn For n = {0, 1, 2, 3} MUA Transmit Register n Empty. (Read-only) - The TEn bit is set to "1" after the MUB RRn register is read on the MUB side. - After the TEn bit is set to "1", the TEn bit signals the MUA side that the MUA TRn register is ready to be written on the MUA side, and a Transmit n interrupt is issued on the MUA side (if the TEn bit in the MUA CR register is set to "1"). - TEn bit is cleared after the MUA TRn register is written on the MUA side. - TEn bit is set to "1" when the MU is reset. 0000b - MUA TRn register is not empty. 0001b - MUA TRn register is empty (default).
19-15 —	Reserved
14-12 —	Reserved
11 —	Reserved
10 RAIP	RAIP Processor B Reset Asserted Interrupt Pending. (Read-Write) - RAIP bit signals the Processor A-side that the Processor B-side has entered reset. - RAIP bit is set to "1" after the MU Processor B-side enters reset (after synchronization). The interrupt generated by a Processor B-side reset assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B-side enters reset, if the interrupt is enabled by the RAIE bit. - To clear the RAIP bit, write "1", which also clears general purpose interrupt 3. - When Processor A-side of MU comes out of reset, the RAIP bit has value "0" (default). Processor A then monitors the reset of Processor B-side and asserts RAIP if its reset asserts.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bitfield is only available on the Processor A side. 0b - Processor B-side did not enter reset 1b - Processor B-side entered reset
9 RDIP	RDIP Processor B Reset De-asserted Interrupt Pending. (Read-Write) - RDIP bit signals the Processor A-side that the Processor B-side has come out of reset. - RDIP bit is set to "1" after the MU Processor B-side comes out of reset, after synchronization. The interrupt generated by a Processor B-side reset de-assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B-side comes out of reset if the interrupt is enabled by the RDIE bit. - To clear the RDIP bit, write "1", which also clears general purpose interrupt 3. - When Processor A-side of MU comes out of reset, the RDIP bit has value "0"(default). Processor A then monitors the reset of Processor B and asserts RDIP reset deasserts. This takes 5-6 clock cycles, so may result in the RDIP bit being asserted after Processor A exits reset. This bitfield is only available on the Processor A side. 0b - Processor B-side did not exit reset 1b - Processor B-side exited from reset
8 FUP	FUP MUA Flags Update Pending. (Read-only) - FUP bit is set to "1" when the MUA side sends a Flags Update request to the MUB side. - A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUA CR register. No flag update changes are allowed while the FUP bit is set to "1". Any write to the Fn[2:0] bits of the MUA CR register, while the FUP bit is set to "1", does not generate a Flags Update event, and the Fn[2:0] bits stays unchanged. - FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU MUB side, and during MU reset. 0b - No flags updated, initiated by the MUA, in progress (default) 1b - MUA initiated flags update, processing
7 RS	RS MUB Reset State. (Read-only) - RS bit indicates if the MUB side of the MU is in a reset state or not. - If the RS bit is set to "1", then the MUB side of the MU is still in the reset state. - If the RS bit is cleared, then the MUB side of the MU are out of reset. - The RS bit is set to "1" during: a MUB system reset, or an MU reset (caused by setting the MUR bit at the CR register).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- The RS bit is cleared when the reset sequence on the MUB side of the MU ends. After issuing any of the reset events mentioned previously, you should verify that the RS bit is cleared before starting any accesses. - When the MUB processor comes out of reset, the RS bit has value "1"(default). 0b - The MUB side of the MU is not in reset. 1b - The MUB side of the MU is in reset.
6-5 PM	PM Processor B-side Power Mode. (Read-only) - PM[1:0] bits indicate the Processor B-side power mode. NOTE The Processor B Power Mode is platform-specific. 00b - The MUB processor is in Run Mode. 01b - The MUB processor is in WAIT Mode. 10b - The MUB processor is in STOP/VLPS Mode. 11b - The MUB processor is in LLS/VLLS Mode.
4 EP	EP MUA Side Event Pending. (Read-only) - EP bit is set to "1" when the MUA side mechanism sends an event update request to the MUB side. - EP bit is cleared when the event update acknowledge is received. An "event" is any hardware message that is reflected in the MUB SR register on the MUB side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. - To ensure events have been posted to MUB before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to "1", you should wait and continue to poll it (EP bit) before entering STOP mode. - Reading the MUA SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions. - The EP bit is cleared when the MU resets. 0b - The MUA side event is not pending (default). 1b - The MUA side event is pending.
3 —	Reserved
2-0 Fn	Fn For n = {0, 1, 2} MUA Side Flag n. (Read-only)

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- Fn bit is the MUA side flag that reflects the values written to the Fn bit in the MUB control register. - Every time that the Fn bit in the MUB CR register is written, the Fn bit in the MUB CR register write event updates the Fn bit in the MUA SR register after the event update latency, which is measured in terms of the number of clocks of the MUB and the MUA. 000b - Fn bit in the MUB CR register is written 0 (default). 001b - Fn bit in the MUB CR register is written 1.

42.6.1.7 Control Register (CR)

Offset

Register	Offset
CR	64h

Function

Use the Control Register (CR, 32-bit, read-write) to enable the MU interrupts on the MUA-side, and trigger events and interrupts on the MUB-side (general purpose interrupt, flag update).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GIEn				RIEn				TIEn				GIRn			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			RAIE	Reserv ed	Reserved		Reserv ed	Reserv ed	RDIE	MUR	Reserv ed	Reserv ed	Fn		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28	GIEn
GIEn	For n = {0, 1, 2, 3} MUA General Purpose Interrupt Enable n. (Read-Write) GIEn bit enables MUA General Interrupt n.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- If GIEn bit is set to "1" (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUA SR register is set to "1". - If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request is issued. - GIEn bit is cleared when the MU resets. 0000b - Disables MUA General Interrupt n. (default) 0001b - Enables MUA General Interrupt n.
27-24 RIEn	RIEn For n = {0, 1, 2, 3} MUA Receive Interrupt Enable n. (Read-Write) - RIEn bit enables MUA Receive Interrupt n. - If RIEn bit is set to "1" (enabled), then an MUA Receive Interrupt n request is issued when the RFn bit in the MUA SR register is set to "1". - If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUA Receive Interrupt n request is issued. - RIEn bit is cleared when the MU resets. 0000b - Disables MUA Receive Interrupt n. (default) 0001b - Enables MUA Receive Interrupt n.
23-20 TIEn	TIEn For n = {0, 1, 2, 3} MUA Transmit Interrupt Enable n. (Read-Write) - TIEn bit enables MUA Transmit Interrupt n. - If TIEn bit is set to "1" (enabled), then an MUA Transmit Interrupt n request is issued when the TEn bit in the MUA SR register is set to "1". - If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUA Transmit Interrupt n request is issued. - TIEn bit is cleared when the MU resets. 0000b - Disables MUA Transmit Interrupt n. (default) 0001b - Enables MUA Transmit Interrupt n.
19-16 GIRn	GIRn For n = {0, 1, 2, 3} MUA General Purpose Interrupt Request n. (Read-Write) - Writing "1" to the GIRn bit sets the GIPn bit in the MUB SR register on the Processor B-side. If the GIEn bit in the MUB CR register is set to "1" on the MUB side, a General Purpose Interrupt n request is triggered. - The GIRn bit is cleared if the GIPn bit (in the MUB SR register on the MUB side) is cleared by writing it (GIPn bit) as "1", thereby signalling the MUA that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as "0" on the MUA side.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). - GIRn bit is cleared when the MU resets. 0000b - MUA General Interrupt n is not requested to the MUB (default). 0001b - MUA General Interrupt n is requested to the MUB.
15-13 —	Reserved
12 RAIE	RAIE Processor B Reset Assertion Interrupt Enable. (Read-Write) - RAIE bit enables Processor A General Interrupt 3. - If RAIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the RAIP bit in the SR register is set to "1". - If RAIE is cleared, then the value of the RAIP bit is ignored and no General Interrupt 3 request is issued. - The RAIE bit is cleared when the MU resets. This bit is only available on the Processor A side. 0b - Disables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion. 1b - Enables Processor A General Purpose Interrupt 3 request due to Processor B reset assertion.
11 —	Reserved
10-9 —	Reserved
8 —	Reserved
7 —	Reserved
6 RDIE	RDIE Processor B Reset De-assertion Interrupt Enable. (Read-Write) - RDIE bit enables Processor A General Interrupt 3. - If RDIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the RDIP bit in the MUA SR register is set to "1".

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- If RDIE is cleared, then the value of the RDIP bit is ignored and no General Interrupt 3 request is issued. - The RDIE bit is cleared when the MU resets. This bit is only available on the Processor A side. 0b - Disables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion. 1b - Enables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion.
5 MUR	MUR MU Reset. - Setting MUR bit to "1" resets both the Processor A and the Processor B sides of the MU module, forcing all control and status registers to return to their default values and all internal states to be cleared. - The BOOT and RSTH fields is not affected by the MUR bit. - Before setting the MUR bit to "1", it is advisable to interrupt the Processor B, because setting the MUR bit may affect the ongoing Processor B program. - After setting the MUR bit, you should monitor the value of the RS bit in the MUA SR register to know when the reset sequence on the Processor B-side has ended. - MUR bit can only be written as "1". - MUR bit is always read as "0". - MUR bit is cleared during the MU reset sequence. This bit is only available on the Processor A side. 0b - N/A. Self clearing bit (default). 1b - Asserts the MU reset.
4 —	Reserved
3 —	Reserved
2-0 Fn	Fn For n = {0, 1, 2} MUA to MUB Flag n. (Read-Write) - Fn bit is a read-write flag that is reflected in Fn bit in the MUB SR register on the MUB side. - Fn bit is cleared when the MU resets. 000b - Clears the Fn bit in the SR register. 001b - Sets the Fn bit in the SR register.

42.6.2 MUB register descriptions

This section contains the detailed register descriptions for the MUB registers.

42.6.2.1 MU memory map

MU0.MUB base address: 4011_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VER)	32	R	0100_0001h
4h	Parameter Register (PAR)	32	R	0000_0000h
20h - 2Ch	Transmit Register (TR0 - TR3)	32	RW	0000_0000h
40h - 4Ch	Receive Register (RR0 - RR3)	32	R	0000_0000h
60h	Status Register (SR)	32	RW	00F0_0080h
64h	Control Register (CR)	32	RW	0000_0000h

42.6.2.2 Version ID Register (VER)

Offset

Register	Offset
VER	0h

Function

Use Version ID register to determine the version ID and feature set number of MUB.

Diagram

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

Fields

Field	Function
31-24	Major Version Number
MAJOR	This read only field returns the major version number for MUB.

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-16 MINOR	Minor Version Number This read only field returns the minor version number for MUB.
15-0 FEATURE	Feature Specification Number This read only field returns the feature set number for MUB. 0000_0000_0000_x1xb - Core Control and Status Registers are implemented in both MUA and MUB. 0000_0000_0000_xx1xb - RAIP/RAIE register bits are implemented. 0000_0000_0000_xxx0b - Standard features implemented

42.6.2.3 Parameter Register (PAR)

Offset

Register	Offset
PAR	4h

Function

Use Parameter register to determine the parameter settings of MUB.

Diagram

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

Fields

Field	Function
31-0 PARAMETER	This bitfield contains the parameter settings of MUB.

42.6.2.4 Transmit Register (TR0 - TR3)

Offset

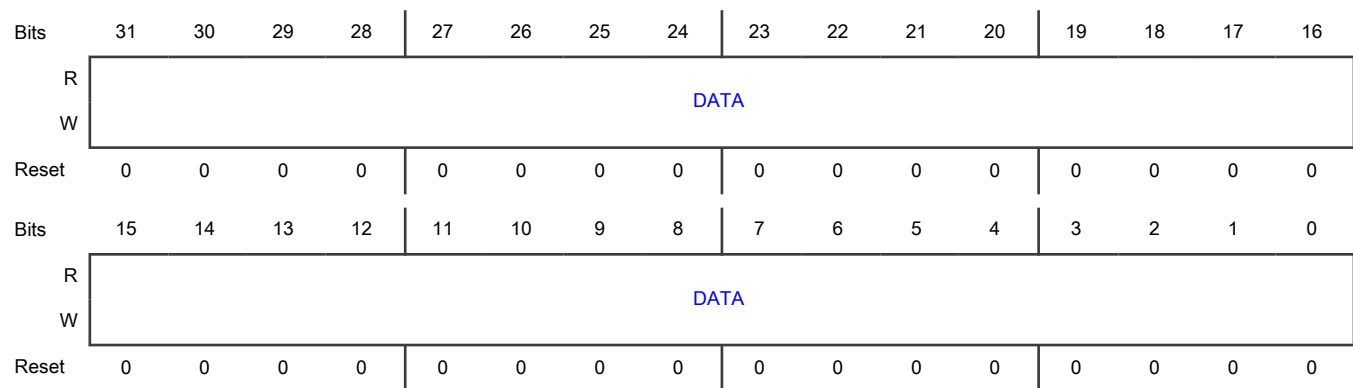
Register	Offset
TR0	20h
TR1	24h
TR2	28h
TR3	2Ch

Function

Use MUB Transmit Register (TRn, 32-bit, write-only) to transmit a message or data to MUA.

- You can only write to the TRn register when the TEn bit in SR register is set to "1".
- Reading the TRn register returns all zeros.

Diagram



Fields

Field	Function
31-0	DATA
DATA	MUB Transmit Register Data. (Write-only) • Data written to the TRn register is reflected in the MUA Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register. • A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUB Status Register (SR) on the transmitter side, and sets a "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUB side). • Any write to the TRn register updates all status information.

42.6.2.5 Receive Register (RR0 - RR3)

Offset

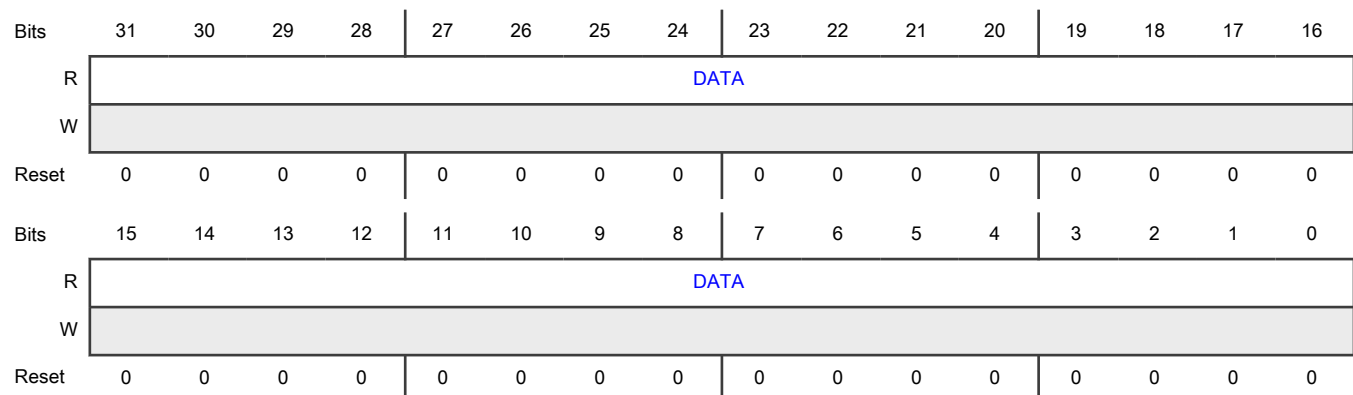
Register	Offset
RR0	40h
RR1	44h
RR2	48h
RR3	4Ch

Function

Use MUB Receive Register (RRn, 32-bit, read-only) to receive a message or data from the MUA.

- Data written to the MUA TRn register is immediately reflected in the MUB RRn register.
- You can only read the RRn register when the RFn bit in the SR register is set to "1".
- Writing to the RRn register generates an error response to the MUB.

Diagram



Fields

Field	Function
31-0	DATA
DATA	MUB Receive Register. (Read-only) • Reflects the data written to MUA Transmit Register 0 (TRn). • Reading the RRn register clears the "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUA Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUA side). • Any read of the RRn register updates all status information.

42.6.2.6 Status Register (SR)

Offset

Register	Offset
SR	60h

Function

Use the Processor B Status Register (SR, 32-bit, read-write) to show interrupt status from the Processor A, general purpose flags, the Processor A power mode, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor B-side programmer
- Other dual-purpose bits are set by the Processor B-side programmer, and cleared by the MU logic.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GIPn				RFn				TEn				Reserved			
W	W1C															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	Reserved			Reserv ed	Reserv ed	Reserv ed	FUP	RS	PM		EP	Reserv ed	Fn		
W					Reserv ed	Reserv ed	Reserv ed									
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Fields

Field	Function
31-28 GIPn	GIPn For n = {0, 1, 2, 3} MUB General Interrupt Request n Pending. (Read-Write) • GIPn bit signals the MUB that the GIPn bit in the CR register on the MUA side was set from "0" to "1". If the GIPn bit in the MUB CR register is set to "1", a General Interrupt n request is issued. • The GIPn bit is cleared by writing it back as "1". Writing "0", or writing "1" when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUB register, set the desired bit, and write it to the MUB SR register, thus clearing the GIPn bit. • GIPn bit is cleared when the MU is reset. 0000b - MUB general purpose interrupt n is not pending. (default) 0001b - MUB general purpose interrupt n is pending.
27-24 RFn	RFn

Table continues on the next page...

Table continued from the previous page...

Field	Function
	For n = {0, 1, 2, 3} MUB Receive Register n Full. (Read-only) - The RFn bit is set to "1" when the MUA TRn register is written on the MUA side. - After the RFn bit is set to "1", the RFn bit signals the MUB side that new data is ready to be read by the MUB in the MUB RRn register, and a Receive n interrupt is issued on the MUB side (if the RIEn bit in the MUB CR register has been set to "1"). - RFn bit is cleared when the MUB RRn register is read, and when the MU is reset. 0000b - MUB RRn register is not full (default). 0001b - MUB RRn register has received data from MUA TRn register and is ready to be read by the MUB.
23-20 TEn	TEn For n = {0, 1, 2, 3} MUB Transmit Register n Empty. (Read-only) - The TEn bit is set to "1" after the MUA RRn register is read on the MUA side. - After the TEn bit is set to "1", the TEn bit signals the MUB side that the MUB TRn register is ready to be written on the MUB side, and a Transmit n interrupt is issued on the MUB side (if the TEn bit in the MUB CR register is set to "1"). - TEn bit is cleared after the MUB TRn register is written on the MUB side. - TEn bit is set to "1" when the MU is reset. 0000b - MUB TRn register is not empty. 0001b - MUB TRn register is empty (default).
19-15 —	Reserved
14-12 —	Reserved
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 FUP	FUP MUB Flags Update Pending. (Read-only) FUP bit is set to "1" when the MUB side sends a Flags Update request to the MUA side.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUB CR register. No flag update changes are allowed while the FUP bit is set to "1". Any write to the Fn[2:0] bits of the MUB CR register, while the FUP bit is set to "1", does not generate a Flags Update event, and the Fn[2:0] bits stays unchanged. - FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU MUA side, and during MU reset. 0b - No flags updated, initiated by the MUB, in progress (default) 1b - MUB initiated flags update, processing
7 RS	RS MUA Reset State. (Read-only) - RS bit indicates if the MUA side of the MU is in a reset state or not. - If the RS bit is set to "1", then the MUA side of the MU is still in the reset state. - If the RS bit is cleared, then the MUA side of the MU are out of reset. - The RS bit is set to "1" during: a MUA system reset, or an MU reset (caused by setting the MUR bit at the CR register). - The RS bit is cleared when the reset sequence on the MUA side of the MU ends. After issuing any of the reset events mentioned previously, you should verify that the RS bit is cleared before starting any accesses. - When the MUA processor comes out of reset, the RS bit has value "1"(default). 0b - The MUA side of the MU is not in reset. 1b - The MUA side of the MU is in reset.
6-5 PM	PM Processor A-side Power Mode. (Read-only) PM[1:0] bits indicate the Processor A-side power mode. NOTE The Processor A Power Mode is platform-specific. 00b - The MUA processor is in Run Mode. 01b - The MUA processor is in WAIT Mode. 10b - The MUA processor is in STOP/VLPS Mode. 11b - The MUA processor is in LLS/VLLS Mode.
4 EP	EP MUB Side Event Pending. (Read-only) EP bit is set to "1" when the MUB side mechanism sends an event update request to the MUA side.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- EP bit is cleared when the event update acknowledge is received. An "event" is any hardware message that is reflected in the MUA SR register on the MUA side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically. - To ensure events have been posted to MUA before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to "1", you should wait and continue to poll it (EP bit) before entering STOP mode. - Reading the MUB SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions. - The EP bit is cleared when the MU resets. 0b - The MUB side event is not pending (default). 1b - The MUB side event is pending.
3 —	Reserved
2-0 Fn	Fn For n = {0, 1, 2} MUB Side Flag n. (Read-only) - Fn bit is the MUB side flag that reflects the values written to the Fn bit in the MUA control register. - Every time that the Fn bit in the MUA CR register is written, the Fn bit in the MUA CR register write event updates the Fn bit in the MUB SR register after the event update latency, which is measured in terms of the number of clocks of the MUA and the MUB. 000b - Fn bit in the MUA CR register is written 0 (default). 001b - Fn bit in the MUA CR register is written 1.

42.6.2.7 Control Register (CR)

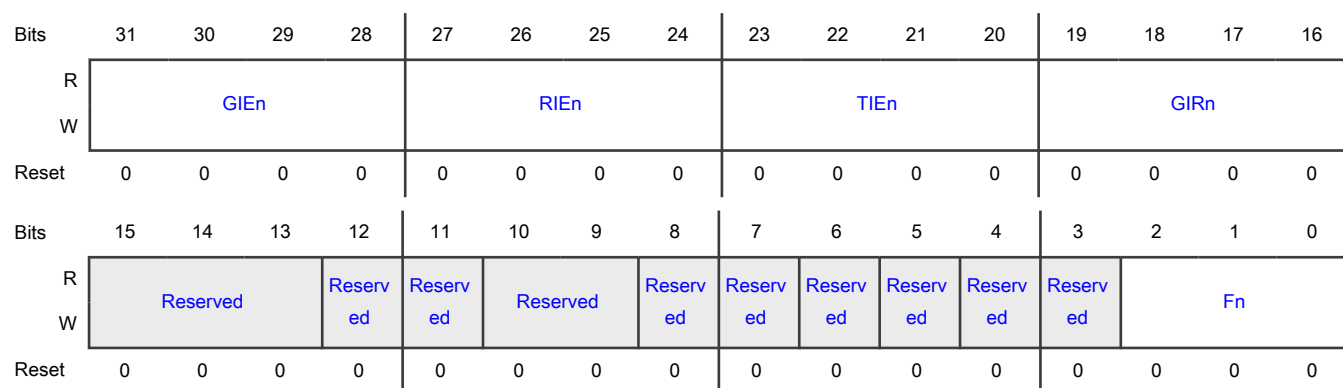
Offset

Register	Offset
CR	64h

Function

Use the Control Register (CR, 32-bit, read-write) to enable the MU interrupts on the MUB-side, and trigger events and interrupts on the MUA-side (general purpose interrupt, flag update).

Diagram



Fields

Field	Function
31-28 GIEn	GIEn For n = {0, 1, 2, 3} MUB General Purpose Interrupt Enable n. (Read-Write) - GIEn bit enables MUB General Interrupt n. - If GIEn bit is set to "1" (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUB SR register is set to "1". - If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request is issued. - GIEn bit is cleared when the MU resets. 0000b - Disables MUB General Interrupt n. (default) 0001b - Enables MUB General Interrupt n.
27-24 RIEn	RIEn For n = {0, 1, 2, 3} MUB Receive Interrupt Enable n. (Read-Write) - RIEn bit enables MUB Receive Interrupt n. - If RIEn bit is set to "1" (enabled), then an MUB Receive Interrupt n request is issued when the RFn bit in the MUB SR register is set to "1". - If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUB Receive Interrupt n request is issued. - RIEn bit is cleared when the MU resets. 0000b - Disables MUB Receive Interrupt n. (default) 0001b - Enables MUB Receive Interrupt n.
23-20 TIEn	TIEn For n = {0, 1, 2, 3} MUB Transmit Interrupt Enable n. (Read-Write) TIEn bit enables MUB Transmit Interrupt n.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- If TIEn bit is set to "1" (enabled), then an MUB Transmit Interrupt n request is issued when the TEn bit in the MUB SR register is set to "1". - If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUB Transmit Interrupt n request is issued. - TIEn bit is cleared when the MU resets. 0000b - Disables MUB Transmit Interrupt n. (default) 0001b - Enables MUB Transmit Interrupt n.
19-16 GIRn	GIRn For n = {0, 1, 2, 3} MUB General Purpose Interrupt Request n. (Read-Write) - Writing "1" to the GIRn bit sets the GIPn bit in the MUA SR register on the Processor A-side. If the GIEn bit in the MUA CR register is set to "1" on the MUA side, a General Purpose Interrupt n request is triggered. - The GIRn bit is cleared if the GIPn bit (in the MUA SR register on the MUA side) is cleared by writing it (GIPn bit) as "1", thereby signalling the MUB that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as "0" on the MUB side. - To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit). - GIRn bit is cleared when the MU resets. 0000b - MUB General Interrupt n is not requested to the MUA (default). 0001b - MUB General Interrupt n is requested to the MUA.
15-13 —	Reserved
12 —	Reserved
11 —	Reserved
10-9 —	Reserved
8 —	Reserved
7 —	Reserved
6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 —	Reserved
4 —	Reserved
3 —	Reserved
2-0 Fn	Fn For n = {0, 1, 2} MUB to MUA Flag n. (Read-Write) • Fn bit is a read-write flag that is reflected in Fn bit in the MUA SR register on the MUA side. • Fn bit is cleared when the MU resets. 000b - Clears the Fn bit in the SR register. 001b - Sets the Fn bit in the SR register.

Chapter 43

Semaphores2 (SEMA42)

43.1 Chip-specific SEMA42 information

Table 466. Reference links to related information

Topic	Related module	Reference
Full description	SEMA42	SEMA42
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Initial configuration of the Semaphore block can be accomplished as follows:

- Enable the clock to the Semaphore block in the CLKCTL1 PSCCTL1 register. This enables the register interface and the peripheral function clock.
- Clear the Semaphore block peripheral reset in the RSTCTL1 PRSTCTL1 register by writing to the PRSTCTL1_CLR register.

43.1.1 Module instances

This device has one instance of the SEMA4 module, SEMA42.

43.2 Introduction

SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single - write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

43.2.1 Multi-core programming: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism for system software to control access to shared data structures, shared hardware resources, and so on. The software uses the gating mechanisms to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and domains.

Consider the following description of a typical use-case: [dmX](#) enters a section of code. where shared data values are to be updated. The domain must first acquire a semaphore. Think of this as the locking (or closing) of a software gate. After the gate locks, a properly-architected software system does not allow other processes (or domains) to execute the same code segment or modify the shared data structure protected by the gate. In other words, the system locks out other processes/domains. Many software implementations include a spin-wait loop within the lock function until the gate locks. After domain X obtains the lock, domain X continues execution and updates the data values protected by the particular lock. After domain X completes the updates, it unlocks (or opens) the software gate, allowing other processes/domains access to the updated data values.

A correctly-implemented system solution must follow these important rules:

- A gate variable must protect all writes to shared data values or shared hardware resources.
- After a domain locks a gate, the system must block other processes/domains from accessing the shared data or resources. Software conventions enforce this.
- The domain that locks a particular gate is the only domain that can open (unlock) that gate.

Information in the hardware gate identifying the locking domain can be extremely useful for system-level debugging.

43.2.2 Block diagram

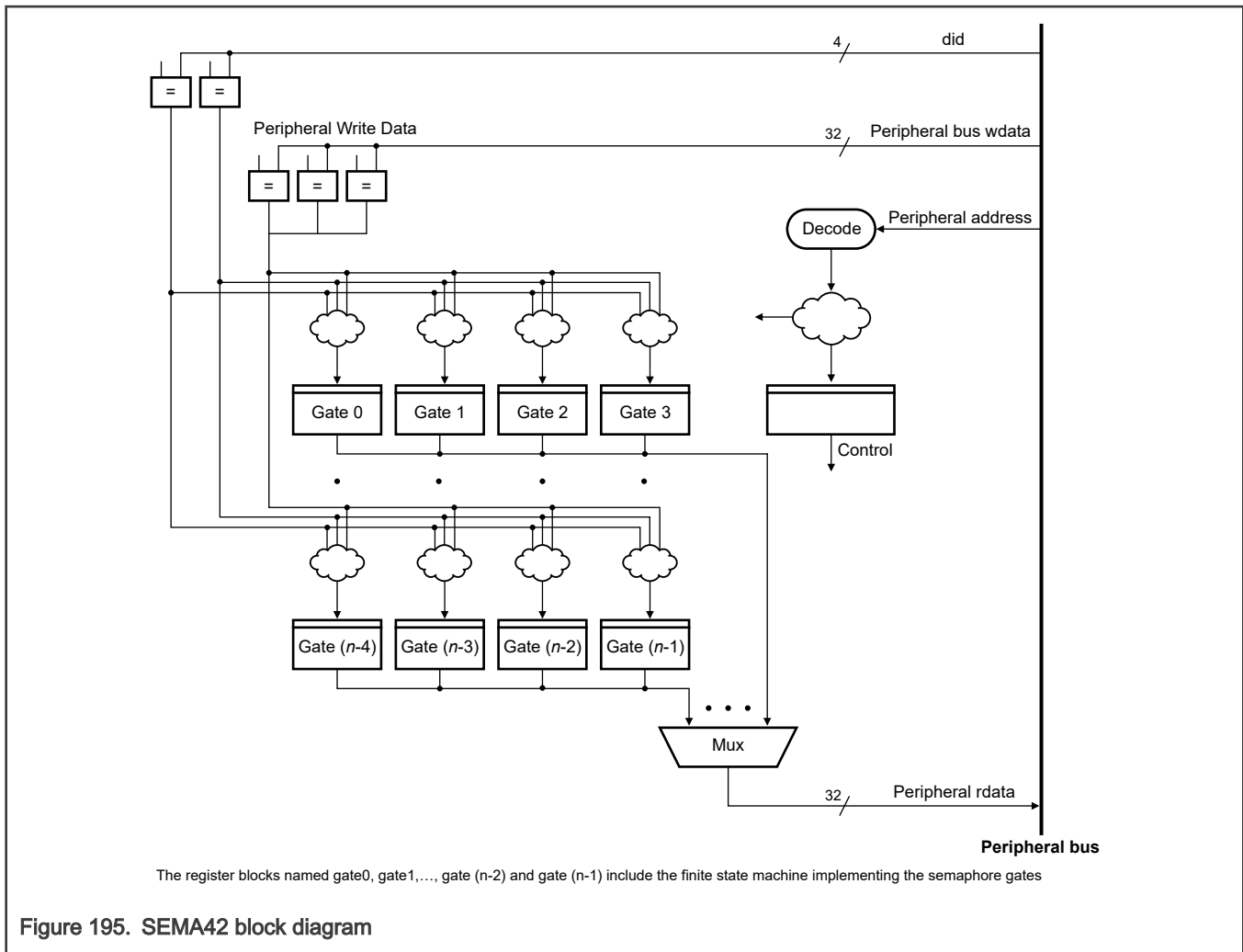


Figure 195. SEMA42 block diagram

43.2.3 Features

SEMA42 implements hardware-enforced semaphores as an [IPS](#)-mapped slave peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a multi-domain configuration that supports up to 15 domains.
- Each hardware gate appears as a 16-state, 4-bit state machine.
- Support for secure reset mechanisms to clear the contents of individual gates, as well as a clear-all capability.
- Memory-mapped slave peripheral that offers programming-model accesses.

43.3 Functional description

The intent of the hardware gate implementation is to specify a protocol where the locking domain must unlock the gate. However, some systems may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset. To support this special gate reset requirement, SEMA42 implements a secure reset mechanism that allows you to initialize a hardware gate (or all the gates) by following a specific dual-write access pattern. The secure-gate reset:

- Uses a technique similar to that required for the servicing of a software watchdog timer

- Requires two consecutive writes with pre-defined data patterns from the same domain

You must do this to force the clearing of the specified gate(s). The required access pattern as follows:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte (`RSTGT_W[RSTGDP]`) must be E2h. The value of least significant byte is irrelevant for this reference and can be anything.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte (`RSTGT_W[RSTGDP]`) is the logical complement of the first data pattern (1Dh) and the lower byte (`RSTGT_W[RSTGTN]`) specifies the gate(s) to be reset. This gate field can specify a single gate or all gates to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, SEMA42 aborts the special gate reset sequence and does not assert an error signal.
3. Reads of the RSTGT location return information on the 2-bit reset state machine (`RSTGT_R[RSTGSM]`) that implements these functions:
 - The domain performing the reset (`RSTGT_R[RSTGMS]`)
 - The last-cleared gate number(s) (`RSTGT_R[RSTGTN]`)

Reads of the RSTGT register do not affect the secure-reset finite state machine in any manner.

43.3.1 16 Hardware-enforced gates

Gates appear as a 16-entry byte-size array with read and write accesses.

Domains lock gates by writing "domainID_number+1" to the appropriate gate and must read back the gate value to verify that the lock operation succeeded.

After the gate locks, the locking domain unlocks the gate by writing zeroes.

- 16-state implementation
 - If gate = 0h, then state = unlocked
 - if gate = 1h, then state = locked by domain (master) 0
 - if gate = 2h, then state = locked by domain (master) 1
 - ...
 - if gate = Fh, then state = locked by domain (master) 14

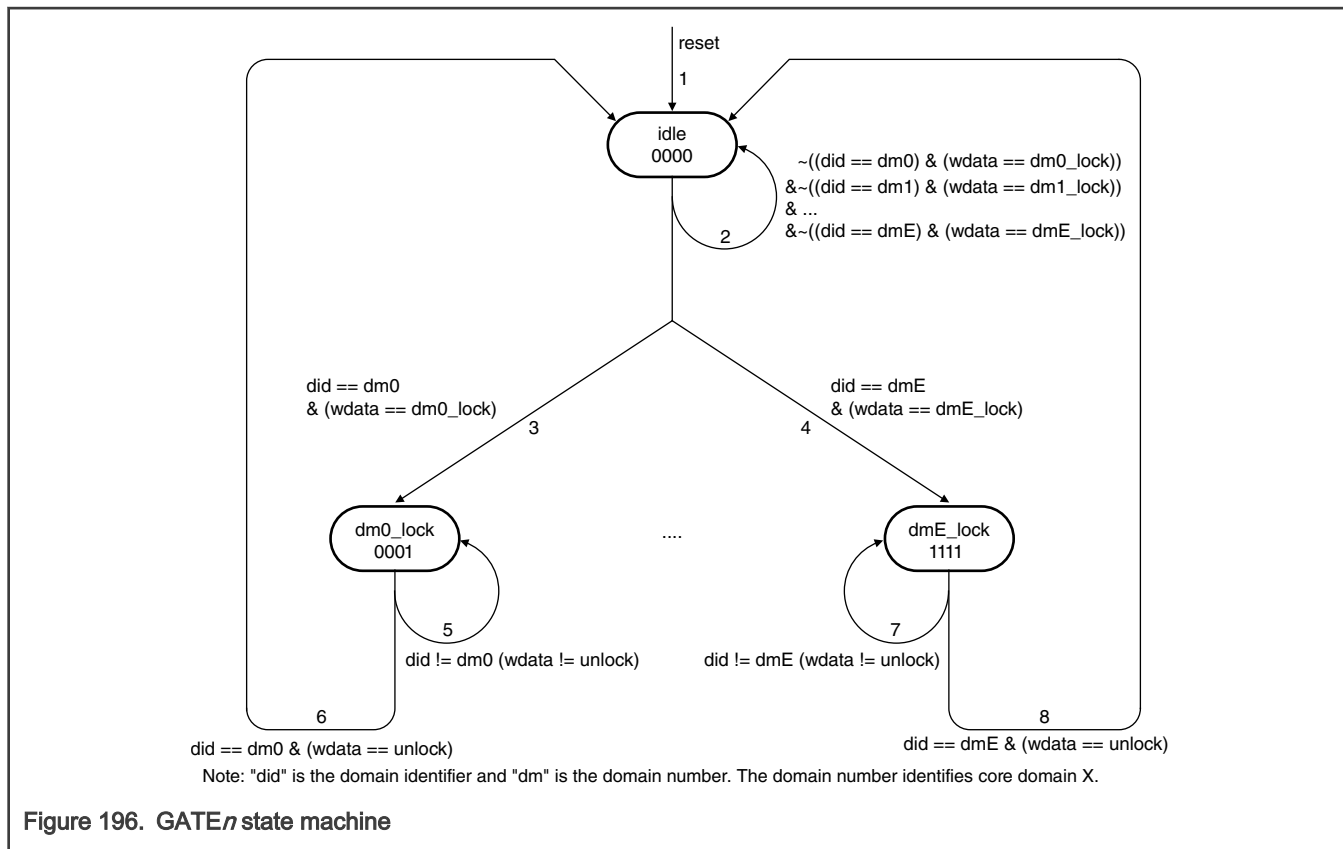
SEMA42 uses the logical domain number and the specified data patterns as reference attributes to validate all write operation.

After a gate locks, the locking domain must unlock the gate by writing zeroes.

43.3.2 State machine of the GATE_n registers

This section describes more about the SEMA42 functional operation and specific details of the state machines of the GATE_n registers.

As described previously, each of the GATE_n registers implements a 4-bit, 16-state machine. The following figure shows a simplified diagram of the state transitions for each gate.



In the figure above, "dmE" represents domain 14 (E in hexadecimal). The platform passes the domain number to SEMA42. The following table defines the GATE n state transitions.

Table 467. GATE n state transitions

Current state	Next state	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding domain occurs, the gate remains in the idle state.
idle	dm0_lock	3	When domain 0h initiates a write of the dm0_lock data value, the gate transitions into the dm0_lock state.
idle	dmE_lock	4	When domain Eh initiates a write of the dmE_lock value, the gate transitions into the dmE_lock state.
dm0_lock	dm0_lock	5	When in this state, the gate remains here if any attempted write is not from domain 0h with the unlock data value.
dm0_lock	idle	6	The gate returns to the idle (unlocked) state after a write from domain 0h with the unlock data value occurs.

Table continues on the next page...

Table 467. GATE_n state transitions (continued)

Current state	Next state	Transition	Description
dmE_lock	dmE_lock	7	When in this state, the gate remains here if any attempted write is not from domain Eh with the unlock data value.
dmE_lock	idle	8	The gate returns to the idle (unlocked) state after a write from domain Eh with the unlock data value occurs.

SEMA42 uses these gate data values:

- The lock data value is (domain number) + 1.
- The unlock data value is 00h.

43.3.3 Clocking

This module has no clocking considerations.

43.3.4 Interrupts

This module has no interrupts.

43.4 Memory map/register definition

You can access these registers only in Supervisor mode. User accesses terminate with an error.

43.4.1 SEMA42 register descriptions

43.4.1.1 SEMA42 memory map

SEMA42 base address: 4011_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h - Fh	Gate (GATE0 - GATE15) ¹	8	RW	00h
42h	Reset Gate Read (RSTGT_R)	16	R	0000h
42h	Reset Gate Write (RSTGT_W)	16	W	See section

1. In this array, the index and offset values of the registers do not increment in direct alignment. For details, see the register description.

43.4.1.2 Gate (GATE0 - GATE15)

Offset

For n = 0 to 15:

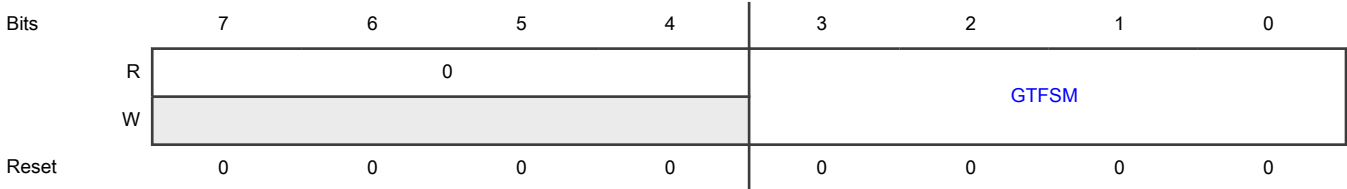
Register	Offset
GATE _n	0h + (n + 3 - 2 × (n mod 4))

Function

Implements each semaphore gate in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical domain-identifier number in conjunction with the data patterns to validate all attempted write operations. Only domain masters can modify the gate registers. After a gate locks, only the locking domain must open (unlock) the gate.

You can read multiple gate values in a single access. However, you can update only a single gate at a time, via a write operation. If you attempt to write a byte-wide value that is neither the unlock value (00h) nor the appropriate lock value (domainID_number + 1), SEMA42 considers this as "no operation" and does not change any gate state. Attempts to write multiple gates in a single-aligned access with a size larger than 8 bits (byte) generate an error termination and do not allow any gate state changes.

Diagram



Fields

Field	Function
7-4 —	Reserved
3-0 GTFSM	Gate Finite State Machine The state of the gate reflects the last domain that locked it. This can be useful during system debug. The hardware gate has a 16-state implementation, defined as: 0000b - The gate is unlocked (free). 0001b - Domain 0 locked the gate. 0010b - Domain 1 locked the gate. 0011b - Domain 2 locked the gate. 0100b - Domain 3 locked the gate. 0101b - Domain 4 locked the gate. 0110b - Domain 5 locked the gate. 0111b - Domain 6 locked the gate. 1000b - Domain 7 locked the gate. 1001b - Domain 8 locked the gate. 1010b - Domain 9 locked the gate. 1011b - Domain 10 locked the gate. 1100b - Domain 11 locked the gate. 1101b - Domain 12 locked the gate. 1110b - Domain 13 locked the gate. 1111b - Domain 14 locked the gate.

43.4.1.3 Reset Gate Read (RSTGT_R)

Offset

Register	Offset
RSTGT_R	42h

Function

Describes how register field appear when you read the register. [Reset Gate Write \(RSTGT_W\)](#) also describe the same register showing the fields when you write it.

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		RSTGSM				RSTGMS						RSTGTN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
15-14 —	Reserved
13-12 RSTGSM	Reset Gate Finite State Machine Reads of the RSTGT register return the encoded state machine value. RSTGSM = 10b is valid for only a single machine cycle, so a read can never return this value. SEMA42 maintains the reset state machine in a 2-bit, 3-state implementation, defined as follows: 00b - Idle, waiting for the first data pattern write. 01b - Waiting for the second data pattern write 10b - The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. 11b - This state encoding is never used and therefore reserved.
11-8 RSTGMS	Reset Gate Domain Records the logical number of the domain performing the gate reset function. The logical number is the domain number. This domain number is determined by the XRDC's Master Domain Assignment Controller (XRDC_MDAC). To succeed, this function requires that the same domain initiate the two consecutive writes to this register. SEMA42 updates the field each time a write to this register occurs.
7-0 RSTGTN	Reset Gate Number Specifies the specific hardware gate to be reset. The second write updates this field. If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates.

43.4.1.4 Reset Gate Write (RSTGT_W)

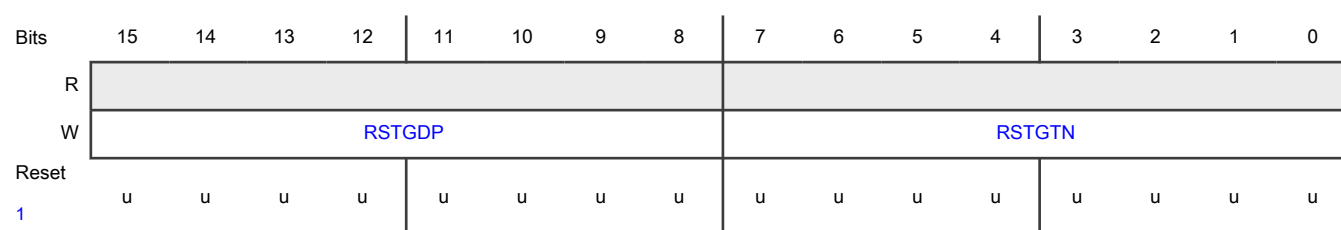
Offset

Register	Offset
RSTGT_W	42h

Function

Describes how register field appear when you write the register. See [Reset Gate Read \(RSTGT_R\)](#) for the description of the register's overall operation and how the register fields appear when you read the register.

Diagram



- Reset value is not applicable for writes.

Fields

Field	Function
15-8 RSTGDP	Reset Gate Data Pattern You access this field with the specified data patterns on the two consecutive writes to enable the gate-reset mechanism. For the first write, RSTGDP must be E2h. For the second write, RSTGDP must be 1Dh.
7-0 RSTGTN	Reset Gate Number Specifies the specific hardware gate to be reset. The second write updates this field. If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates.

43.5 External signals

This module has no external signals.

43.6 Initialization

This module does not require initialization.

43.7 Glossary

IPS	Protocol used for peripheral accesses to the programming model
dmX	Domain X

Chapter 44

Cache Controller

44.1 Chip-specific CACHE64 information

Table 468. Reference links to related information

Topic	Related module	Reference
Full description	CACHE64 Controller	CACHE64 Controller
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Cache Controller register bitfield CSAR[28] is always 1'b0.

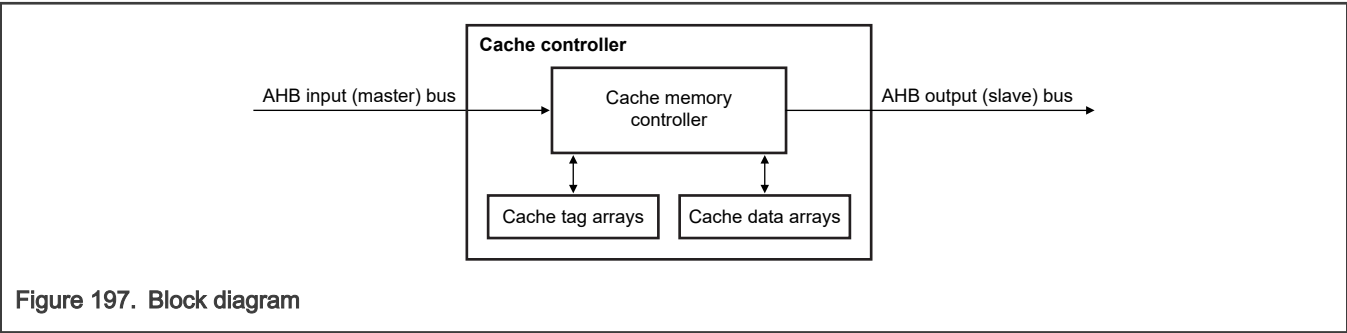
44.1.1 Module instances

This device has two instances of the CACHE64 module, CACHE_Control_0 and CACHE_Control_1.

44.2 Overview

A cache is a block of high-speed memory locations that contains address information (commonly known as a tag) and its associated data. It decreases the average time of a memory access. CACHE_CTRL includes cache memory controller (CACHE64), which is a general purpose AMBA AHB bus protocol cache.

44.2.1 Block diagram



44.2.2 Features

CACHE_CTRL is designed to work with AMBA AHB bus systems. It includes:

- An input (master) AHB bus that you can access using Cache mode attributes for cacheable copyback, cacheable write-through, or noncacheable mode operation and write bufferable mode operation on an access-by-access basis.
- An output (slave) 64-bit data path AHB bus for downstream cache read miss, cache write-through write, and noncacheable accesses.
- A peripheral interface for accessing the cache's program model.

44.3 Functional description

A cache operates on two principles of locality:

- Spatial locality—Access to one location possibly follows an access from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality—Access to an area of memory possibly repeats within a short period of time (for example, execution of a code loop).

To minimize the quantity of control information stored, the spatial locality property groups several locations together, under the same tag. This logical block is known as a cache line.

You achieve temporal locality by retaining recently accessed lines in the cache.

When data loads into a cache, access times for subsequent loads and stores are reduced, resulting in overall performance benefits. An access to information already in a cache is known as a cache hit, and other accesses are known as cache misses.

Normally, caches are self-managing, with the updates occurring automatically. Whenever the bus master wants to access a cacheable location, the cache is checked. If the access is a cache hit, the access occurs immediately. Otherwise, a location is allocated, and the cache line is loaded from memory. Different cache topologies and access policies are possible. However, they must comply with the memory coherency model of the underlying architecture.

Caches introduce a number of potential problems, mainly because of:

- Memory accesses occurring at times other than when the programmer would normally expect them.
- The existence of multiple physical locations where a data item can be held.

44.3.1 Modes of operation

Table 469. Modes of operation

Mode	Description
Write-Through	Access to address spaces with this cache mode are cacheable: • If all cacheable spaces are read-only spaces, then cache contains read-only data and all writes to the cache cause a fault. See the chip-specific cacheable space information. • A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address that contains the desired address. A read miss data loads into the cache and marks as valid and not modified. • A read hit to a valid cache location returns data from the cache with no output bus accesses. • A write miss bypasses the cache and writes to the output bus (no allocation on write-miss policy for Write-Through mode spaces). • A write hit updates the cache hit line with the write data and writes to the output bus. • The caches are bus-master-local and do not support hardware cache coherency. If the bus master accesses write-through regions and another bus master that can access the same target memory, without going through the cache (such as DMA), needs to update these regions, you must first perform explicit cache clears to

Table continues on the next page...

Table 469. Modes of operation (continued)

Mode	Description
	ensure all modified cache lines update their associated memories, before they are modified by external masters and subsequent cache bus master accesses get the updated memory.
Write-Back	Access to address spaces with this cache mode are cacheable: • A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address that contains the desired address. This miss data loads into the cache and marks it as valid and not modified. • A read hit to a valid cache location returns data from the cache with no output bus access. • A write miss performs a read-to-write (executes write-miss policy for Write-Back mode spaces). You perform a line read on the output bus of a 32-byte-aligned memory address containing the desired write address. This miss data is loaded into the cache and marked as valid and modified. Then the write data updates the appropriate cache data locations. • A write hit updates the cache hit line with the write data and marks the line as modified. • Caches do not support hardware cache coherency. If the cache's bus master has written to write-back regions, and another bus master that can access the same target memory without going through the cache needs to see these updates, you must perform explicit cache pushes to any needed cache memory range to ensure that all modified cache lines update their associated memories before being this master reads them. Likewise, if the cache's bus master has accessed write-back regions, and another bus master that can access the same target memory without going through the cache needs to update these regions, you must should first perform explicit cache clears to any needed cache memory range to ensure that all modified cache lines update their associated memories before this master modifies them, and subsequent cache master bus accesses get the updated memory.
Noncacheable	Access to address spaces with this cache mode are not cacheable. These accesses bypass the cache and access the output bus.

44.3.2 Cache configuration

CACHE_CTRL receives master bus requests on the master (M) bus.

You can access the cache programming model via the cache controller's AMBA AHB bus.

CACHE_CTRL then processes the cacheable accesses as needed, when bypassing the noncacheable, cache write-through, cache miss, and cache maintenance accesses to its slave bus.

The cache on this chip has a 2-way set associative cache structure with a total size of 32 KB. The cache has a 32-bit address, 64-bit data paths, and a 32-byte line size. The cache tags and data storage use single-port, synchronous RAMs.

For this 32 KB cache, each cache TAG function uses two 512 x 20-bit RAM arrays, and the cache DATA function uses two 2048 x 64-bit RAM arrays. The cache TAG entries store 18 bits of the upper address as well as a modified and valid bit per cache line. The cache DATA entries store eight bytes of code or data.

All normal cache accesses use physical addresses. This leads to the use of the following cache address:

CACHE - 32 KB size = (512 sets) x (32-byte lines) x (2-way set associative)

Table 470. Tag cache address use

Tag cache address use	Cache
Tag hit address range	Address[31:14]
Tag set select address range	Address[13:5] used to select 1 of 512 sets
Not used	Address[4:0]

Table 471. Data cache address use

Data cache address use	System cache
Not used	Address[31:14]
Data set select address range	Address[13:5] used to select one of 512 sets
64-bit word select	Address[4:3] used to select one of four 64-bit words within a set
Byte select	Address[2:0] used to select the byte within the 64-bit word

44.3.3 Cache control

The cache is disabled at reset. Cache tag and data arrays are not cleared at reset. Therefore, to enable and configure the cache, you must clear the cache commands and initialize the required tag array.

44.3.3.1 Cache set commands

The cache set commands may operate on one of the following:

- All of way 0
- All of way 1
- All of both ways (complete cache)

Cache set commands are specified using the upper bits in [Cache Control \(CCR\)](#). Cache set commands perform their operation on the cache independent of [CCR\[ENCACHE\]](#).

You initiate a cache set command by writing 1 to [CCR\[GO\]](#). This field also acts as a busy bit for set commands. The field remains 1 when the command is active, and the module writes 0 to it after the set command completes.

Supported cache set commands are provided in [Table 472](#). Following are the set commands:

- Invalidate—Unconditionally clears valid and modify bits of a cache entry.
- Push—Pushes a cache entry if it is valid and modified, then clears the modify bit. If the entry is not valid or not modified, leave it as is.
- Clear—Pushes a cache entry if it is valid and modified, then clears the valid and modify bits. If the entry is not valid or not modified, clear the valid bit.

Table 472. Cache set commands

Cache Control (CCR)[27:24]				Command
PUSHW1	INVW1	PUSHW0	INVW0	
0	0	0	0	No operation (NOP)
0	0	0	1	Invalidate all of way 0
0	0	1	0	Push all of way 0
0	0	1	1	Clear all of way 0
0	1	0	0	Invalidate all of way 1
0	1	0	1	Invalidate all of way 1 and way 0 (invalidate complete cache)
0	1	1	0	Invalidate all of way 1; push all of way 0
0	1	1	1	Invalidate all of way 1; clear all of way 0
1	0	0	0	Push all of way 1
1	0	0	1	Push all of way 1; invalidate all of way 0
1	0	1	0	Push all of way 1 and way 0 (push complete cache)
1	0	1	1	Push all of way 1; clear all of way 0
1	1	0	0	Clear all of way 1
1	1	0	1	Clear all of way 1; invalidate all of way 0
1	1	1	0	Clear all of way 1; push all of way 0
1	1	1	1	Clear all of way 1 and way 0 (clear complete cache)

After a reset, complete an invalidate cache command before using the cache. It is possible to combine the cache invalidate command with the cache enable. That is, writing 8500_0003h to [Cache Control \(CCR\)](#) invalidates the cache and enables the cache and write buffer.

44.3.3.2 Cache line commands

Cache line commands operate on a single line in the cache at a time. You can perform these commands using a physical or cache address that consists of a set address and a way select. The line command acts on the specified cache line.

Cache line commands with physical addresses first search both ways of the cache set specified by the following physical address bits. If they hit, the commands perform their action on the hit way: For cache-[13:5].

You specify cache line commands using the upper bits in [Cache Line Control \(CLCR\)](#). Cache line commands perform their operation on the cache independent of [CCR\[ENCACHE\]](#). Using a cache address, the command can be completely specified using [Cache Line Control \(CLCR\)](#). Using a physical address, the command must also use [Cache Search Address \(CSAR\)](#) to specify the physical address.

You can initiate a line cache command by writing 1 to [CLCR\[LGO\]](#) or [CSAR\[LGO\]](#). This field also acts as a busy bit for line commands. The field remains 1 when the command is active and [CACHE_CTRL](#) writes 0 to it after the set command completes.

Table 473. Cache line commands

Cache Line Control (CLCR)[27:24]			Command
CLCR[LACC]	CLCR[LADSEL]	CLCR[LCMD]	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	NOP
1	0	10	NOP
1	0	11	NOP
1	1	xx	NOP

44.3.3.2.1 Executing a series of line commands using cache addresses

You can perform a series of line commands with incremental cache addresses by writing to [Cache Line Control \(CLCR\)](#):

1. Place the command in [Cache Line Control \(CLCR\)](#)[27:24].
2. Write 1 to [CLCR\[WSEL\]](#) and [CLCR\[TDSEL\]](#) as needed.
3. Place the cache address in [CLCR\[CACHEADDR\]](#).
4. Write 1 to [CLCR\[LGO\]](#).
5. Wait for the command to complete ([CLCR\[LGO\]](#) becomes 0).
6. Increment the cache address (at bit 3 to step through data or at bit 5 to step through lines).
7. Write 1 to [CLCR\[LGO\]](#).
8. Wait for the command to complete ([CLCR\[LGO\]](#) becomes 0).
9. Repeat steps 5 - 8 as needed.

44.3.3.2.2 Executing a series of line commands using physical addresses

You can perform a series of line commands with incremental physical addresses using the following steps:

1. Write to [Cache Line Control \(CLCR\)](#):
 - a. Place the command in [Cache Line Control \(CLCR\)](#) [27:24].
 - b. Write 1 to [CLCR\[TDSEL\]](#).
2. Place the physical address in [CSAR\[PHYADDR\]](#) and write 1 to [CSAR\[LGO\]](#).
3. Wait for the command to complete ([CLCR\[LGO\]](#) becomes 0).
4. Increment the physical address (at bit 3 to step through data or at bit 5 to step through lines).
5. Write 1 to [CLCR\[LGO\]](#).

6. Wait for the command to complete ([CLCR\[LGO\]](#) becomes 0).
7. Repeat steps 3 - 6 as needed.

[Cache Line Control \(CLCR\)](#) and [Cache Search Address \(CSAR\)](#) both contain a field called LGO, and this field has the same functionality. Therefore, you can complete the above steps in a single write to [Cache Search Address \(CSAR\)](#).

44.3.3.2.3 Line command results

At completion of a line command, [Cache Line Control \(CLCR\)](#) contains information on the initial state of the line targeted by the command. For line commands with cache addresses, read this information before performing a line command action from the targeted cache line. For line commands with physical addresses, read this information on a hit before the line command action is performed from the hit cache line. In the cache, for line commands with a physical address miss, [CLCR\[LCIVB\]](#) becomes 0, and no line command action is performed. In general, if [CLCR\[LCIVB\]](#) becomes 0, the targeted line is invalid at the start of the line command and no line operation is performed.

Table 474. Line command results

Cache Line Control (CLCR)[22:20]			For cache address commands	For physical address commands
CLCR[LCWAY]	CLCR[LCIMB]	CLCR[LCIVB]		
0	0	0	Way 0 line is invalid	No hit
0	0	1	Way 0 valid and not modified	Way 0 valid and not modified
0	1	0	Way 0 line is invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line is invalid	No hit
1	0	1	Way 1 valid and not modified	Way 1 valid and not modified
1	1	0	Way 1 line is invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified

On completion of a line command other than a write, [Cache Read/Write Value \(CCVR\)](#) contains information on the initial state of the line tag or data targeted by the command. For line commands, [CLCR\[TDSEL\]](#) selects between tag and data. If the line command used a physical address and missed, the data is don't care. For write commands, [Cache Read/Write Value \(CCVR\)](#) holds the write data.

The cache does not have line lock capability.

44.3.4 Clocks

Caches use the AHB master bus clock.

44.3.5 Resets

Caches are disabled at reset. However, cache tag and data arrays are not cleared at reset. See [Initialization](#) and [Cache control](#) for more information regarding cache initialization and configuration.

44.4 External signals

There are no external signals for CACHE_CTRL.

44.5 Initialization

Perform this procedure to initialize cache:

1. Write 1 to [CCR\[INVW1\]](#) and [CCR\[INVW0\]](#).

2. Write 1 to [CCR\[GO\]](#).
3. Write 1 to [CCR\[ENWRBUF\]](#) and [CCR\[ENCACHE\]](#).

NOTE

Writing 1 to [CCR\[ENWRBUF\]](#) and [CCR\[ENCACHE\]](#) can be done simultaneously. That is, writing 8500_0003h to [Cache Control \(CCR\)](#) invalidates the cache and enables the cache and write buffer.

44.6 Memory map and registers

The cache programmer's model provides a variety of registers for configuring and controlling the cache, as well as indirect access paths to all cache tag and data storage.

NOTE

The cache registers are accessible in Supervisor mode only.

44.6.1 CACHE64 register descriptions

44.6.1.1 CACHE64 memory map

CACHE64_CTRL0 base address: 4003_3000h

CACHE64_CTRL1 base address: 4003_4000h

Offset	Register	Width (In bits)	Access	Reset value
800h	Cache Control (CCR)	32	RW	0000_0000h
804h	Cache Line Control (CLCR)	32	RW	0000_0000h
808h	Cache Search Address (CSAR)	32	RW	0000_0000h
80Ch	Cache Read/Write Value (CCVR)	32	RW	0000_0000h

44.6.1.2 Cache Control (CCR)

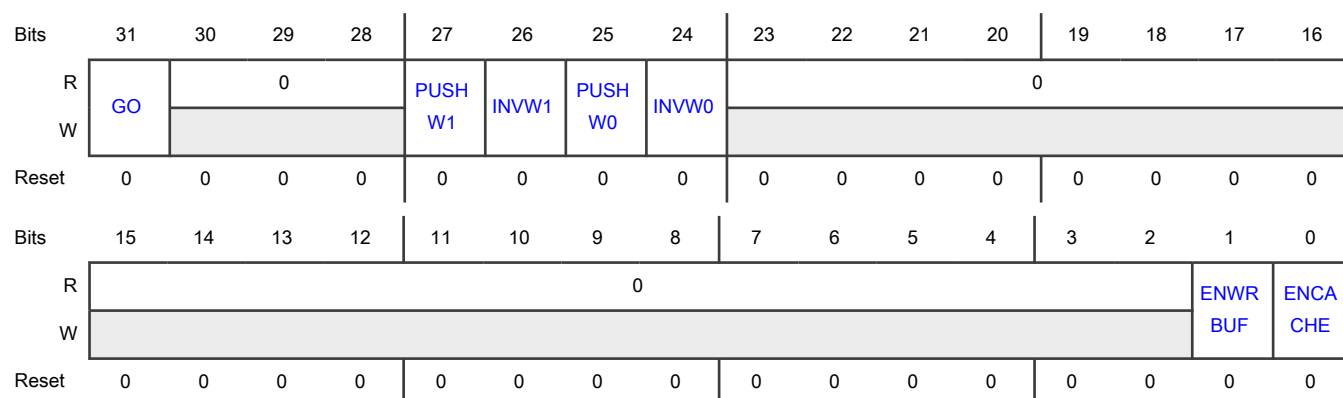
Offset

Register	Offset
CCR	800h

Function

Enables the cache and write buffer. This register also contains the fields used for commands specified in [Cache set commands](#).

Diagram



Fields

Field	Function
31 GO	Initiate Cache Command Initiates the cache command that CCR[INVW0], CCR[PUSHW0], CCR[INVW1], and CCR[PUSHW1] indicate, if you write 1 to this field. Reading this field indicates whether a cache command is active. NOTE This field remains 1 until the cache command completes. Writing 0 has no effect. 0b - Write: no effect; Read: no cache command active 1b - Write: initiates cache command; Read: cache command active
30-28 —	Reserved
27 PUSHW1	Push Way 1 Specifies the cache command to execute the next time CCR[GO] becomes 1. 0b - No operation 1b - Push all modified lines in way 1
26 INVW1	Invalidate Way 1 Specifies the cache command to execute the next time CCR[GO] becomes 1. NOTE If CCR[PUSHW1] and CCR[INVW1] are 1, then after you write 1 to CCR[GO], this field pushes all modified lines in way 1 and invalidates all lines in way 1 (clear way 1). 0b - No operation 1b - Invalidates all lines in way 1
25 PUSHW0	Push Way 0 Specifies the cache command to execute the next time CCR[GO] becomes 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No operation 1b - Push all modified lines in way 0
24 INVW0	Invalidate Way 0 Specifies the cache command to execute the next time CCR[GO] becomes 1. NOTE If CCR[PUSHW0] and CCR[INVW0] are 1, then after you write 1 to CCR[GO], this field pushes all modified lines in way 0 and invalidates all lines in way 0 (clear way 0). 0b - No operation 1b - Invalidates all lines in way 0
23-2 —	Reserved
1 ENWRBUF	Enable Write Buffer Enables the write buffer. 0b - Disables 1b - Enables
0 ENCACHE	Cache Enable Enables the cache. 0b - Disables 1b - Enables

44.6.1.3 Cache Line Control (CLCR)

Offset

Register	Offset
CLCR	804h

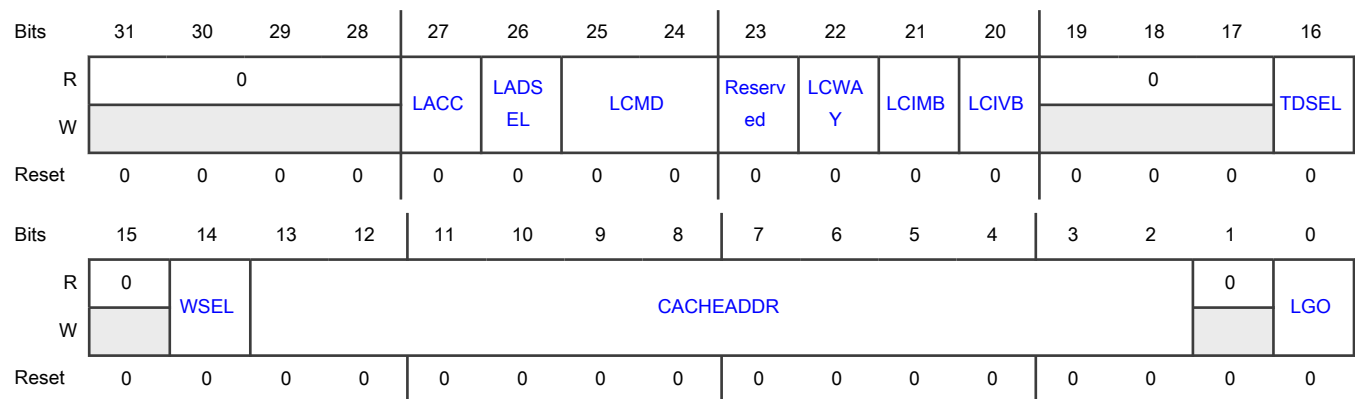
Function

Defines the commands specified in [Cache line commands](#) that you must perform using a specific cache line address or physical address.

If you specify the physical address, you can search cache in both the ways, and the command only performs on the way that hits.

You can write to [CLCR\[LCWAY\]](#), [CLCR\[LCIMB\]](#), and [CLCR\[LCIVB\]](#); however, the values that you write to these fields have no functional purpose.

Diagram



Fields

Field	Function
31-28 —	Reserved
27 LACC	Line Access Type Specifies the operation that you must perform (read or write) after a hit, when performing a search command. NOTE Only used for search commands (CLCR[LCMD] = 00). 0b - Read 1b - Write
26 LADSEL	Line Address Select Specifies the address (cache or physical). When using the cache address (specified in CLCR[WSEL] and CLCR[CACHEADDR]), the way must also be specified in CLCR[WSEL]. When using the physical address (specified in Cache Search Address (CSAR)), both ways are searched and the command is performed only after a hit. 0b - Cache 1b - Physical
25-24 LCMD	Line Command Performs one of the four commands after a cache hit. CLCR[LACC] determines whether the operation is a read or write. 00b - Search and read or write 01b - Invalidate 10b - Push 11b - Clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 —	Reserved
22 LCWAY	Line Command Way Indicates the way that the line command uses. Applies only if CLCR[LCIVB] = 1. 0b - Way 0 1b - Way 1
21 LCIMB	Line Command Initial Modified Bit Shows the initial state of the modified bit. If this field = 0, and the command uses cache address and way, this field shows the initial state of the modified bit, which is 0. If the command uses physical address and a hit, this field shows the initial state of the modified bit, which is 0. If the command uses physical address and is a miss, this field reads 0. If this field = 1, and the command uses cache address and way, this field shows the initial state of the modified bit, which is 1. If the command uses physical address and way, this field shows the initial state of the modified bit, which is 1. If a miss, this field reads 0. 0b - Initial state 0 1b - Initial state 1
20 LCIVB	Line Command Initial Valid Bit Shows the initial state of the valid bit. If this field = 0, and the command uses cache address and way, this field shows the initial state of the valid bit, which is 0. If the command uses physical address and a hit, this field shows the initial state of the valid bit, which is 0. If the command uses physical address and is a miss, this field reads 0. If this field = 1, and the command uses cache address and way, this field shows the initial state of the valid bit, which is 1. If the command uses physical address and way, this field shows the initial state of the modified bit, which is 1. If a miss, this field reads 0. 0b - Initial state 0 1b - Initial state 1
19-17 —	Reserved
16 TDSEL	Tag Or Data Select Selects tag or data for search and read or write commands. 0b - Data 1b - Tag
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 WSEL	Way Select Selects the way for line commands. 0b - Way 0 1b - Way 1
13-2 CACHEADDR	Cache Address Represents bits [12:2] of the cache address: CLCR[13:5] accesses the tag arrays. CLCR[13:3] accesses the data arrays. CLCR[2] specifies which half of the 64 bits of data to place in Cache Read/Write Value (CCVR) .
1 —	Reserved
0 LGO	Initiate Cache Line Command Initiates the cache command that CLCR[LCMD] , CLCR[LADSEL] , and CLCR[LACC] indicate, if you write 1 to this field. Reading this field indicates whether a cache command is active. NOTE • This field remains 1 until the command completes. Writing 0 has no effect. • This field is shared with CSAR[LGO]. You can write either register address bit[0] interchangeably. This saves cache writes (helps performance), when performing cache operations. Cache line operations start when you write to Cache Line Control (CLCR) with CLCR[LGO] being 1 or the cache search address can be changed and a search is started by writing to Cache Search Address (CSAR) with CSAR[LGO] being 1. 0b - Write: no effect; Read: no line command active 1b - Write: initiate line command; Read: line command active

44.6.1.4 Cache Search Address (CSAR)

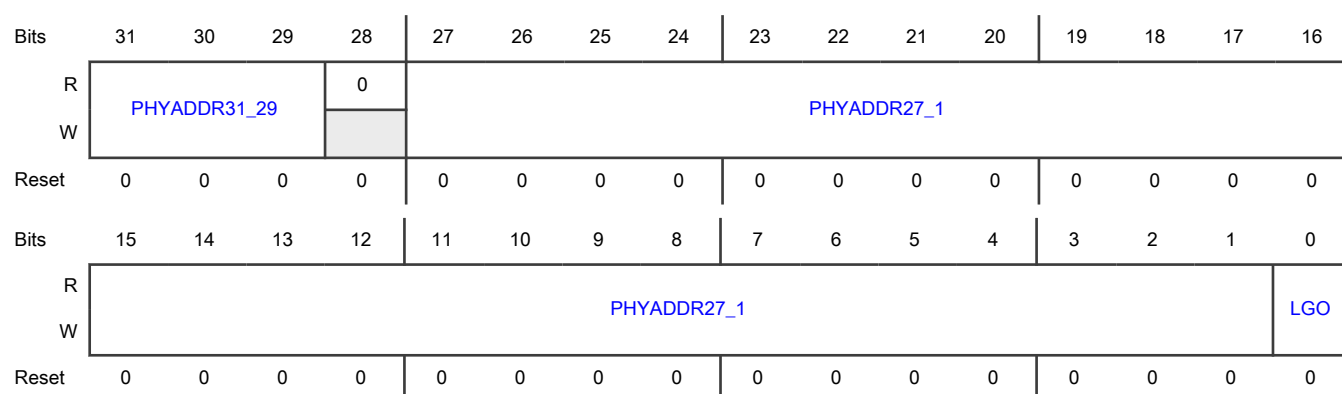
Offset

Register	Offset
CSAR	808h

Function

Defines the physical address for cache line commands, when [CLCR\[LADSEL\]](#) = 1.

Diagram



Fields

Field	Function
31-29 PHYADDR31_29	Physical Address PHYADDR31_29 represents bits [31:29] and PHYADDR27_1 represents bits [27:1] of the system address. CSAR[31:14] bits are used for tag compare CSAR[13:5] bits are used to access the tag arrays CSAR[13:3] bits are used to access the data arrays CSAR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (PSCCVR) CSAR[1] bit is not used. Writes to this bit are allowed, and the read value will be whatever value was previously written.
28 —	Reserved Always read as zero.
27-1 PHYADDR27_1	Physical Address PHYADDR31_29 represents bits [31:29] and PHYADDR27_1 represents bits [27:1] of the system address. CSAR[31:14] bits are used for tag compare CSAR[13:5] bits are used to access the tag arrays CSAR[13:3] bits are used to access the data arrays CSAR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (PSCCVR) CSAR[1] bit is not used. Writes to this bit are allowed, and the read value will be whatever value was previously written.
0 LGO	Initiate Cache Line Command Initiates the cache command that CSAR[PHYADDR] indicate, if you write 1 to this field. Reading this field indicates whether a cache command is active.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE • This field remains 1 until the command completes. Writing 0 has no effect. • This field is shared with CLCR[LGO]. 0b - Write: no effect; Read: no line command active 1b - Write: initiate line command; Read: line command active

44.6.1.5 Cache Read/Write Value (CCVR)

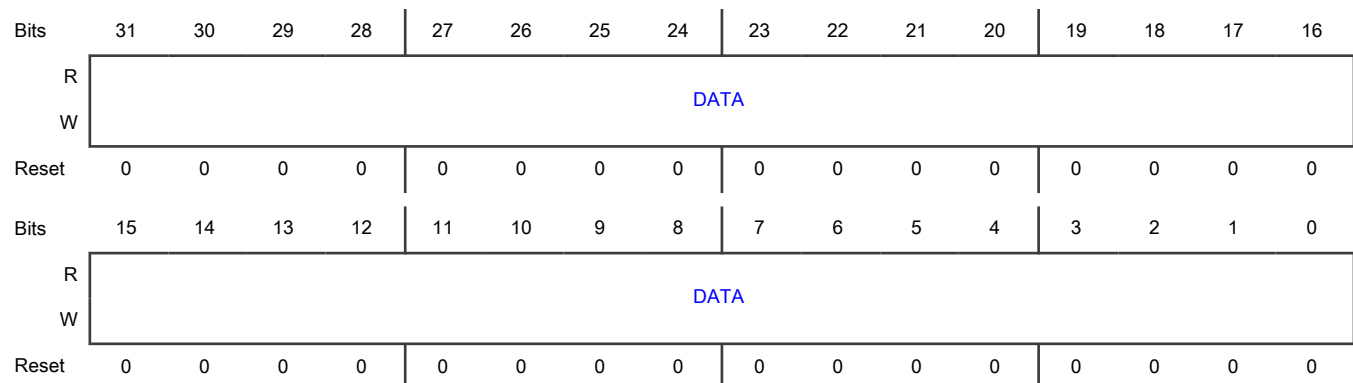
Offset

Register	Offset
CCVR	80Ch

Function

Sources write data or return read data when a search line command is requested ([CLCR\[LCMD\]](#) = 00). [CLCR\[LACC\]](#) determines whether the access is a read or write, and [CLCR\[TDSEL\]](#) determines whether the access is to the data or tag arrays.

Diagram



Fields

Field	Function
31-0 DATA	Cache Read/Write Data Used to read or write data during tag search and data search. For tag search, read or write: • CCVR[31:14] for tag array R/W value.

Table continues on the next page...

Field	Function
	• CCVR[13:5] for tag set address on reads; unused on writes. • CCVR[4:2] are reserved. • CCVR[1] for tag modify bit. • CCVR[0] for tag valid bit. For data search, read or write: • CCVR[31:0] for data array R/W value.

Chapter 45

CACHE64 Policy Select (POLSEL)

45.1 Chip-specific CACHE64 POLSEL information

Each of the CACHE64 modules interface to a FLEXSPI octal/quad SPI module for access to off-chip memory. The 128 Mbyte space divided into three policy regions, is allocated for each CACHE64 POLSEL interface. POLICY_SEL register has three 2-bit fields associated with the three defined regions. Each field designates the policy to be applied to addresses falling within that region. The three policies are completely independent so, it is possible to assign the same policy to all three regions.

Name	Default value	Description
APB_ADDR_WIDTH	12	APB Address bus width.
APB_DATA_WIDTH	32	APB Data bus width.
BASE_ADDR	32'h000	Memory Map base address position for this IP.
CACHE64_SIZE	32	Cache64 Size limit in KByte.
CACHE64_BPW	32	Cache64 bits per word.

45.2 Overview

POLSEL monitors incoming AHB addresses to define the policy that the cache controller (CACHE64_CTRL) uses. Both POLSEL and the cache controller work in tandem with each other.

45.2.1 Features

- Provides cache policy information to the cache controller.
- Supports up to three regions, each with a programmable cache policy:
 - Each region can be independently designated as noncache, write-through cache, and write-back cache.
 - User-defined boundaries are supported between the three regions with 1 KB granularity.

45.3 Functional description

POLSEL permits dividing the memory space that the cache controller accesses into three contiguous regions and applies one of these three cache policies to each region:

- First region (region 0) starts at the bottom (lowest address) in the target address space and goes up to the top of the 1 KB boundary designated in [REG0_TOP](#).
- Second region (region 1) starts at the next address and extends to the top of the boundary designated in [REG1_TOP](#).
- Third region starts at the address after that and continues to the top of the memory space accessed via the cache controller.

45.3.1 Region mapping

POLSEL controls the cache policy for memory between the base address and end address. Within the POLSEL memory area, you can define up to three contiguous memory regions, with each region having independently programmable cache policy settings.

As shown in the following figure, [REG0_TOP](#) and [REG1_TOP](#) control the boundaries between the regions. These registers provide bits 26-11 of the address and the lower 10 bits are always 1s. For example, if [REG0_TOP](#) is 0001_0000h, then region 0 ends at BASE_ADDRESS + 0001_03FFh. Region 1 starts at 0001_0400h.

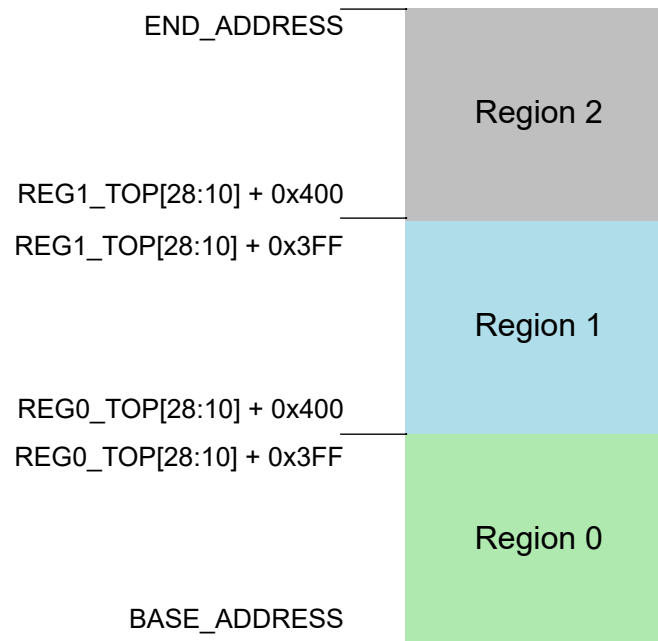


Figure 198. Cache memory regions

For POLSEL implementations that have multiple memory regions (more than one base address and end address), the regions are aliased across all of the memory regions. Therefore, the configured cache policies are also aliased across all of the memory regions.

45.3.2 Region priority

When you access an address, POLSEL attempts to match it to one of the regions in a sequential order that starts with region 0. In pseudocode, the region hit logic does this:

```
if (address < REG0_TOP + 0x3FF)
    address hits in Region 0;
else
    if (address < REG1_TOP + 0x3FF)
        address hits in Region 1;
    else
        address hits in Region 2;
```

NOTE

If you configure `REG0_TOP` and `REG1_TOP` to create overlapping regions, then the lowest number region in which the address hits, is used.

45.4 Application information

This section provides an example POLSEL configuration. The example assumes:

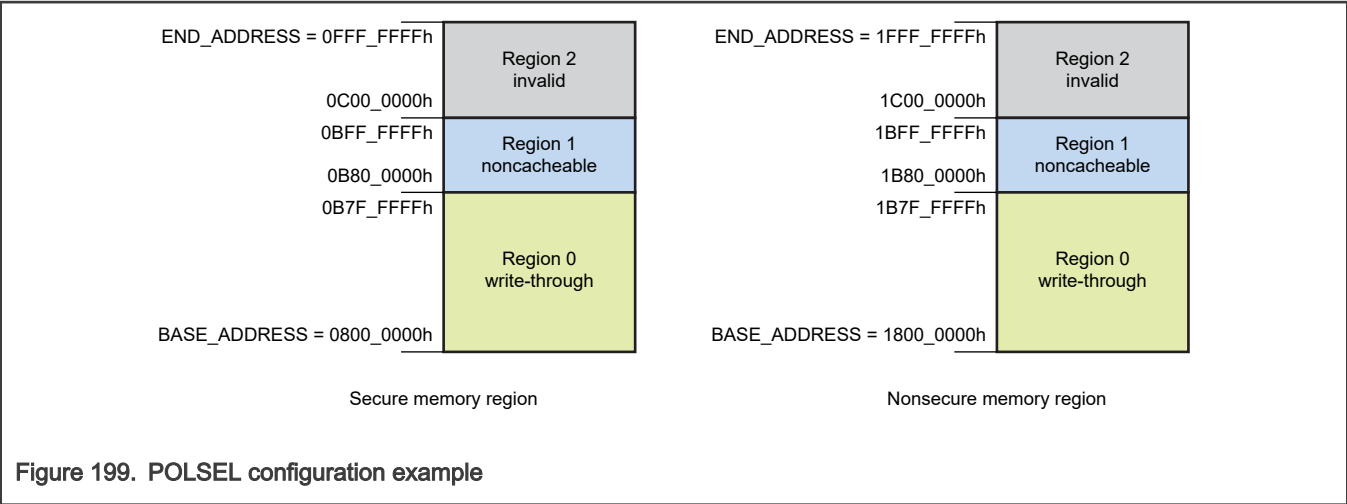
- The MCU has two memory regions mapped to POLSEL:

- Nonsecure region: BASE_ADDRESS = 0800_0000h, END_ADDRESS = 0FFF_FFFFh
- Secure region: BASE_ADDRESS = 1800_0000h, END_ADDRESS = 1FFF_FFFFh
- Region 0: 56 MB is configured as write-through.
- Region 1: 8 MB is configured as noncacheable.
- Region 2: The remaining memory region is unused and configured as invalid.

The following register settings are used to achieve this configuration:

- REG0_TOP = 037F_FC00h
- REG1_TOP = 03FF_FC00h
- POLSEL = 0000_0031h

The following figure shows the resulting memory configuration.



45.5 Memory map and register definition

The POLSEL registers are included in the register space of the CACHE64 module, utilizing addresses that CACHE64 does not use internally.

45.5.1 POLSEL register descriptions

45.5.1.1 CACHE64_POLSEL memory map

CACHE64_POLSEL0 base address: 4003_3000h

CACHE64_POLSEL1 base address: 4003_4000h

Offset	Register	Width (In bits)	Access	Reset value
14h	Region 0 Top Boundary (REG0_TOP)	32	RW	02AA_A800h
18h	Region 1 Top Boundary (REG1_TOP)	32	RW	0555_5400h
1Ch	Policy Select (POLSEL)	32	RW	0000_0000h

45.5.1.2 Region 0 Top Boundary (REG0_TOP)

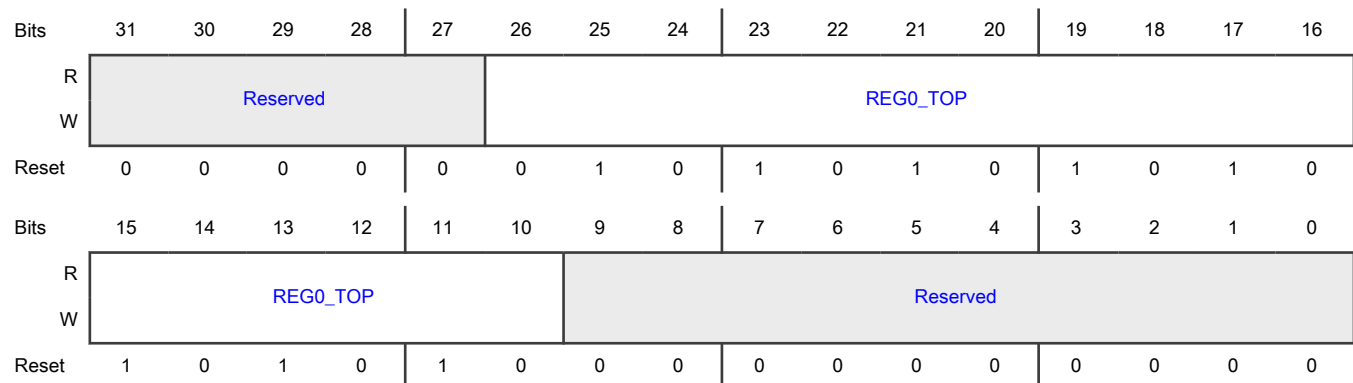
Offset

Register	Offset
REG0_TOP	14h

Function

Provides bits 28-10 of the region 0 compare address. The actual granularity implemented is 1 KB.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-10 REG0_TOP	Upper Limit Of Region 0 Provides bits 26-10 of the region 0 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For Example, if REG0_TOP[26:10] = 0b00000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).
9-0 —	Reserved

45.5.1.3 Region 1 Top Boundary (REG1_TOP)

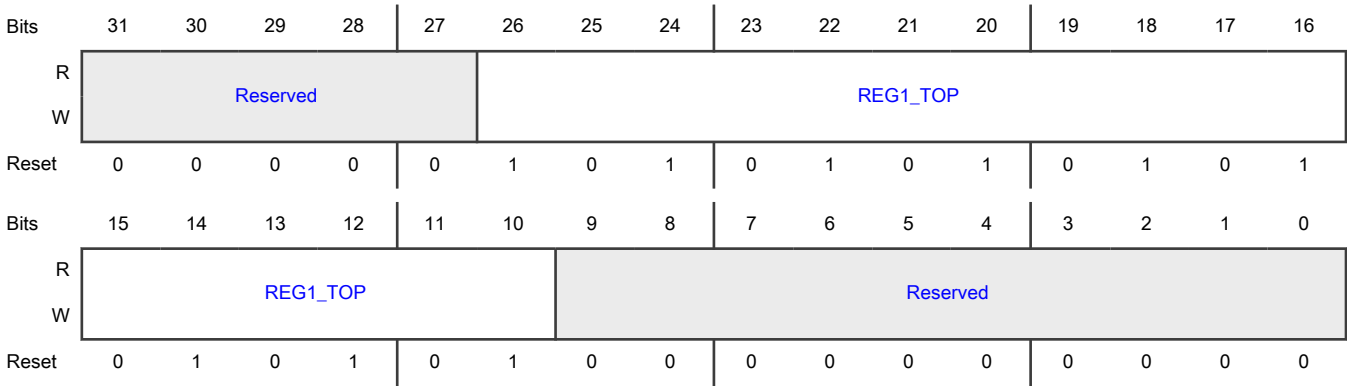
Offset

Register	Offset
REG1_TOP	18h

Function

Provides bits 28-10 of the region 1 compare address. The actual granularity implemented is 1 KB.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-10 REG1_TOP	Upper Limit Of Region 1 Provides bits 26-10 of the region 1 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For example, if REG1_TOP[26:10] = 0b000000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).
9-0 —	Reserved

45.5.1.4 Policy Select (POLSEL)

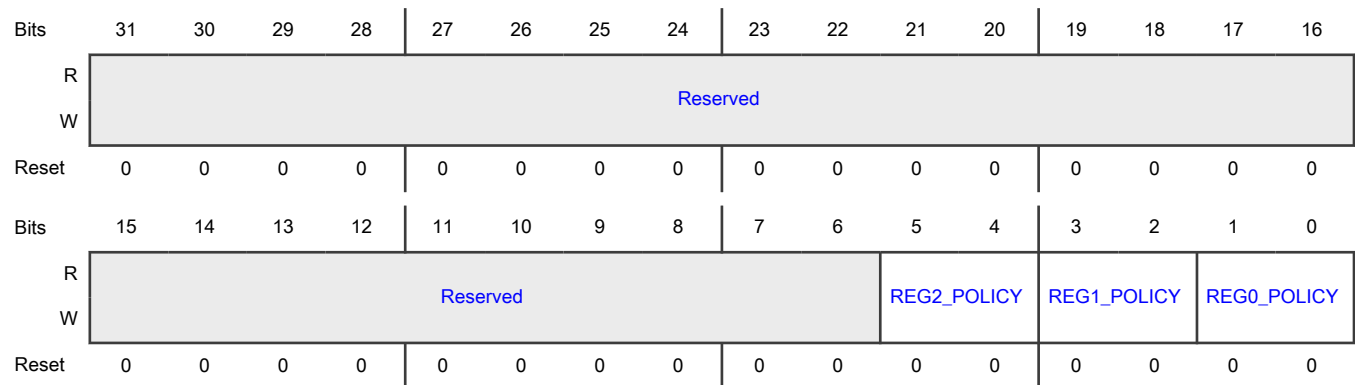
Offset

Register	Offset
POLSEL	1Ch

Function

Specifies the policy to be applied to all addresses falling within the designated space.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-4 REG2_POLICY	Policy Select For Region 2 Specifies the policy to be applied to all addresses falling within the space designated as region 2. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid
3-2 REG1_POLICY	Policy Select For Region 1 Specifies the policy to be applied to all addresses falling within the space designated as region 1. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid
1-0 REG0_POLICY	Policy Select For Region 0 Specifies the policy to be applied to all addresses falling within the space designated as region 0. 00b - Noncacheable 01b - Write-through 10b - Write-back 11b - Invalid

Chapter 46

LCD Interface (LCDIF)

46.1 Chip-specific LCDIF information

Table 475. Reference links to related information

Topic	Related module	Reference
Full description	LCDIF	LCDIF
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

46.1.1 Module instances

This device has one instance of the LCDIF module, LCDIF0.

46.1.2 Chip-specific LCDIF Signals

Table 476. LCDIF Signals

Signal	Description
LCD_D[23:0]	DPI Data
LCD_ENABLE	DPI Display enable
LCD_DTCLK	DPI Display pixel clock
LCD_HSYNC	DPI Display horizontal sync
LCD_VSYNC	DPI Display vertical sync
DBI_CSX	Type A or B CSX Chip Select, active low
DBI_DCX	Type A or B D/CX: Data/Command. Command is active low
DBI_RWDX	Indicates if current cycle is for read (1) or write (0). For Type A R/WX Read data when high or writes when low. For Type B RDX Read data when high.
DBI_WRX	Type A or B WRX Write signal. Host processor writes data at falling edge
DBI_E	Enable DBI E mode: Clock E = 0, Fixed E = 1
DBI_D[13:0]	DBI Data

46.2 Overview

NOTE

This chapter describes the Vivante® IP (DCNano) and contains copyright material disclosed with permission of VeriSilicon.

The LCDIF high-performance graphics core is based on the Vivante DCNano IP core that can be used for reading rendered images from the frame buffer. In addition to providing hardware cursor patterns, the display controller performs format conversions, dithering and gamma corrections. With an addressable screen size up to 1920x1080 at 60fps for linear formats and 1024x768 at 60fps for tiled formats, the LCDIF provides powerful graphics processing with low power consumption.

46.2.1 Block Diagram

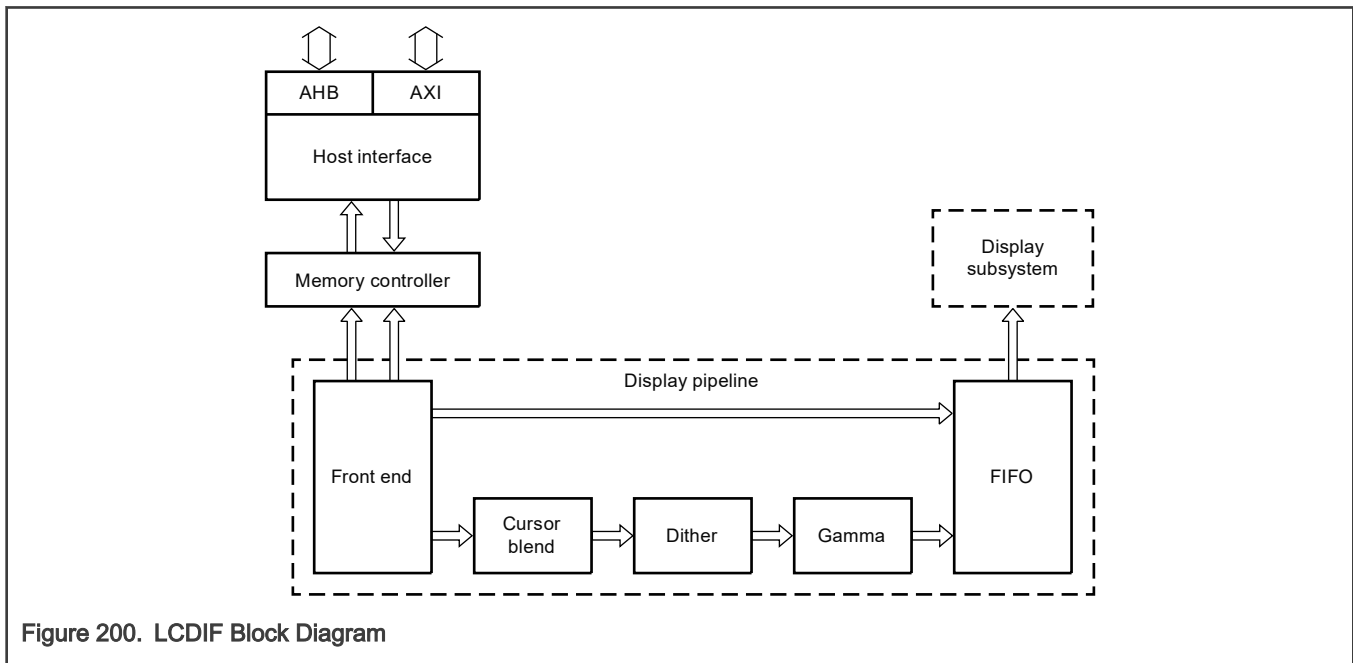


Figure 200. LCDIF Block Diagram

46.2.2 Features

Display controller key features are:

- Video Timing Generation
 - HSYNC, VSYNC, Data Enable (DE) signals
 - Programmable timers
- MIPI Display Protocols
 - Display Pixel Interface-2 (DPI-2) formats
 - DPI 24-bit, 18-bit (2 configs) and 16-bit support (3 configs)
 - Display Bus Interface 2.0 (DBI-2)
- Display Interface
 - Parallel Pixel Output with 24-bit Data, HSync, VSync, Data enable
 - Easily adaptable to external serialization logic, e.g. HDMI
- Display
 - Display size up to 1920x1080 at 60fps for linear formats and 1024x768 at 60fps for tiled formats
 - Sync and blank signals
 - Gamma and dither tables
- Input Formats
 - ARGB2101010/ARGB8888/ARGB1555/RGB565/ARGB4444

- YUV422 packed & semi planar (YUY2, NV12)
- Format Conversion
 - Pixel inputs accepted from multiple RGB formats
 - Pixel output is 24 bit RGB in multiple formats
- Output Formats
 - RGB888/DPI_D16CFG1/DPI_D16CFG2/DPI_D16CFG3/DPI_D18CFG1/ DPI_D18CFG2/ DPI_D24
- Hardware Cursor
 - Supports ARGB888 and Mask cursor formats
- Color
 - Overlay with coordinate generator
 - Alpha Blending: 8 Porter Duff Blending modes
- Dither and Gamma Correction
 - A separate Look Up Table for Dither
 - A separate Look Up Table for Gamma Correction

46.2.2.1 LCDIF Host and Memory Interface Features

Table 477. Host and memory interface features

Feature	GPU Support
AHB interface	32-bit AHB slave for register access
AXI interface	64-bit AXI master for external memory access and to read frame buffer data
AXI address width	32 bits

46.2.2.2 LCDIF Power Management Features

Feature	Support
Low power CMOS technology compatible	Yes
Control bits for RAM clock gating	Yes
Independent clock for the display	Yes
Independent enable bit for the display	Yes

46.2.2.3 LCDIF Display Features

Table 478. Display features

Feature	Support
Maximum display size	1080p

Table continues on the next page...

Table 478. Display features (continued)

Feature	Support
	NOTE See the Chip-specific LCDIF information for details about Maximum display resolution.
Sync and blank signals	Yes
Independent gamma and dither	Yes
Double buffered registers for flipping from shadow to working set	Atomic Commit, single cycle flip
Rotation support	No
Coordinate system	32K x 32K

46.2.2.4 LCDIF MIPI Protocol Features

Table 479. MIPI Protocol features

Feature	Support
Display Pixel Interface (DPI-2)	Yes
Display Bus Interface support (DBI-2)	Yes
DPI: selectable enable and polarity settings for HYSNC, VSYNC and DE	Yes
DPI: programmable timing control parameters	Yes
DPI: output color coding	Yes 16 bit: 2 mappings 18/24 bit: 3 mappings
DPI shutdown and color mode control	Yes, use GPIO in SOC
DBI Type Support	Type A fixed E mode Type A clock E mode Type B
DBI interface data widths	8/9/16

46.3 Functional Description

46.3.1 Display Controller Clocking

LCDIF employs a simple clocking scheme. There are three independent clock domains in LCDIF:

- AXI bus clock and LCDIF core clock domain, which is derived from the clk pin,
- AHB interface clock domain, which is derived from the HCLK pin, and
- LCDIF Pixel Clock used in the Display's outputFifo, and this clock is also output to the display with the name of LCD_DTCLK.

NOTE

See LCDIF chip-specific chapter for details on the pins used in this chip.

The AHB interface clock is HCLK. clk is the functional clock for the core logic as well as the AXI bus clock. Communication between the different domains in the display controller occurs mostly by way of asynchronous FIFOs. All the clocks are asynchronous to each other.

Table 480. Clocks

Clock Name	Description	I/O
clk	AXI bus clock/LCDIF core clock. Clock source from pin.	I
HCLK	AHB interface clock (for AHB slave)	I
LCDIF Pixel Clock	Pixel clock for the display	Input, also output to the display as LCD_DTCLK

46.3.2 DBI output timing

The method for programming the DBI interface timing will vary depend on the DBI Type. For DBI output timing, a basic time unit configured through the register field DbiConfig0[DBI_AC_TIME_UNIT]. Basic time unit is DbiConfig0[DBI_AC_TIME_UNIT] + 1. Other timing related fields are all based on this field.

For DBI Type A and Type B: The basic time unit counts the "clk" cycle. The value of field DbiWrChar10[DBI_WR_EOR_WR_ASSERT] has to be at least 1 to conform with the MIPI DBI spec, and the following rule needs to be maintained:

$$\text{DbiWrChar10[DBI\_WR\_EOR\_WR\_ASSERT]} < \text{DbiWrChar20[DBI\_WR\_EOR\_DE\_ASRT]} < \text{DbiWrChar10[DBI\_WR\_PERIOD]}$$

When value of DbiWrChar10[DBI_WR_PERIOD] is set, that means the output clk frequency is:

$$\text{Output clk frequency} = \frac{\text{clk}}{(\text{DbiWrChar10[DBI\_WR\_PERIOD]} * (\text{DbiConfig[DBI\_AC\_TIME\_UNIT]} + 1))}$$

Equation 1. Output clk frequency

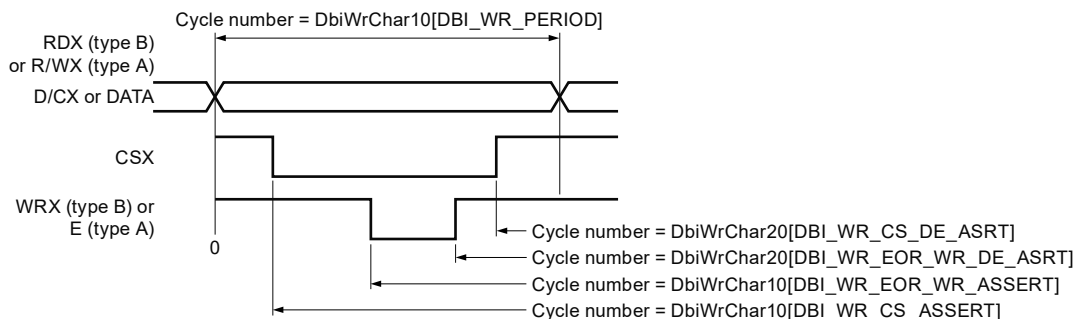


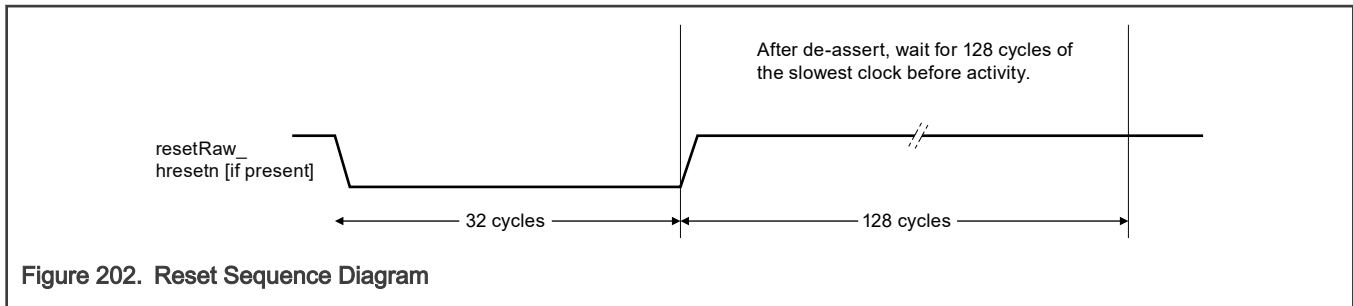
Figure 201. DBI output timing definitions for Type A and Type B

46.3.3 Reset

Reset input is synchronized to the core clock domain by a dual flip-flop synchronizer. The resetRaw_ signal must be asserted for a minimum of 32 core clock cycles, using the slowest of the three/four clocks (see [Table 480](#)).

After deassertion, wait for 128 cycles of the slowest clock before starting any activity.

All registers use synchronous reset flip-flops to minimize noise sensitivity on the reset line. Note, however, that this means the clocks must be running in order to reset the state of the core. Assuming the clock is generated from a PLL, the diagram in the figure below illustrates the proper reset sequence.



resetRaw_ is used to reset the AXI interface and core clocks. When the design includes AHB, hresetn is used to reset the AHB interface clock. The resetRaw_ and hresetn signals can be tied together.

For reset signal assertion and de-assertion to work properly:

- Best practice is to have all reset signals assert and de-assert together.
- If ordering is needed, the AXI resets should be asserted first and de asserted last. The other resets can be in between.
- For AXI reset:
 - Make sure no AXI transactions are in progress.
 - AXI bus clock needs to be running when asserting and de-asserting ARESETn
- For Core reset:
 - Make sure all the LCDIF blocks are idle.
- For AHB reset:
 - Make sure no AHB transaction is in progress.
 - HCLK needs to be running when asserting and de-asserting HRESETn.
- If reset signals are not de-asserting together, the clock (preferably an AXI clock) associated with the last reset is the clock that should be used for the 128 cycle wait.
- All de-assertions except the last should adhere to the 32-cycle assertion wait time:
 - HCLK and AXI bus clock may de-assert separately.

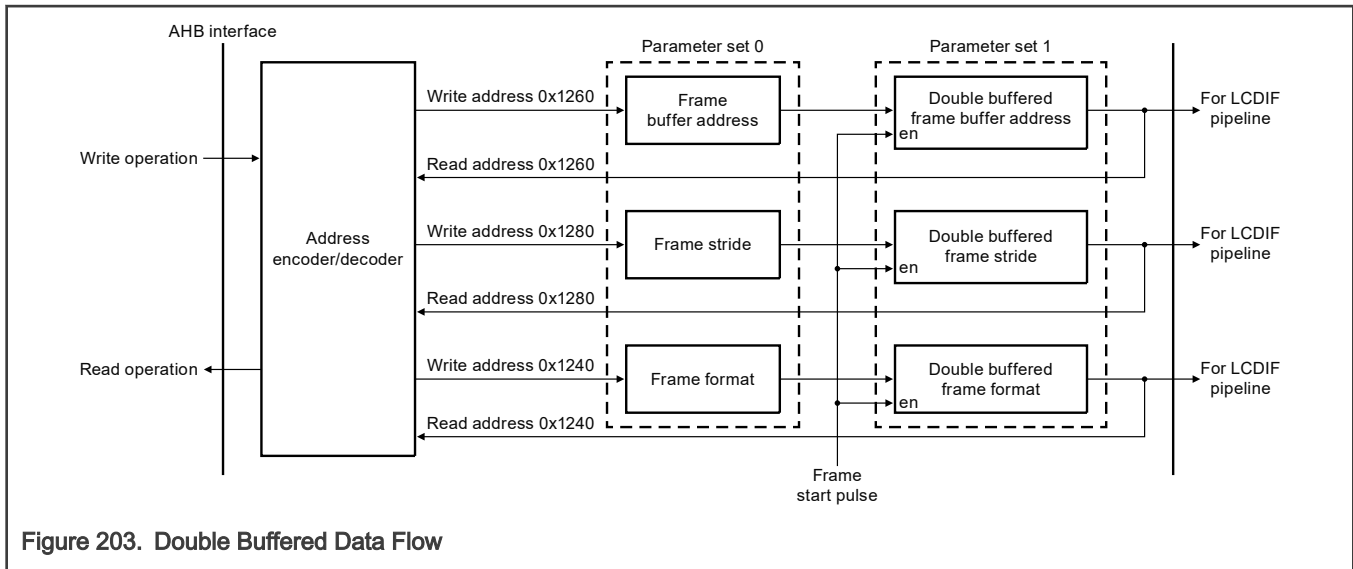
46.3.4 Interrupts

The Interrupt rises at the last pixel of visible area in a frame and holds high until the DisplayIntr register is read, then is automatically cleared. After reading the interrupt register, the CPU should configure the LCDIF registers for the next frame (for example, FrameBufferAddress0, CursorAddress, etc.) All changes will be updated at the end of vSync, and the LCDIF's DMA will fetch data after the vSync.

46.3.5 Double Buffered Display Management

The display controller has many double-buffered register fields for managing the display. The diagram below outlines how it can be established which set of the double-buffered register fields the host has ownership of and which set is currently in use by the display.

For example, the frame buffer address has an offset address of 0x1260. A write to the address will change the value of set 0. A read from the address will return back the value of set 1 (in use). Only when one frame starts (for DPI, it is a VSYNC end point and for DBI, it is a write-memory-start command issued), is the value of set 0 will be loaded to set 1.



46.4 Initialization

- Initialization should be done after the panel is powered ON.
- PanelConfig0[CLOCK] = 0b

NOTE

In DBI mode, make sure PanelConfig0[CLOCK] = 0b. When the pixel clock is active, this clock will appear at the output LCD_DTCLK. In DBI mode, this core ignores and does not care about the value of any of the other bits in the PanelConfig0 and PanelTiming0 registers.

- Set the color mapping for the DBI output in DbiParam[DBI_TYPE] to TYPE_AFIXED_E, TYPE_ACLOCK_E, or TYPE_B:
DbiParam[DBI_TYPE] = TYPE_AFIXED_E, TYPE_ACLOCK_E, or TYPE_B
- Write 1b to DbiParamReset0[DBI_IF_LEVEL_RESET] to put the DBI bus into idle status:
DbiParamReset0[DBI_IF_LEVEL_RESET] = 1b
- Set the DBI output timing parameters according to the DBI Output Timing Definitions for Types A and B diagram and the actual panel's AC timing specifications:
 - DbiParamWrChar10
 - DbiParamWrChar20
- Set the output bus mode to DBI:
 - DbiParamConfig0[BUS_OUTPUT_SEL] = 1b
- Use the DbiParamCmd0 register to write to send configuration data. Commands are entered one by one:
 - Use DbiParamCmd0[DBI_COMMANDFLAG] = 00b to indicate an address is to be written, where DbiParamCmd0[DBI_COMMAND_WORD] contains the current address to be sent to the DBI port.
 - Use DbiParamCmd0[DBI_COMMANDFLAG] = 10b to write one parameter to the panel, where DbiParamCmd0[DBI_COMMAND_WORD] contains the current parameter.

46.4.1 Refresh panel frame buffer (single frame example)

- Set timing parameters according to the actual panel specification. Parameters are set before the first frame refresh, then ignored after that.
 - HDisplay0[TOTAL] = HDisplay0[DISPLAY_END] = panel width size

- HSync0[START] = 0, HSync0[END] = panel width size
- VDisplay0[TOTAL] = panel height size + 1 (The extra '1' is used for waiting for the data FIFO to be filled, and when tile data format should be '2')
- VDisplay0[DISPLAY_END] = panel height size
- VSync0 is don't care
- Set registers for cursor/gamma/dither function (if needed):
 - Cursor*
 - Gamma*
 - DisplayDither*
- Set frame buffer address and stride registers:
 - FrameBufferAddress0
 - FrameBufferOrigin0
 - FrameBufferStride0
 - DcTileUvFrameBufferAdr0/DcTileUvFrameBufferStr0 (when tiled ARGB, FrameBufferStride0 = one line stride x 4 lines)
- Configure the data format of the frame buffer:
 - FrameBufferConfig0 (Do not set bit FrameBufferConfig0[RESET] to 1b and must set bit FrameBufferConfig0[OUTPUT] to 1b)
 - DcTileInCfg0 (when linear ARGB8888/ARGB1555/ARGB565/ARGB444 format is expected, tile format must be set to "none")
- Use the DbiCmd0 register to write to send configuration data. Commands are entered one by one:
 - Use DbiCmd0[DBI_COMMANDFLAG] = 01b to indicate the write to the DBI port. For this flag, values in DbiCmd0[DBI_COMMAND_WORD] are ignored.
- Use the interrupt registers to wait for the signal that the write is complete, or check the interrupt status:
 - DisplayIntr
 - DisplayIntrEnable

46.4.2 Partial refresh of panel frame buffer (single frame example)

- Set timing parameters according to the actual panel specification. Parameters are set before the first frame refresh, then ignored after that.
 - HDisplay0[TOTAL] = HDisplay0[DISPLAY_END] = partial window width size
 - HSync0[START] = actual clipped window x start (see [Notes for Window Alignment for DBI](#))
 - HSync0[END] = actual clipped window x end
 - VDisplay0[TOTAL] = partial window height size + 1 (The extra '1' is used for waiting for the data FIFO to be filled, and when tile data format should be '2')
 - VDisplay0[DISPLAY_END] = partial window height size
 - VSync0 don't care for linear format.
- For tiled data:
 - VSync0[START] = actual clipped window y start
 - VSync0[END] = actual clipped window y end

- Set registers for cursor/gamma/dither function (if needed):
 - Cursor*
 - Gamma*
 - DisplayDither*
- Set frame buffer address and stride registers:
 - FrameBufferAddress0
 - FrameBufferOrigin0
 - FrameBufferStride0
 - DcTileUvFrameBufferAdr0/DcTileUvFrameBufferStr0 (when tiled ARGB, FrameBufferStride0 = one line stride x 4 lines)
- Configure the data format of the frame buffer:
 - FrameBufferConfig0 (Do not set bit FrameBufferConfig0[RESET] to 1b and must set bit FrameBufferConfig0[OUTPUT] to 1b)
 - DcTileInCfg0 (when linear ARGB8888/ARGB1555/ARGB565/ARGB444 format is expected, tile format must be set to "none")
- Use the DbiCmd0 register to write to send configuration data. Commands are entered one by one:
 - Use DbiCmd0[DBI_COMMANDFLAG] = 01b to indicate the write to the DBI port. For this flag, values in DbiCmd0[DBI_COMMAND_WORD] are ignored.
- Use the interrupt registers to wait for the signal that the write is complete, or check the interrupt status:
 - DisplayIntr
 - DisplayIntrEnable

46.4.3 Notes for Window Alignment for DBI

If A4R4G4G4B4/A1R5G5B5/R5G6B5, the window size must be aligned to 64x (burst length 16, 64-bit bus width) and its start x position must also be aligned to 0x80 (do not support 4K boundary). If A8R8G8B8, the window size must be aligned to 32x (burst length 16, 64-bit bus width) and its start position must be aligned to 0x80 (do not support 4K boundary).

Table 481. Window alignment

DBI window format alignments	A4R4G4G4B4 A1R5G5B5 R5G6B5	A8R8G8B8 Linear	A8R8G8B8 Tiled
Window size alignment X(burst length 16, 64-bit bus width)	64x	32x	width 8x
Window size alignment Y(burst length 16, 64-bit bus width)	NA	NA	height 8x
Start X position alignment	0x80 (does not support 4K boundary)		0x40

46.5 Memory Map and Register Definition

46.5.1 LCDIF register descriptions

46.5.1.1 LCDIF memory map

LCDIF base address: 4021_0000h

Offset	Register	Width (In bits)	Access	Reset value
1240h	Frame Buffer Configuration 0 (FrameBufferConfig0)	32	RW	0100_0000h
1260h	Starting Address of the Frame Buffer (FrameBufferAddress0)	32	RW	0000_0000h
1280h	Stride of the Frame Buffer in Bytes (FrameBufferStride0)	32	RW	0000_0000h
1360h	Configuration for Dithering (DisplayDitherConfig0)	32	RW	0000_0000h
1380h	Dither Table Low (DisplayDitherTableLow0)	32	RW	0000_0000h
13A0h	Dither Table High (DisplayDitherTableHigh0)	32	RW	0000_0000h
13C0h	Panel Configuration (PanelConfig0)	32	RW	8000_0000h
13E0h	Timing for Hardware Panel Sequencing (PanelTiming0)	32	RW	0000_0000h
1400h	Horizontal Total and Display End Counters (HDisplay0)	32	RW	0000_0000h
1420h	Horizontal Sync Counters (HSync0)	32	RW	0000_0000h
1480h	Vertical Total and Display End Counters (VDisplay0)	32	RW	0000_0000h
14A0h	Vertical Sync Counters (VSync0)	32	RW	0000_0000h
14C0h	Current x,y Location of Display Controller (DisplayCurrentLocation0)	32	R	0000_0000h
14E0h	Index into Gamma Table (GammaIndex0)	32	RW	0000_0000h
1500h	Translation Values for the Gamma Table (GammaData0)	32	RW	0000_0000h
1520h	Configuration for the Cursor (CursorConfig)	32	RW	0000_0000h
1530h	Address of the Cursor Shape (CursorAddress)	32	RW	0000_0000h
1540h	Location of the cursor on the owning display (CursorLocation)	32	RW	0000_0000h
1550h	Background Color for Masked Cursors (CursorBackground)	32	RW	0000_0000h
1560h	Foreground Color for Masked Cursors (CursorForeground)	32	RW	0000_0000h
1600h	Display Interrupt (DisplayIntr)	32	RW	0000_0000h
1610h	Interrupt Enable for Display_0 (and Display_1 if present) (DisplayIntrEnable)	32	RW	0000_0011h
1620h	DBI Configuration 0 (DbiConfig0)	32	RW	0000_0040h
1640h	Reset DBI Interface to Idle State (DbiIfReset0)	32	RW	See section
1660h	DBI Write Characteristics 1 (DbiWrChar10)	32	RW	0000_0000h
1680h	DBI Write Characteristics 2 (DbiWrChar20)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
16A0h	DBI Command In/Out Port (DbiCmd0)	32	RW	0000_0000h
16C0h	DPI Configuration 0 (DpiConfig0)	32	RW	0000_0005h
16F0h	Revision for the LCDIF Peripheral in BCD (DcChipRev)	32	R	0000_5543h
1700h	Shows the release date for the IP in YYYYMMDD (year, month) (DcChipDate)	32	R	2018_0612h
1720h	Patch Revision (DcChipPatchRev)	32	R	0000_0003h
1740h	Tile Input Configuration (DcTileInCfg0)	32	RW	0000_0000h
1760h	UV Frame Buffer Address when Tile Input (DcTileUvFrameBufferAdr0)	32	RW	0000_0000h
1780h	UV Frame Buffer Stride when Tile Input (DcTileUvFrameBufferStr0)	32	RW	0000_0000h
17B0h	Product ID (DcProductId)	32	R	0200_0361h
1820h	Debug Counter Select (DebugCounterSelect0)	32	RW	0000_0000h
1840h	Debug Counter Value (DebugCounterValue0)	32	RW	0000_0000h

46.5.1.2 Frame Buffer Configuration 0 (FrameBufferConfig0)

Offset

Register	Offset
FrameBufferConfig0	1240h

Function

Frame Buffer and Timing control. Some fields are double buffered.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved			FLIP_I N...	Reserved				UNDE RFL...	Reserved				Reserved			VALID
W				BACK_ PR...													
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			GAMM A	Reserved			SWITC HP...	OUTP UT	Reserved			MODE	Reserv ed	FORMAT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-30 —	Reserved
29 BACK_PRESS URE_DISABLE	Disable Back Pressure 0b - Disabled 1b - Enabled
28 FLIP_IN_PROG RESS	Flip in Progress When the frame buffer address gets written to, this bit gets set to one. It will be reset to zero at the start of the next VBLANK when the register gets copied into the working set. 0b - Disabled 1b - Enabled
27-25 —	Reserved
24 UNDERFLOW	Underflow When the display FIFO underflows, this bit gets set to one. Reading this register will reset it back to zero. 0b - Disabled 1b - Enabled
23-21 —	Reserved
20 RESET	Reset Writing a one in this register will force a reset of the display controller. All registers will be copied to the working set and counters will be reset to the end of HSYNC and VSYNC. 1=RESET; Enable reset for the display controller. Write 0 to this bit to reset the display controller, then configure the other registers. Finally write a 1 to this bit to let the display controller start. When the display controller starts, it begins at VBLANK_START, and all registers get flopped to the working set at VSYNC_END. Counters will be reset to the end of HSYNC and VSYNC. For DBI, do not write 1 to this field. Use DbilfReset0[DBI_IF_LEVEL_RESET] instead. 0b - For DBI, this field should be = 0. 1b - Enable DPI Timing, start a DPI transfer.
19-17 —	Reserved
16	Valid

Table continues on the next page...

Table continued from the previous page...

Field	Function
VALID	The valid field defines whether we can copy a new set of registers at the next VBLANK or not. This ensures a frame will always start with a valid working set if this register is programmed last, which reduces the need for SW to wait for the start of a VBLANK signal in order to ensure all states are loaded before the next VBLANK. 0b - Working 1b - Pending
15-13 —	Reserved
12 GAMMA	Gamma When Gamma is enabled, the R, G, and B channels will be routed through the Gamma LUT to perform gamma correction. NOTE This field is double buffered. 0b - Disabled 1b - Enabled
11-10 —	Reserved
9 SWITCHPANEL	Switch Panel When SwitchPanel of panel 0 is enabled, output channel 0 will show the image of panel 1, not panel 0's. When SwitchPanel of panel 0 is disabled, output channel 0 will show its original image which is from panel 0. When SwitchPanel of panel 1 is enabled, output channel 1 will show the image of panel 0, not panel 1's. When SwitchPanel of panel 1 is disabled, output channel 1 will show the original image which is from panel 1. disabled(normal) enabled(switch) output channel 0 panel 0 panel 1 output channel 1 panel 1 panel 0. NOTE This field is double buffered. 0b - Disabled 1b - Enabled
8 OUTPUT	Output When Output is enabled, pixels will be displayed. When Output is disabled, all pixels will be black. This allows a panel to have correct timing but without any pixels. NOTE This field is double buffered. 0b - Disabled 1b - Enabled
7-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4 MODE	Mode of the frame buffer. NOTE All fields are double buffered, i.e., they will only be updated on the next VSYNC pulse. 0b - LINEAR 1b - TILE4x4 INPUT
3 —	Reserved
2-0 FORMAT	The format of the frame buffer. None means there is no frame buffer and the display controller will not produce any output. NOTE This field is double buffered. 0 = NONE; 1 = R4G4B4; 2 = R5G5B5; 3 = R5G6B5; 4 = R8G8B8.

46.5.1.3 Starting Address of the Frame Buffer (FrameBufferAddress0)

Offset

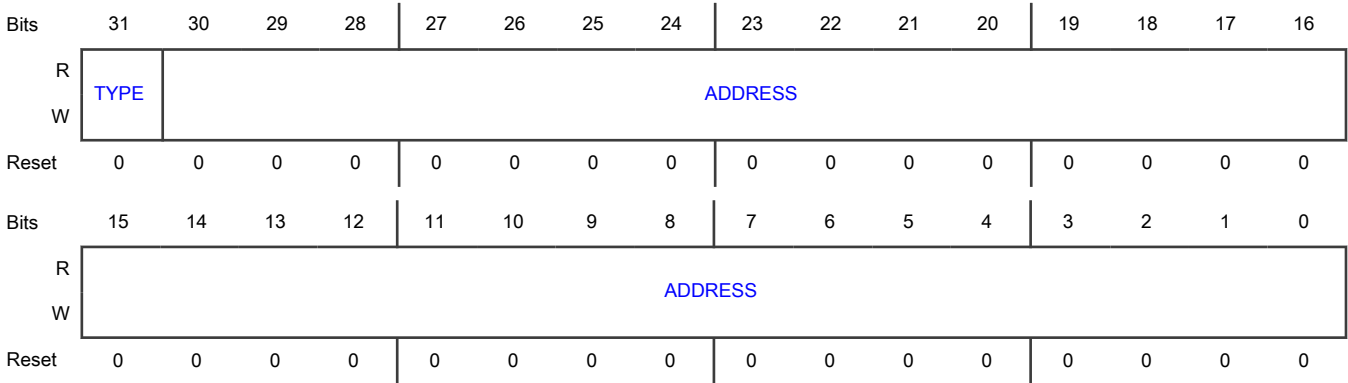
Register	Offset
FrameBufferAddress0	1260h

Function

NOTE

This register is double buffered.

Diagram



Fields

Field	Function
31 TYPE	System Type 0b - System 1b - Virtual system
30-0 ADDRESS	Address

46.5.1.4 Stride of the Frame Buffer in Bytes (FrameBufferStride0)

Offset

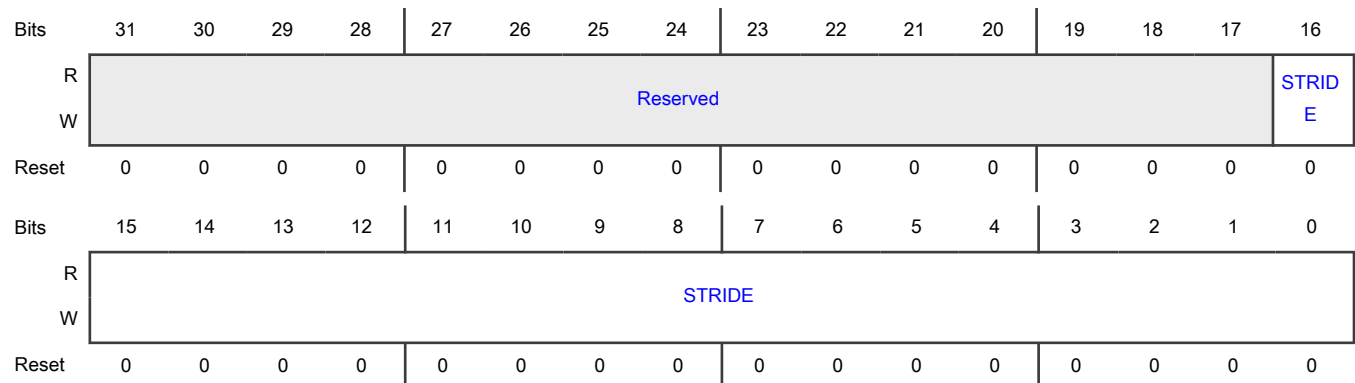
Register	Offset
FrameBufferStride0	1280h

Function

NOTE

This register is double buffered.

Diagram



Fields

Field	Function
31-17 —	Reserved
16-0 STRIDE	Number of bytes from start of one line to next line. This value needs to be 128 byte aligned.

46.5.1.5 Configuration for Dithering (DisplayDitherConfig0)

Offset

Register	Offset
DisplayDitherConfig0	1360h

Function

Some fields are double buffered.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ENAB	Reserved												RED_SIZE		
W	E															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				GREEN_SIZE				Reserved				BLUE_SIZE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 ENABLE	Enable Dithering Enabling dithering allows R8G8B8 modes to show better on panels with less bits-per-pixel. NOTE This field is double buffered. 0b - Disabled 1b - Enabled
30-20 —	Reserved
19-16 RED_SIZE	Red Size Number of important bits for the red channel for the panel. Start counting from MSB.
15-12 —	Reserved
11-8	Green Size

Table continues on the next page...

Table continued from the previous page...

Field	Function
GREEN_SIZE	Number of important bits for the green channel for the panel. Start counting from MSB.
7-4 —	Reserved
3-0 BLUE_SIZE	Blue Size Number of important bits for the blue channel for the panel. Start counting from MSB.

46.5.1.6 Dither Table Low (DisplayDitherTableLow0)

Offset

Register	Offset
DisplayDitherTableLow0	1380h

Function

Contains the dither threshold values.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Y1_X3				Y1_X2				Y1_X1				Y1_X0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Y0_X3				Y0_X2				Y0_X1				Y0_X0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 Y1_X3	Dither threshold value for x,y=3,1.
27-24 Y1_X2	Dither threshold value for x,y=2,1.
23-20	Dither threshold value for x,y=1,1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
Y1_X1	
19-16 Y1_X0	Dither threshold value for x,y=0,1.
15-12 Y0_X3	Dither threshold value for x,y=3,0.
11-8 Y0_X2	Dither threshold value for x,y=2,0.
7-4 Y0_X1	Dither threshold value for x,y=1,0.
3-0 Y0_X0	Dither threshold value for x,y=0,0.

46.5.1.7 Dither Table High (DisplayDitherTableHigh0)

Offset

Register	Offset
DisplayDitherTableHigh0	13A0h

Function

Contains the dither value threshold values.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Y3_X3				Y3_X2				Y3_X1				Y3_X0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Y2_X3				Y2_X2				Y2_X1				Y2_X0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

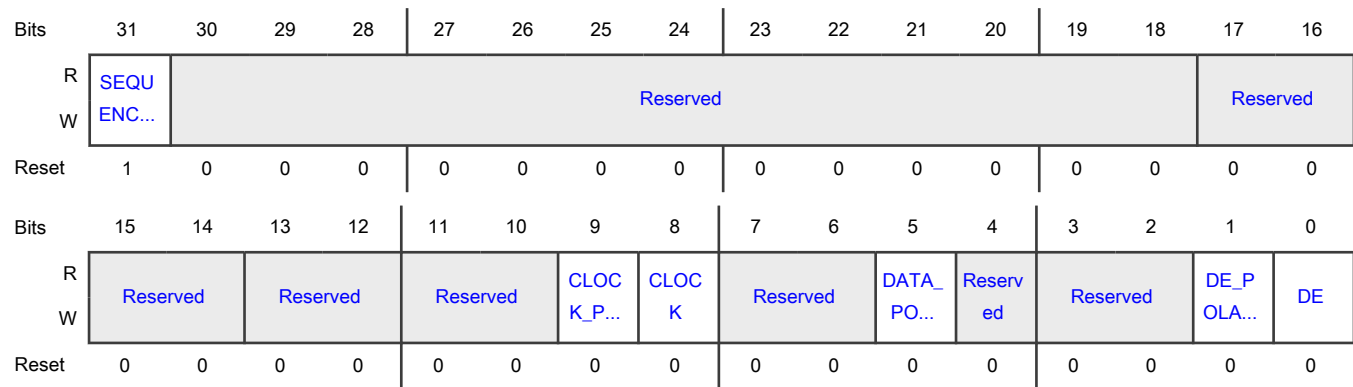
Field	Function
31-28 Y3_X3	Dither threshold value for x,y=3,3.
27-24 Y3_X2	Dither threshold value for x,y=2,3.
23-20 Y3_X1	Dither threshold value for x,y=1,3.
19-16 Y3_X0	Dither threshold value for x,y=0,3.
15-12 Y2_X3	Dither threshold value for x,y=3,2.
11-8 Y2_X2	Dither threshold value for x,y=2,2.
7-4 Y2_X1	Dither threshold value for x,y=1,2.
3-0 Y2_X0	Dither threshold value for x,y=0,2.

46.5.1.8 Panel Configuration (PanelConfig0)**Offset**

Register	Offset
PanelConfig0	13C0h

Function**NOTE**

This register is only accessible through the register interface.

Diagram**Fields**

Field	Function
31 SEQUENCING	Enable software or hardware panel sequencing. This bit is always set as 1'b1. 0b - Hardware 1b - Software
30-18 —	Reserved
17-16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11-10 —	Reserved
9 CLOCK_POLA RITY	Clock Polarity 0b - Positive 1b - Negative
8 CLOCK	Clock Enable/Disable 0b - Disabled 1b - Enabled
7-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
5 DATA_POLARITY	Data Polarity 0b - Positive 1b - Negative
4 —	Reserved
3-2 —	Reserved
1 DE_POLARITY	Data Enable Polarity 0b - Positive 1b - Negative
0 DE	Data Enable 0b - Disabled 1b - Enabled

46.5.1.9 Timing for Hardware Panel Sequencing (PanelTiming0)

Offset

Register	Offset
PanelTiming0	13E0h

Function

NOTE

This register is only accessible through the register interface.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POWER_DISABLE				BACKLIGHT_DISABLE				CLOCK_DISABLE				DATA_DISABLE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA_ENABLE				CLOCK_ENABLE				BACKLIGHT_ENABLE				POWER_ENABLE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 POWER_DISABLE	Number of VSYNCs to wait after power has been disabled.
27-24 BACKLIGHT_DISABLE	Number of VSYNCs to wait after backlight has been disabled.
23-20 CLOCK_DISABLE	Number of VSYNCs to wait after clock has been disabled.
19-16 DATA_DISABLE	Number of VSYNCs to wait after data has been disabled.
15-12 DATA_ENABLE	Number of VSYNCs to wait after data has been enabled.
11-8 CLOCK_ENABLE	Number of VSYNCs to wait after clock has been enabled.
7-4 BACKLIGHT_ENABLE	Number of VSYNCs to wait after backlight has been enabled.
3-0 POWER_ENABLE	Number of VSYNCs to wait after power has been enabled.

46.5.1.10 Horizontal Total and Display End Counters (HDisplay0)

Offset

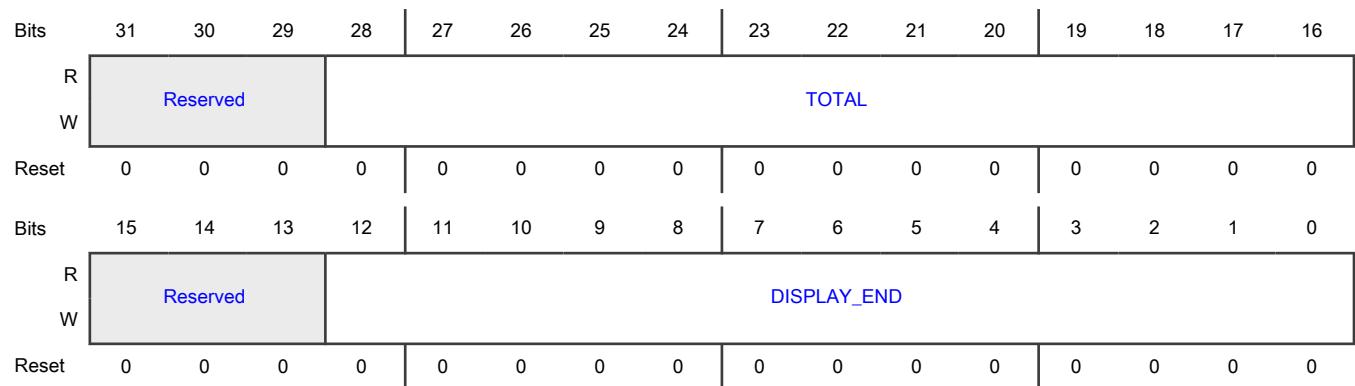
Register	Offset
HDisplay0	1400h

Function

NOTE

This register is only accessible through the register interface.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-16 TOTAL	Total number of horizontal pixels.
15-13 —	Reserved
12-0 DISPLAY_END	Number of visible horizontal pixels.

46.5.1.11 Horizontal Sync Counters (HSync0)

Offset

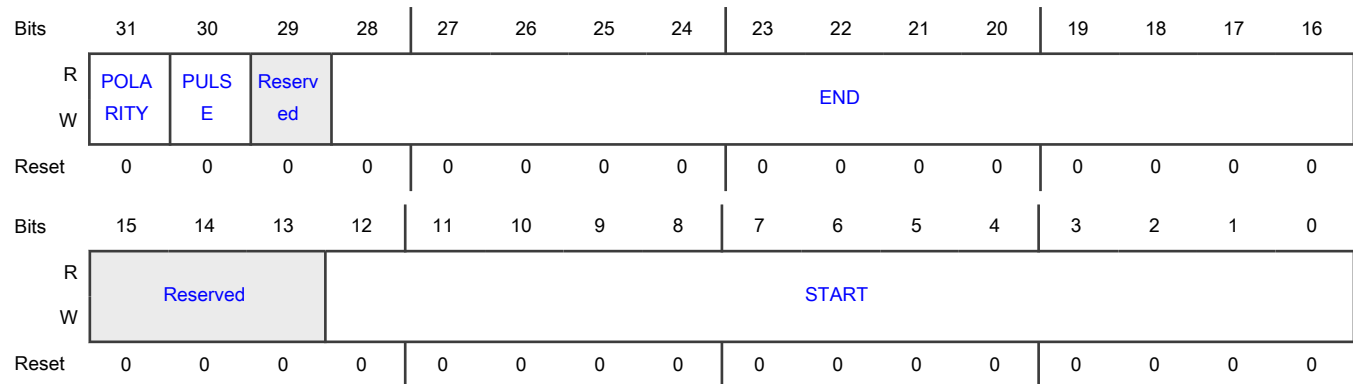
Register	Offset
HSync0	1420h

Function

NOTE

This register is only accessible through the register interface.

Diagram



Fields

Field	Function
31 POLARITY	Polarity of the horizontal sync pulse 0b - Positive 1b - Negative
30 PULSE	Horizontal sync pulse control. 0b - Disabled 1b - Enabled
29 —	Reserved
28-16 END	End of horizontal sync pulse.
15-13 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
12-0 START	Start of horizontal sync pulse.

46.5.1.12 Vertical Total and Display End Counters (VDisplay0)

Offset

Register	Offset
VDisplay0	1480h

Function

NOTE

This register is only accessible through the register interface.

Diagram

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

Fields

Field	Function
31-28 —	Reserved
27-16 TOTAL	Total number of vertical lines.
15-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-0 DISPLAY_END	Number of visible vertical lines.

46.5.1.13 Vertical Sync Counters (VSync0)

Offset

Register	Offset
VSync0	14A0h

Function

NOTE

This register is only accessible through the register interface.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POLA	PULSE	Reserved													
W	RITY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 POLARITY	Polarity of the vertical sync pulse. 0b - Positive 1b - Active-low
30 PULSE	Vertical sync pulse control. 0b - Disabled 1b - Enabled
29-28	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
27-16 END	End of the vertical sync pulse.
15-12 —	Reserved
11-0 START	Start of the vertical sync pulse.

46.5.1.14 Current x,y Location of Display Controller (DisplayCurrentLocation0)

Offset

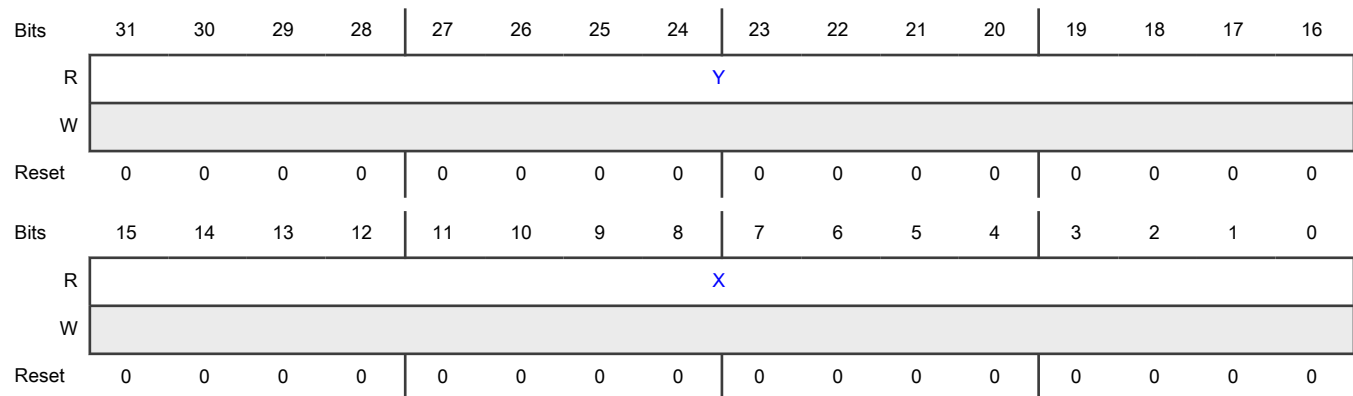
Register	Offset
DisplayCurrentLocation0	14C0h

Function

NOTE

This register is only accessible through the register interface.

Diagram



Fields

Field	Function
31-16	Current Y location.

Table continues on the next page...

Table continued from the previous page...

Field	Function
Y	
15-0 X	Current X location.

46.5.1.15 Index into Gamma Table (GammaIndex0)

Offset

Register	Offset
GammaIndex0	14E0h

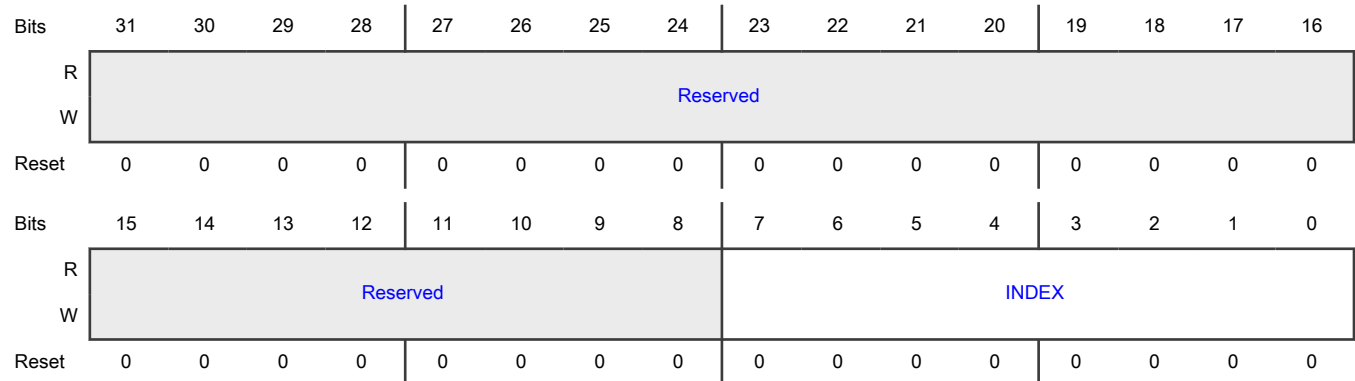
Function

See GammaData for more information.

NOTE

This register is accessible only through the register interface.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 INDEX	Index into Gamma Table.

46.5.1.16 Translation Values for the Gamma Table (GammaData0)

Offset

Register	Offset
GammaData0	1500h

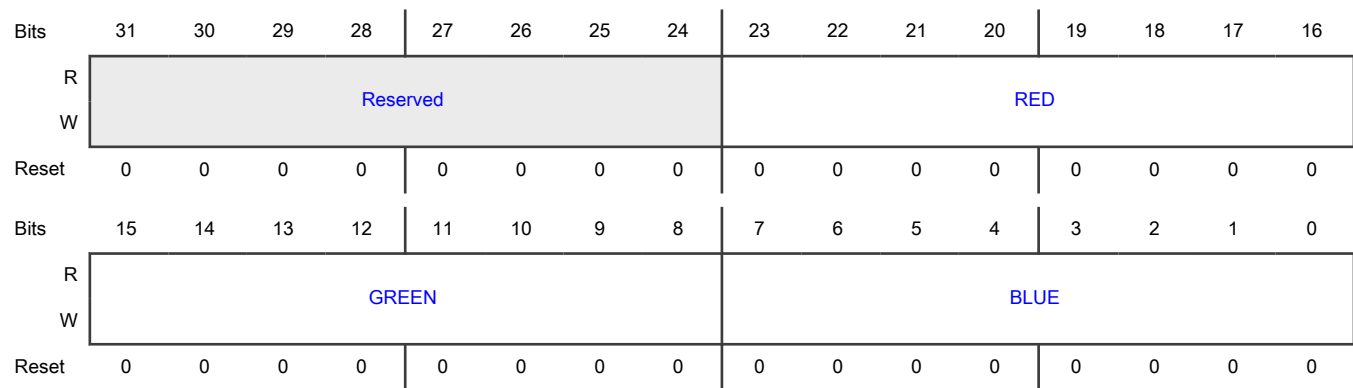
Function

When this register gets written, the data gets stored in the gamma table at the index specified by the GammaIndex register. After the register is written, the index gets incremented.

NOTE

This register is accessible only through the register interface.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 RED	Red translation value.
15-8 GREEN	Green translation value.
7-0 BLUE	Blue translation value.

46.5.1.17 Configuration for the Cursor (CursorConfig)

Offset

Register	Offset
CursorConfig	1520h

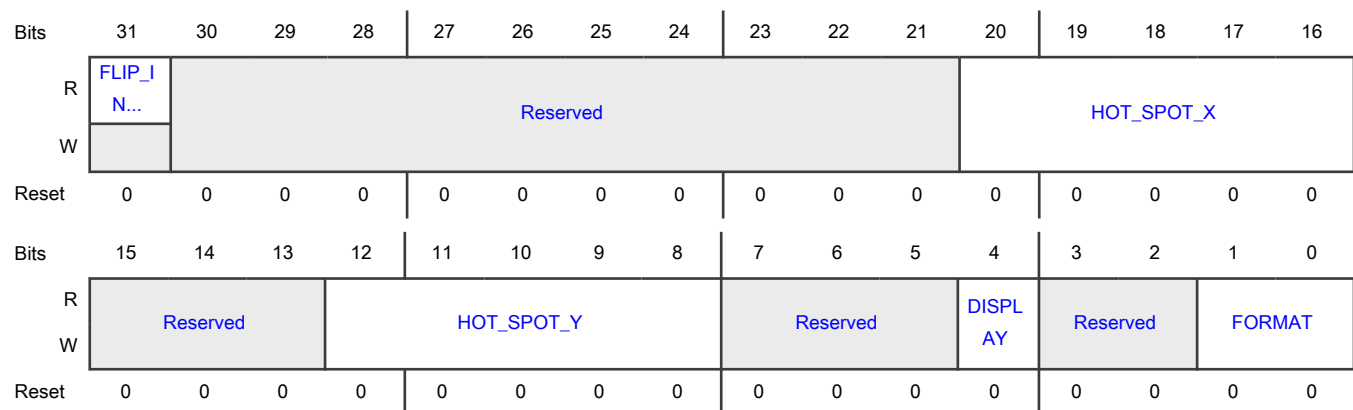
Function

Some fields are double buffered. Double-buffered values in this register cannot be read while a flip is in progress.

NOTE

This register is accessible only through the register interface.

Diagram



Fields

Field	Function
31	When the cursor address gets written to, this bit gets set to one.
FLIP_IN_PROG RESS	It will be reset to zero at the start of the next VBLANK of the owning display when the register gets copied into the working set. 0=NO; 1=YES;
30-21 —	Reserved
20-16 HOT_SPOT_X	Horizontal offset to cursor hotspot.
	NOTE This field is double buffered.
15-13 —	Reserved
12-8	Vertical offset to cursor hotspot.

Table continues on the next page...

Table continued from the previous page...

Field	Function
HOT_SPOT_Y	NOTE This field is double buffered.
7-5 —	Reserved
4 DISPLAY	Display Controller owning the cursor. Always 0 for this IP. 0b - DISPLAY0 1b - DISPLAY1
3-2 —	Reserved
1-0 FORMAT	Format of the cursor. NOTE This field is double buffered. 00b - DISABLED 01b - MASKED 10b - A8R8G8B8

46.5.1.18 Address of the Cursor Shape (CursorAddress)

Offset

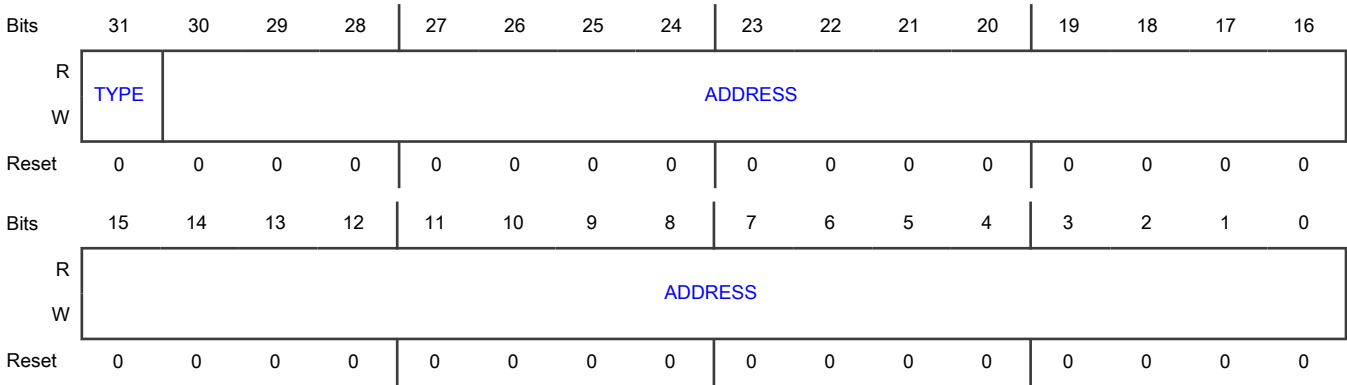
Register	Offset
CursorAddress	1530h

Function

NOTE

This register is double buffered and is accessible only through the register interface.

Diagram



Fields

Field	Function
31 TYPE	System Type 0b - System. 1b - Virtual system.
30-0 ADDRESS	ADDRESS

46.5.1.19 Location of the cursor on the owning display (CursorLocation)

Offset

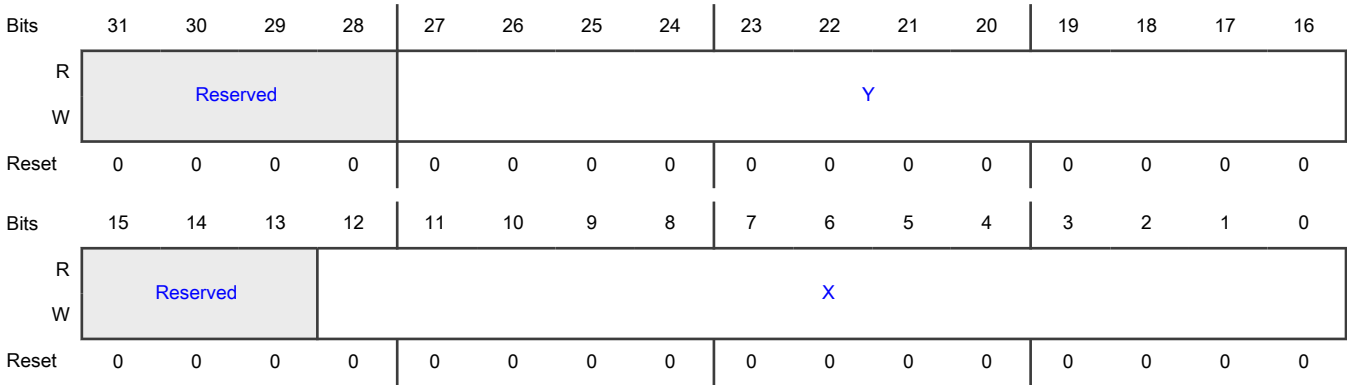
Register	Offset
CursorLocation	1540h

Function

NOTE

This register is double buffered and is accessible only through the register interface.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-16 Y	Y location of cursor's hotspot.
15-13 —	Reserved
12-0 X	X location of cursor's hotspot.

46.5.1.20 Background Color for Masked Cursors (CursorBackground)

Offset

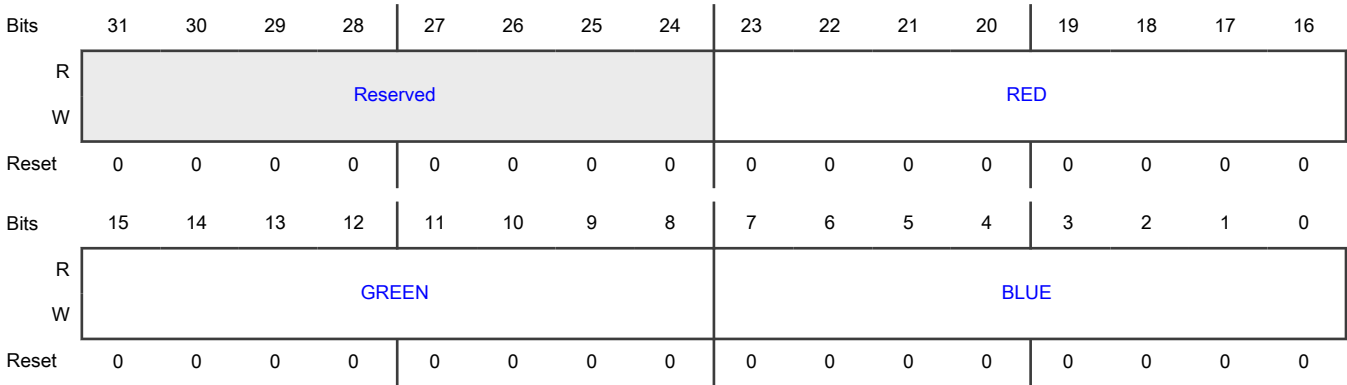
Register	Offset
CursorBackground	1550h

Function

NOTE

This register is double buffered and is accessible only through the register interface.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 RED	Red value
15-8 GREEN	Green value
7-0 BLUE	Blue value

46.5.1.21 Foreground Color for Masked Cursors (CursorForeground)

Offset

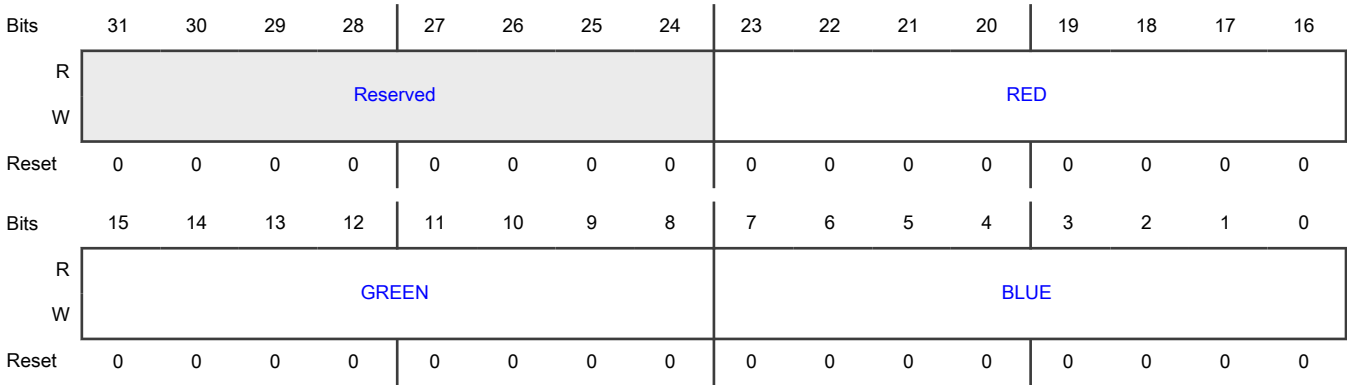
Register	Offset
CursorForeground	1560h

Function

NOTE

This register is double buffered and is accessible only through the register interface.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 RED	Red value
15-8 GREEN	Green value
7-0 BLUE	Blue value

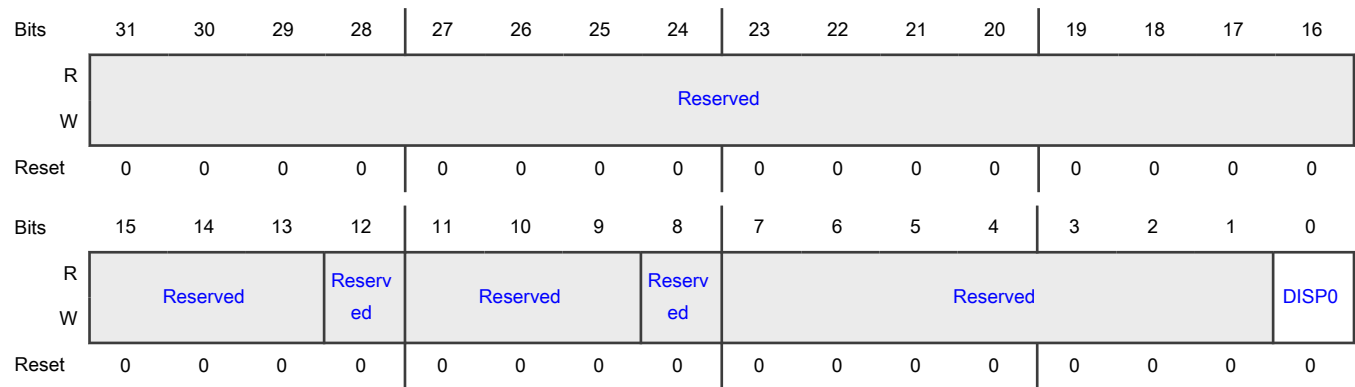
46.5.1.22 Display Interrupt (DisplayIntr)

Offset

Register	Offset
DisplayIntr	1600h

Function

This register will automatically clear after a read. The interrupt bit is set when the current frame buffer is used up. The interrupt signal will be pulled up only when the interrupt enable bit in register DisplayIntrEnable is enabled.

Diagram**Fields**

Field	Function
31-13 —	Reserved
12 —	Reserved
11-9 —	Reserved
8 —	Reserved
7-1 —	Reserved
0 DISP0	Display0 Interrupt

46.5.1.23 Interrupt Enable for Display_0 (and Display_1 if present) (DisplayIntrEnable)**Offset**

Register	Offset
DisplayIntrEnable	1610h

Function**NOTE**

Interrupt enable for register DisplayIntr.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															DISP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Fields

Field	Function
31-1 —	Reserved
0 DISP0	Display0 Interrupt Enable

46.5.1.24 DBI Configuration 0 (DbiConfig0)**Offset**

Register	Offset
DbiConfig0	1620h

Function

Configuration register for DBI output

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved	DBI_AC_TIME_UNIT						DBIX_ PO...	BUS_ OUT...	DBI_DATA_FORMAT			DBI_TYPE		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Fields

Field	Function
31-14 —	Reserved
13-12 —	Reserved
11-8 DBI_AC_TIME_ UNIT	Time Unit for AC Characteristics
7 DBIX_POLARIT Y	D/CX Pin polarity. 0b - Default 1b - Reverse
6 BUS_OUTPUT_ SEL	Output Bus Select 0b - DPI 1b - DBI
5-2 DBI_DATA_FO RMA_T	DBI Interface Data Format. Refer to DBI specification section on Interface Color Coding for detail. • 0=D8R3G3B2 • 1=D8R4G4B4 • 2=D8R5G6B5 • 3=D8R6G6B6 • 4=D8R8G8B8 • 5=D9R6G6B6 • 6=D16R3G3B2 • 7=D16R4G4B4 • 8=D16R5G6B5 • 9=D16_R6_G6_B6_OP1 • A=D16_R6_G6_B6_OP2 • B=D16_R8_G8_B8_OP1 • C=D16_R8_G8_B8_OP2
1-0 DBI_TYPE	DBI Type Select 00b - TYPE_AFIXED_E 01b - TYPE_ACLOCK_E 10b - TYPE_B 11b - Reserved

46.5.1.25 Reset DBI Interface to Idle State (DbilfReset0)

Offset

Register	Offset
DbilfReset0	1640h

Function

Reset DBI interface to idle state. WRITE ONLY.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 DBI_IF_LEVEL_RESET	Reset DBI interface to idle state 1=RESET;

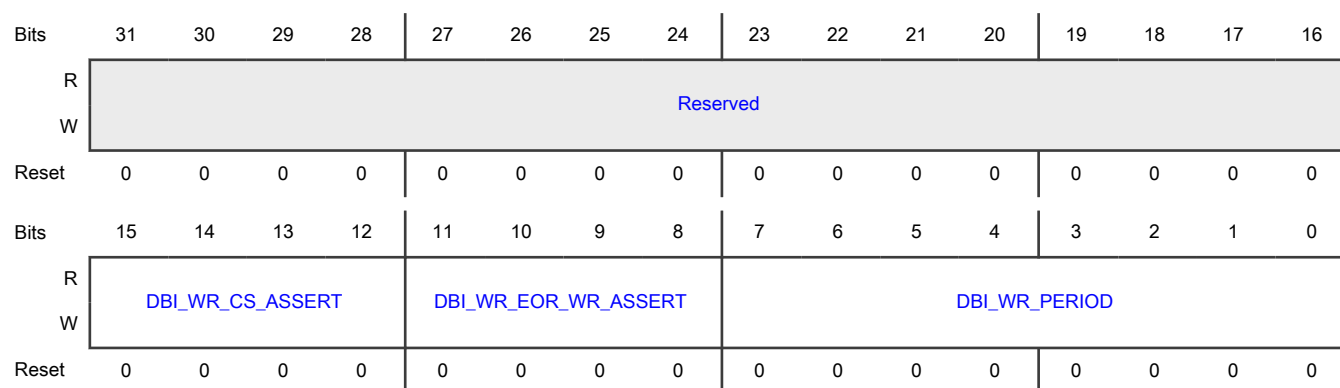
46.5.1.26 DBI Write Characteristics 1 (DbiWrChar10)

Offset

Register	Offset
DbiWrChar10	1660h

Function

DBI write AC characteristics definition register 1

Diagram**Fields**

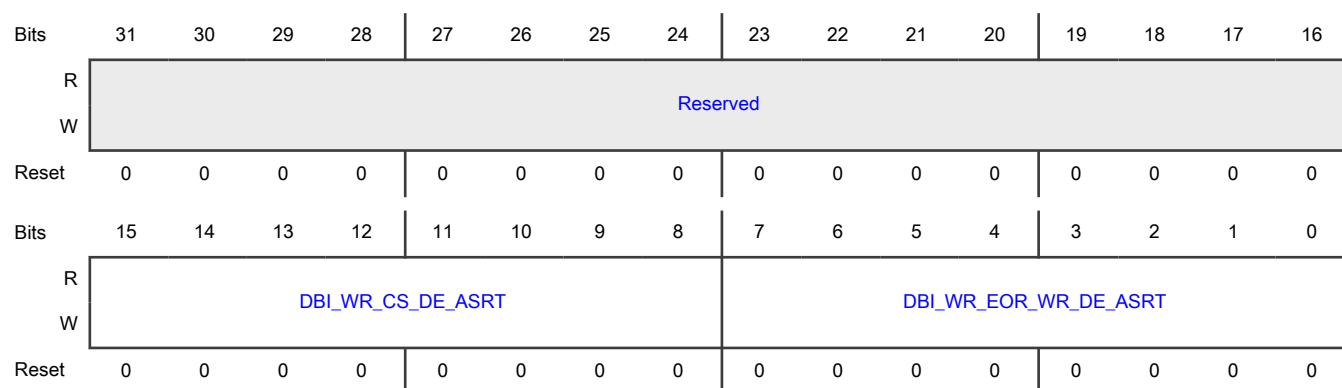
Field	Function
31-16 —	Reserved
15-12 DBI_WR_CS_ASSERT	Cycle number=Setting*(DbiAcTimeUnit+1). 0: When Type A Fixed E mode: Time to assert CSX. 1: When Type A Clock E mode: Not used. 2: When Yype B mode: Time to assert CSX.
11-8 DBI_WR_EOR_WR_ASSERT	Cycle number=Setting*(DbiAcTimeUnit+1). 0: When Type A Fixed E mode: Not used. 1: When Type A Clock E mode: Time to assert E. 2: When Type B mode: Time to assert WRX.
7-0 DBI_WR_PERIOD	Single Write Period Duration Cycle number=Setting*(DbiAcTimeUnit+1). This field must be no less than 3.

46.5.1.27 DBI Write Characteristics 2 (DbiWrChar20)**Offset**

Register	Offset
DbiWrChar20	1680h

Function

DBI write AC characteristics definition register 2

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-8 DBI_WR_CS_D E_ASRT	Cycle number=Setting*(DbiAcTimeUnit+1). 0: When type A fixed E mode: Time to de-assert CSX. 1: When type A clock E mode: Not used. 2: When type B mode: Time to de-assert CSX.
7-0 DBI_WR_EOR_ WR_DE_ASRT	Cycle number=Setting*(DbiAcTimeUnit+1). 0: When Type A Fixed E mode: Not used. 1: When type A Clock E mode: Time to de-assert E. 2: When Type B mode: Time to de-assert WRX.

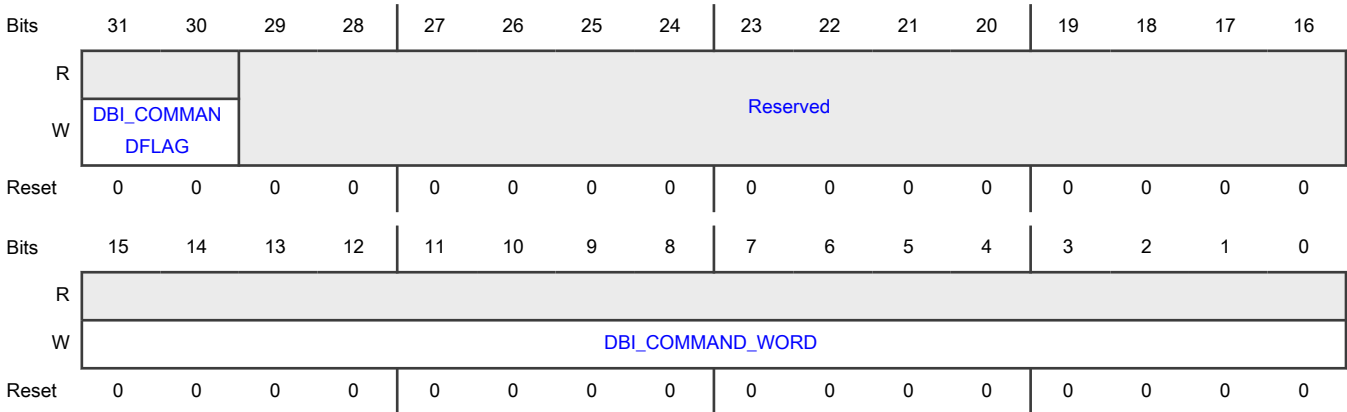
46.5.1.28 DBI Command In/Out Port (DbiCmd0)**Offset**

Register	Offset
DbiCmd0	16A0h

Function

DBI Command in/out port. Writes to this register will send command/data to the DBI port. WRITE ONLY.

Diagram



Fields

Field	Function
31-30 DBI_COMMAN DFLAG	DBI Command Flag DBI command flag. 00b - ADDRESS 01b - WRITE_MEM_START 10b - PARAMETER_OR_DATA 11b - Not used
29-16 —	Reserved
15-0 DBI_COMMAN D_WORD	DBI Command Word The type of data contained in this word is specified using DBI_COMMANDFLAG[bits 31:30]. DBI_COMMAND_WORD[8] = 0 indicates DBI_COMMAND_WORD[7:0] is COMMAND WORD. DBI_COMMAND_WORD[8] = 1 indicates DBI_COMMAND_WORD[7:0] is DATA WORD.

46.5.1.29 DPI Configuration 0 (DpiConfig0)

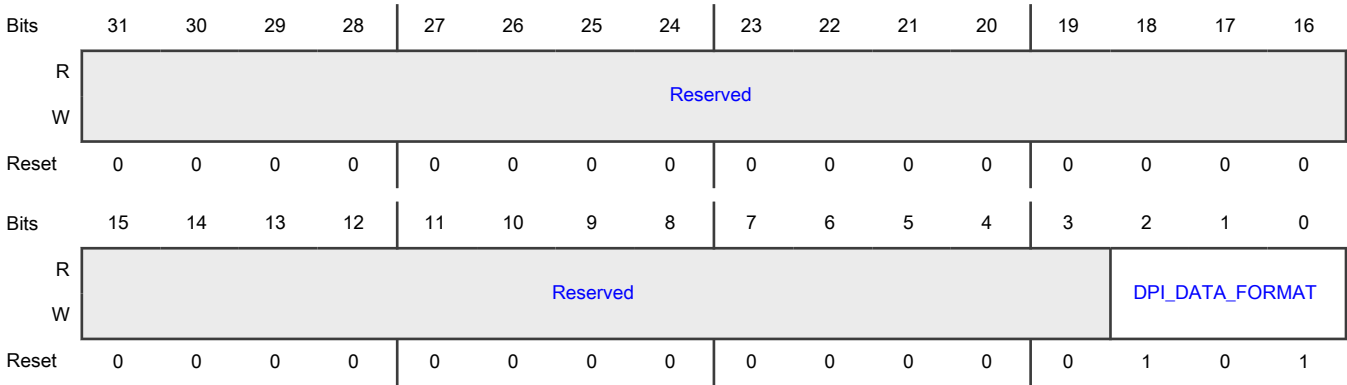
Offset

Register	Offset
DpiConfig0	16C0h

Function

The configuration register for DPI output

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 DPI_DATA_FORMAT	DPI Interface Data Format Refer to DPI specification section for Interface Color Coding' for detail. 000b - D16CFG1 001b - D16CFG2 010b - D16CFG3 011b - D18CFG1 100b - D18CFG2 101b - D24 110b-111b - -

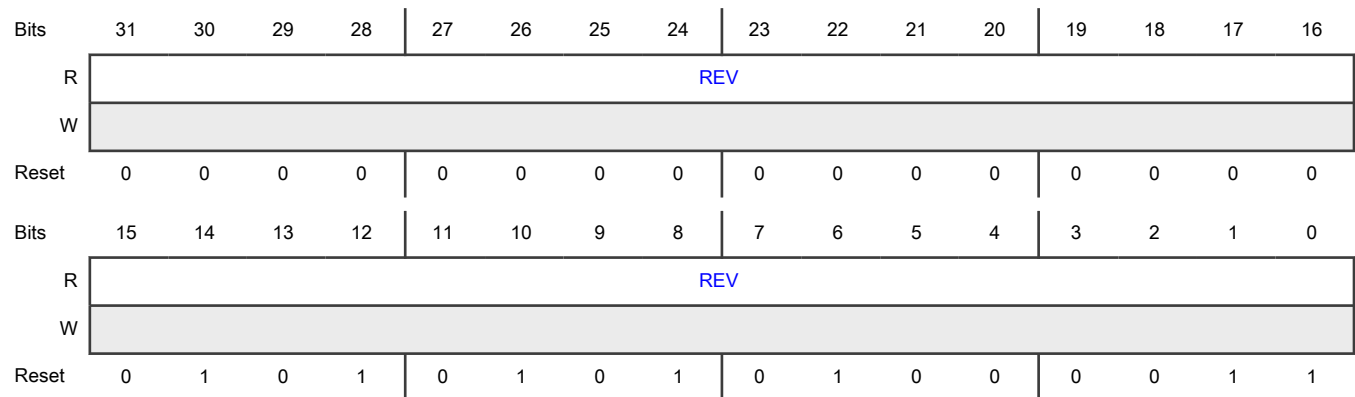
46.5.1.30 Revision for the LCDIF Peripheral in BCD (DcChipRev)

Offset

Register	Offset
DcChipRev	16F0h

Function

Shows the revision for the LCDIF peripheral in BCD. This register has no set reset value. It varies with the implementation. READ ONLY.

Diagram**Fields**

Field	Function
31-0	Revision
REV	Revision

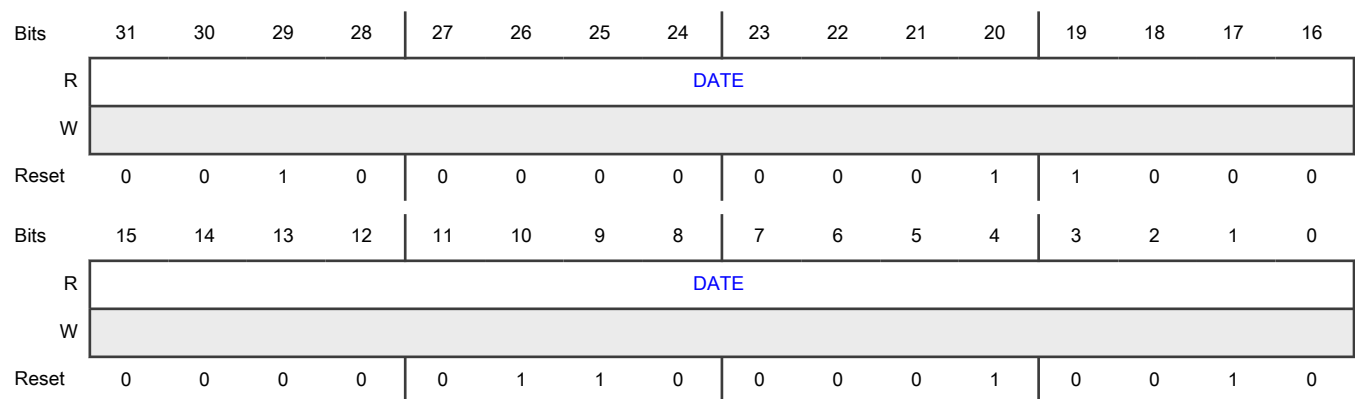
46.5.1.31 Shows the release date for the IP in YYYYMMDD (year, month) (DcChipDate)

Offset

Register	Offset
DcChipDate	1700h

Function

This register has no set reset value. It varies with the implementation. READ ONLY.

Diagram

Fields

Field	Function
31-0 DATE	Date

46.5.1.32 Patch Revision (DcChipPatchRev)**Offset**

Register	Offset
DcChipPatchRev	1720h

Function

Patch revision level for the LCDIF peripheral. This is a READ ONLY register.

Diagram

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

Fields

Field	Function
31-0 PATCH_REV	Patch Revision

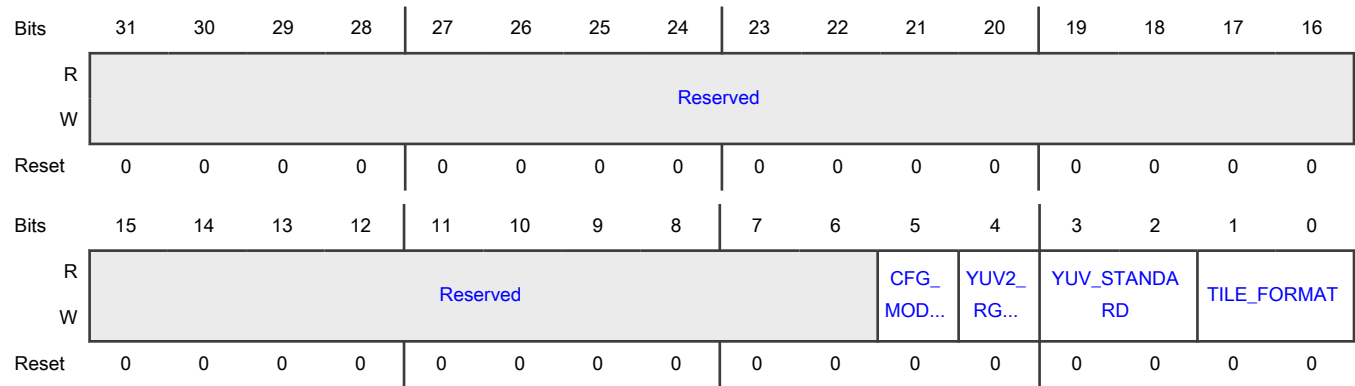
46.5.1.33 Tile Input Configuration (DcTileInCfg0)**Offset**

Register	Offset
DcTileInCfg0	1740h

Function

Some fields are double buffered.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 CFG_MODE_EN	Configuration Mode Enable. 0b - Disabled 1b - Enabled
4 YUV2_RGB_EN	YUV2RGB Module Enable NOTE This field is double buffered. 0b - Disabled 1b - Enabled
3-2 YUV_STANDAR	YUV Standard Select NOTE This field is double buffered. 00b - BT601 01b - BT709
1-0 TILE_FORMAT	Tile Format Tile input data format 0 means non-tile input. NOTE This field is double buffered. 0=NONE; 1=ARGB8888; 2=YUY2; 3=NV12;

46.5.1.34 UV Frame Buffer Address when Tile Input (DcTileUvFrameBufferAdr0)

Offset

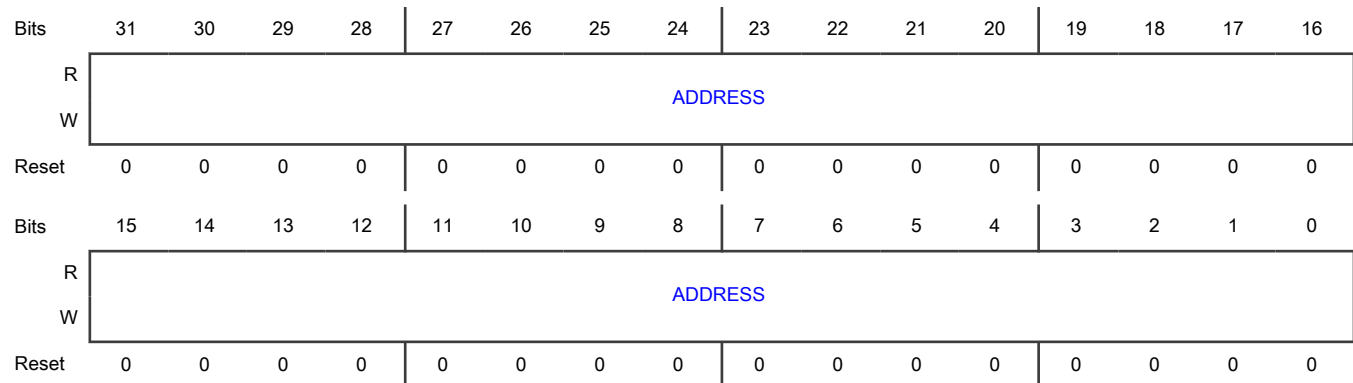
Register	Offset
DcTileUvFrameBufferAdr0	1760h

Function

NOTE

This register is double buffered.

Diagram



Fields

Field	Function
31-0 ADDRESS	UV Frame Buffer Address when Tile Input

46.5.1.35 UV Frame Buffer Stride when Tile Input (DcTileUvFrameBufferStr0)

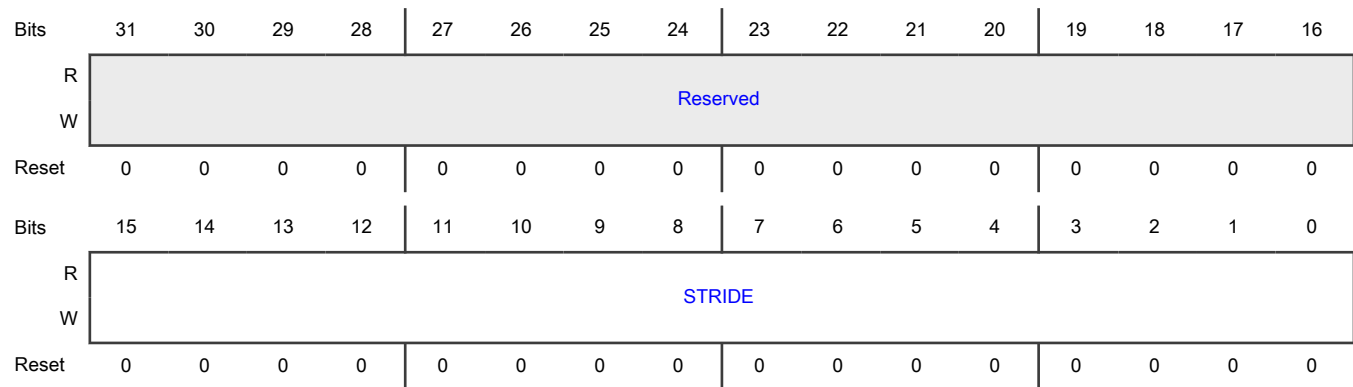
Offset

Register	Offset
DcTileUvFrameBufferStr0	1780h

Function

NOTE

This register is double buffered.

Diagram**Fields**

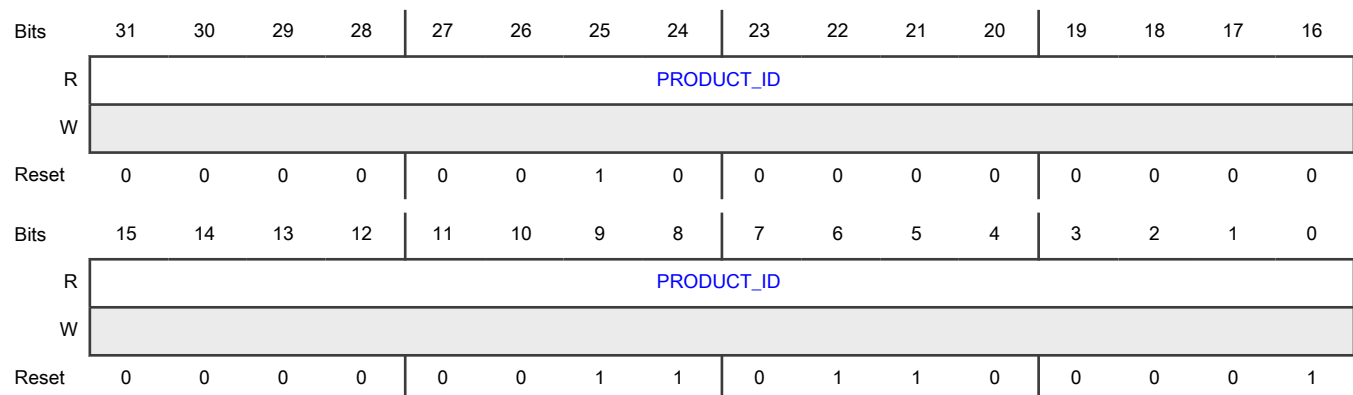
Field	Function
31-16 —	Reserved
15-0 STRIDE	UV Frame Buffer Stride when Tile Input

46.5.1.36 Product ID (DcProductId)**Offset**

Register	Offset
DcProductId	17B0h

Function

Contains Product Identification.

Diagram

Fields

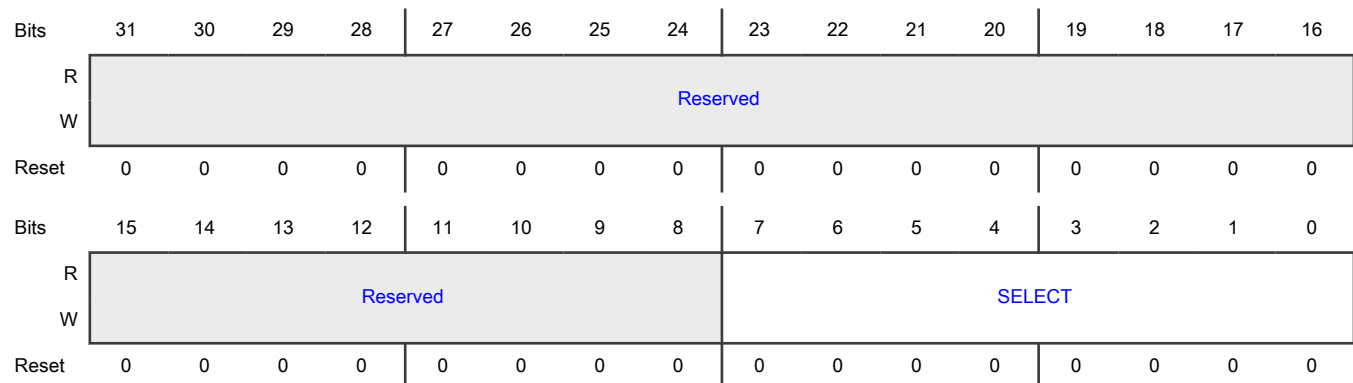
Field	Function
31-0 PRODUCT_ID	Product ID

46.5.1.37 Debug Counter Select (DebugCounterSelect0)**Offset**

Register	Offset
DebugCounterSelect0	1820h

Function

Contains Debug counter select value.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 SELECT	Select Write a value to this field to pick up from 0~255 counters. Then the counter will be in DebugCounterValue. • 00=TOTAL_AXI_RD_REQ_CNT • 01=TOTAL_AXI_RD_LAST_CNT • 02=TOTAL_AXI_REQ_BURST_CNT • 03=TOTAL_AXI_RD_BURST_CNT

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 04=TOTAL_PIXEL_CNT • 05=TOTAL_FRAME_CNT • 06=TOTAL_INPUT_DBI_CMD_CNT • 07=TOTAL_OUTPUT_DBI_CMD_CNT • 08=DEBUG_SIGNALS0 • FF=RESET_ALL_DEBUG_COUNTERS

46.5.1.38 Debug Counter Value (DebugCounterValue0)

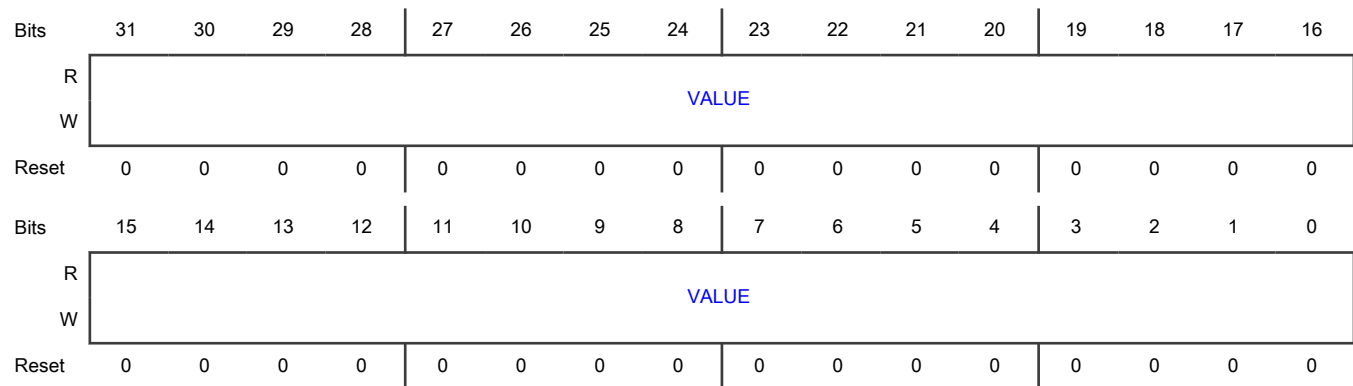
Offset

Register	Offset
DebugCounterValue0	1840h

Function

Debug counter value as specified in DebugCounterSelect.

Diagram



Fields

Field	Function
31-0 VALUE	Selected Debug Counter Value

Chapter 47

Vector Graphics Processing Unit (GPU)

47.1 Chip-specific GPU information

Table 482. Reference links to related information

Topic	Related module	Reference
Full description	GPU	GPU
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

47.2 Overview

The vector graphics processing unit is based on the Vivante GCNanoLite-V IP core. The graphics processing unit (GPU) provides high performance, high quality graphics, low power consumption, and the smallest silicon footprint for its class. The dynamic power consumption is minimized by extensive use of multi-level hierarchical clock gating. An optimized software stack, complete software development tools, and a growing application ecosystem are supported by a robust graphics pipeline. The GPU supports Vivante vgLite which is ultralow memory footprint API.

47.3 Features

- Pixel throughput of 1 pixel/clock
- Hardware tessellation
- 8x CAA (Coverage Anti Aliasing)
- Configurable Coordinate Systems and transformations (3x3 matrix)
- Blending including 8 Porter Duff blending modes
- Support for index and Color lookup table
- Rotation to any angle
- Odd/even and non-zero fill rules
- Linear and curved paint paths (linear gradient)
- Solid and linear pattern
- Pixel formats for image interpolation – 7 pixel formats for image interpolation
- Support for A4/A8 input

47.3.1 GPU Host and Memory Interface Features

- 32-bit AHB interface
- 64-bit AXI interface for external memory access
- No Code and data memory location restrictions; arbitrary memory reads and writes
- 32-bits physical address
- Memory controller supports 8, 16, 32, 64 byte read and write requests

- Semaphore lock with CPU

47.3.2 GCNanoLiteV Hardware Features

Table 483. Graphics Hardware Features

Feature	Description
Coordinate systems and transformations	Image drawing uses a 3x3 perspective transformation matrix
Blending	8 Porter Duff blending modes
Rotation	To any angle
Fill rules	Odd/even and non-zero
Paint paths	Linear and curved paint paths
Paints	Solid and linear pattern
Pixel formats for image interpolation	14 pixel formats for image interpolation, including A4/A8

47.3.3 Format Support

Table 484. Graphics Formats Available

Feature	GPU Support			
Image formats	The graphics engine supports these formats for source image and destination render targets.			
	Bit Depth	Format	Supported for Source IMAGE	Supported for Destination
	1	INDEX1	Yes	
	2	INDEX2	Yes	
	4	INDEX4	Yes	
	8	INDEX8	Yes	
	4	A4	Yes	
	8	A8	Yes	Yes
	8	L8	Yes	Yes
	8	A2R2G2B2	Yes	Yes
	16	R5G6B5	Yes	Yes
	16	A1R5G5B5	Yes	Yes
	16	YUY2/YUYV	Yes	
	16	A4R4G4B4	Yes	Yes
	32	A8R8G8B8	Yes	Yes
	32	X8R8G8B8	Yes	Yes
Color component swizzle support	Supported color component swizzles are:			

Table continues on the next page...

Table 484. Graphics Formats Available (continued)

	• SWIZZLE_ABGR • SWIZZLE_ARGB • SWIZZLE_BGRA • SWIZZLE_RGBA

47.4 Block Diagram

A block diagram and a description of the functional blocks of the complete graphics pipeline is given below:

Host Interface	Allows the GPU to communicate with external memory and the CPU through the AXI or AHB bus. In this block data crosses clock domain boundaries.
Memory Controller	Internal memory management unit that controls the block-to-host memory request interface.
Graphics Pipeline Front End	Inserts high level primitives and commands into the graphics pipeline.
Tessellation Engine	Transforms vertices and control points. Tessellates lines, quadratic and cubic Bezier curves.
Vector Graphics Engine	Rasterizer that converts primitives to pixels.
Imaging Engine	Paint and image generator that colors each pixel.
Pixel Engine	Renderer that combines different sources into the final pixel value.

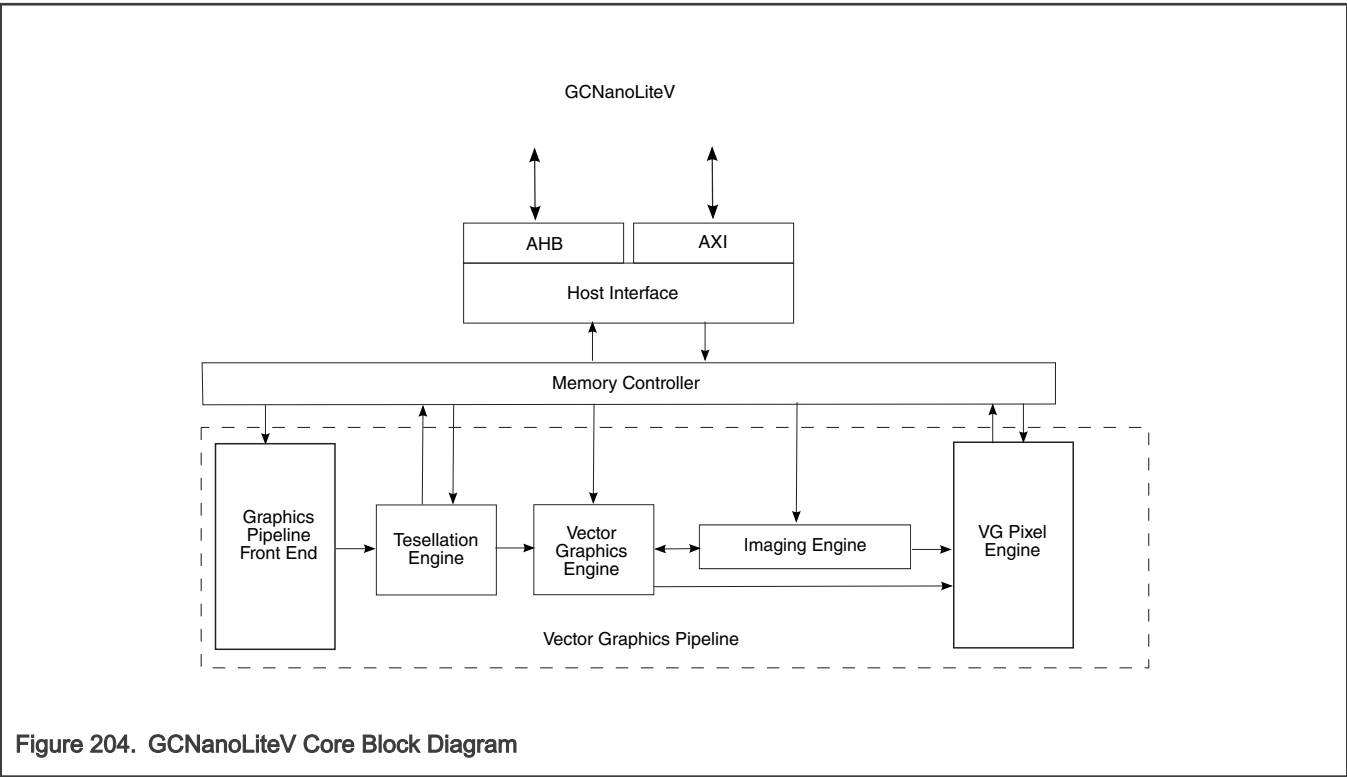


Figure 204. GCNanoLiteV Core Block Diagram

47.5 GPU Power Management Features

- Low power CMOS technology compatible
- Automatic clock gating of flip flops and RAM's
- Global clock gating of unused macro blocks
- Active(ON), Idle, Standby and Sleep(OFF) Programmable Power Modes

47.6 Software API

The Vivante VGLite Graphics API is designed specifically for use with GCNanoLiteV vector graphics processing unit. It provides mechanisms for hardware accelerated two-dimensional vector and raster graphics, thereby allowing the user to implement customized applications and drivers for their hardware accelerated graphical user interface.

47.6.1 Vivante VGLite Graphics API

Vivante VGLite Graphics API (Application Programming Interface) is designed to support menu driven user interfaces optimized for a system's overall resource requirements. Its goal is to provide maximum performance, while keeping the memory footprint to a minimum.

Vivante VGLite Graphics API document will be available as a part of the RT500 API Reference Manual.

47.6.2 Organization of VGLite Graphics API

The VGLite API is partitioned to provide controls for three main areas of functionality:

- Initialization - for hardware and software initialization
- Blit - for raster rendering
- Draw - for draw operations

Hardware supported Blit and/or Draw capabilities available for software control through the API include:

Table 485. Blit and Draw Controls in VGLite API

Capability	GCNanoLiteV GPU with VGLite API Support
Fill	Non-zero or Even-Odd
Path coordinate formats	S8, S16, S32, FP32
HW Anti-alias levels	Low, Medium, High
Blending	SrcOver, DstOver, SrcIn, DstIn, Screen, Multiply, Additive, Subtractive
Data Formats	INDEX1/2/4/8, ARGB2/4/8, RGB565, A4/8, L8 and BRGA variants
Path definitions	bounding box, path quality, coordinate format, buffer pointer, path length, path data

Chapter 48

MIPI DSI (MIPI_DSI) Interface

48.1 Chip-specific MIPI-DSI information

Table 486. Reference links to related information

Topic	Related module	Reference
Full description	MIPI-DSI	MIPI-DSI
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

RT500 does not support ULPS mode.

Chip-specific Reset and Initialization procedure for the DSI Host Controller:

- Assert all resets through PRSTCTL0[MIPI_DSI_PHY].
- Setup DSI related clocks through chip-specific registers provided for this purpose.
- Setup MIPI DSI application by programming all DSI Host Controller and DSI DPHY required parameters (refer to MIPI DSI Host Memory Map/Register for details).
- Deassert all remaining resets through PRSTCTL0[MIPI_DSI_PHY] .

DSI Host Controller should be ready for use.

Chip-specific procedure required for MIPI-DSI DPHY in Deep Sleep mode:

- Set display in Deep Sleep mode
- Disable MIPIDPHY before entering MCU Deep Sleep (DPHY Power Down Register PD_DPHY[PD_DPHY])
- Configure PDSLEEPCFG with PD_EXCLUDE parameter to exclude or include MIPI-DSI from deep sleep
- Enter Deep Sleep
- Wake up
- Enable MIPI-DPHY after waking up from Deep Sleep
- Exit display from Deep Sleep mode

48.1.1 Module instances

This device has one instance of the MIPI-DSI module.

48.2 Overview

MIPI_DSI_HOST controller is a flexible, high-performance, and easy-to-use digital core that implements all protocol functions defined in the MIPI_DSI_HOST Specification. The MIPI_DSI_HOST controller provides an interface that allows communication with MIPI_DSI_HOST-compliant peripherals.

The MIPI_DSI_HOST D-PHY is a high frequency, low power, low-cost, source-synchronous, physical layer supporting the MIPI Alliance standard for D-PHY.

NOTE

MIPI_DSI_HOST in this chip does not have an embedded PLL, hence disregard any PLL related contents in this chapter.

NOTE

This module has an associated supply switch with the module operation release based on LPOSC clock. The module reset assert and de-assert operations require LPOSC active (2 clock pulses minimum). LPOSC can be disabled between module reset operations.

48.2.1 Features

Key features of the MIPI_DSI_HOST Controller Core include:

- Implements all three DSI Layers (Pixel to Byte packing, Low Level Protocol, Lane Management)
- Support for Command and Video Modes
- Host Version
- Scalable data lane support, 1 to 2 Data Lanes
 - Optional bidirectional support on lane 0
- Supports High Speed and Low Power operation
- Support for all DSI data types and formats
- Virtual Channel support
- Full Low-Level Protocol Error and Contention detection and reporting
- Supports continuous and non-continuous Clock Lane operation
- Supports multiple packets per transmission
- Support for all three Video Mode packet sequences
 - Non-Burst Mode with Sync Pulses
 - Non-Burst Mode with Sync Events
 - Burst mode
- Support for bus turnaround signaling
- Flexible packet based user interface
 - APB interface option (status and control)
 - Display Pixel Interface Core (DPI-2) option
- Supports PHY Protocol Interface (PPI) compatible MIPI D-PHYs
- MIPI Alliance Specification for Display Serial Interface Version 1.1 compliant

The following are the key features of the MIPI_DSI_HOST D-PHY:

- 1 Clock lane
- 2 data lanes
- Supports MIPI Standard for D-PHY
- Supports both high speed and low power modes
- High Speed Serializers and Deserializers

48.3 DSI Host Controller Core

The figure below illustrates the DSI Host Controller Core structure. The DSI Host Controller Core operates on the host (transmit) side of a DSI link.

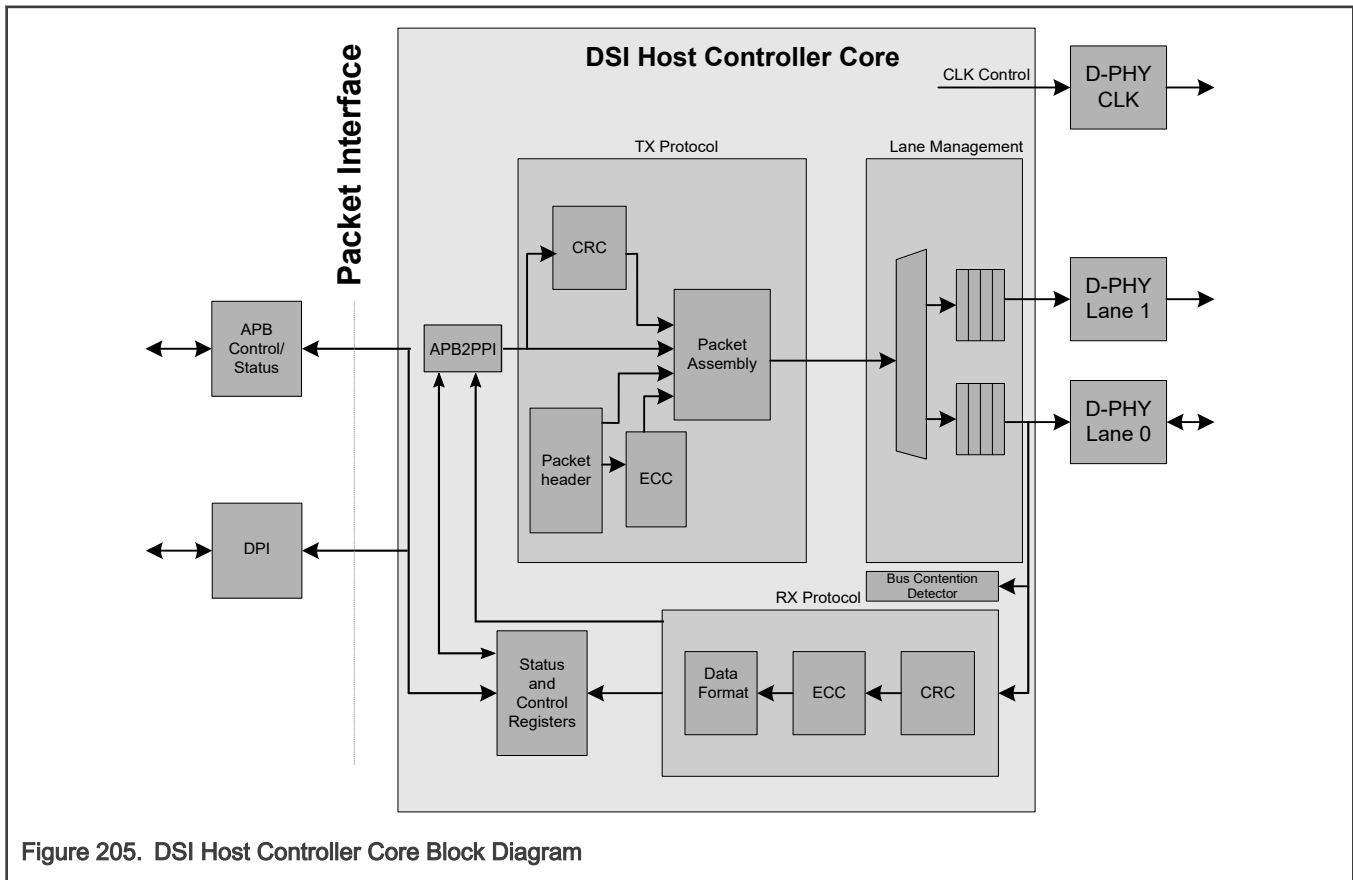


Figure 205. DSI Host Controller Core Block Diagram

The DSI Host Controller Core implements all three layers defined by the DSI Specification: Pixel to Byte Packing in the Application layer, Low Level Protocol, and Lane Management. The DSI Host Controller Core sends and receives DSI commands via the Packet Interface. The Packet Interface can be connected to a DPI translator or to an APB to PPI gasket.

The D-PHY interface of the DSI Host Controller Core supports up to four PHY Protocol Interface (PPI) compatible MIPI D-PHYs.

The Packet Interface is an easy-to-use data interface that accepts commands and data, and sends it over the DSI link. It supports 1 to 4 virtual channels, and the use of 1-2 D-PHY lanes. The DSI Host Controller Core takes care of all packet formatting details and transmission over the MIPI bus.

The DPI Translator connects to the DSI Host Controller Core via the Packet Interface. DPI masters may connect directly to the DPI Translator to send commands across the DSI link.

Sending commands and receiving information through the DSI link is also possible through the APB to PPI gasket. See MIPI_DSI_HOST APB PKT IF Memory Map/Register section for details about the DSI registers which allow the APB to PPI setup and communication.

48.3.1 D-PHY Interface

The DSI Host Controller Core D-PHY Interface connects directly to MIPI PPI compliant D-PHYs. The DSI Host Controller Core uses this interface to transmit data, send Escape sequences, receive and transmit triggers, and detect and report D-PHY error conditions.

48.3.1.1 High-Speed Transmit Interface

The High-Speed Transmit interface is used to transmit High-Speed data across the MIPI interface. All signals are synchronous to the PPI TX byte clock, TxByteClkHS, and are PPI compliant.

48.3.1.2 Escape Mode Transmit Interface

The DSI Host Controller Core uses the Escape Mode Transmit Interface to generate Escape sequences on the MIPI Interface via the D-PHY. Escape sequences are used to change the DPHY transmit mode, go into low-power mode, or initiate a bus turnaround. The Controller generates Escape sequence requests when necessary to implement the DSI protocol. These signals are synchronous to the `clk_esc`.

48.3.2 Packet Interface

The DSI Host Controller Core Packet Interface consists of Transmit, Receive, and Control and Status sections. Through these interfaces, the user application can take complete control of the DSI interface, sending all video timing, sending DSI commands, receiving DSI reads, monitoring the status of the interface and responding to error reporting.

User application may have access to DSI Host Controller Core Packet Interface through `DSI_HOST_APB_PKT_IF` registers, `DSI_HOST_TX_PAYLOAD` and `DSI_HOST_PKT_CONTROL`. For details refer to the section [MIPI DSI Host register descriptions](#)

48.3.2.1 Transmit Ports

The Transmit Packet Interface is the mechanism with which the user creates packets to send over the MIPI Interface.

For Long Packets, the user provides the Virtual Channel (VC) number, Data Type (DT), and Word Count (WC) to the controller. The Controller then creates a packet header and pulls the packet data from the Packet Interface and out to the D-PHY to transmit.

For Short Packets, the user provides the Virtual Channel (VC) number, Data Type (DT), and required parameters (if any) to the controller. The Controller then creates the short packet and sends it to the D-PHY to transmit. This interface enables the user application to transmit and receive any type of DSI packet.

The Packet Interface Transmit Interface ports are listed in the table below. Signals are synchronous to the Tx Byte clock.

NOTE

Only a subset of the Packet Interface Signals are accessible by user application through `DSI_HOST_APB_PKT_IFx` (`DSI_HOST_TX_PAYLOAD` and `DSI_HOST_PKT_CONTROL`) registers. For `reset_byte_n` and `reset_esc_n` signals, refer to chip-specific MIPI_DSI_HOST information. For `tx_payload`, `tx_cmd_data_type`, `tx_cmd_vc`, and `tx_cmd_byte_count` signals, refer to `DSI_HOST_TX_PAYLOAD` register.

Table 487. DSI Host Controller Core Transmit Packet Interface

PORT	TYPE	DESCRIPTION
<code>clk_byte</code>	Input	Byte clock input. The D-PHY PPI interface, the <code>tx_cmd</code> and <code>tx_payload</code> interfaces are synchronous to <code>clk_byte</code> . This clock can be independent and unrelated to <code>clk_esc</code> .
<code>clk_esc</code>	Input	The low-speed D-PHY Escape Mode clock, which becomes <code>TxCkEsc</code> on the D-PHY interface. This clock can be independent and unrelated to <code>clk_byte</code> .
<code>reset_byte_n</code>	Input	Asynchronous reset, active low. This reset applies to all logic in the <code>clk_byte</code> clock domain.
<code>reset_esc_n</code>	Input	Asynchronous reset, active low. This reset applies to all logic in the <code>clk_esc</code> clock domain.
<code>tx_payload[31:0]</code>	Input	Packet data input.
<code>tx_payload_en</code>	Output	Packet data read enable. This active high signal indicates that the controller requires a valid packet during the next <code>clk_byte</code> period.

Table continues on the next page...

Table 487. DSI Host Controller Core Transmit Packet Interface (continued)

PORT	TYPE	DESCRIPTION
tx_payload_en_last	Output	Last packet read enable, active high signals last cycle of tx_payload_en.
tx_cmd_data_type[5:0]	Input	Transmit packet DSI data type. It is written into the command buffer when tx_cmd_ack is asserted high.
tx_cmd_vc[1:0]	Input	Transmit packet command virtual channel. It is written into the command buffer when tx_cmd_ack is asserted high.
tx_cmd_byte_count[15:0]	Input	Transmit packet payload byte count. It is written into the command buffer when tx_cmd_ack is asserted high. For DSI Long packet types, tx_cmd_byte_count defines the number of bytes of packet data to pull from the tx_payload port. For DSI Short packets, the format of tx_cmd_byte_count contains any optional parameters. If the SDI Short packet type does not have any parameters, it is recommended to set tx_cmd_byte_count to all 0s.
tx_cmd_req	Input	Transmit packet command request. This active high signal informs the controller that the packet command is valid. The packet command consists of the ports tx_cmd_data_type, tx_cmd_vc, and tx_cmd_byte_count. The controller will assert tx_cmd_ack when it accepts the command, after which, the user should either update port values for the next transmit packet command or deassert tx_cmd_req.
tx_cmd_ack	Output	Transmit packet command request acknowledge. This active high signal indicates that the controller has accepted the TX packet request and the user logic should either submit a new request or deassert tx_cmd_req on the next rising edge of clk_byte.

48.3.2.2 Receive Ports

The Receive Packet Interface returns data from the Peripheral to the user. The user is provided the Virtual Channel (VC) number, Data Type (DT), Word Count (WC), and Rx Payload Data.

This interface enables the user application to receive any type of DSI packet. These signals are synchronous to the clk_byte clock. The Packet Interface Receive Interface ports are listed in the table below.

NOTE

Only a subset of the Packet Interface signals are accessible by user application through DSI_HOST_APB_PKT_IFx registers and DSI_HOST device registers:

- rx_payload
- rx_cmd_vc
- rx_cmd_data_type
- rx_cmd_byte_count
- ecc_one_bit_error
- ecc_two_bit_error
- ecc_one_bit_error_pos[4:0]
- crc_err

See DSI_HOST_APB_PKT_IFx and DSI_HOST device sections in [MIPI DSI Host register descriptions](#) for details.

Table 488. DSI Host Controller Core Receive Packet interface

PORT	TYPE	DESCRIPTION
rx_payload[31:0]	Output	Received packet data output. The Host Receive Packet Interface presents 4 bytes at a time. Bytes are valid in this interface according to the rx_cmd_byte_count signal, beginning with the lowest byte.
rx_payload_valid	Output	Packet data valid. This active high signal indicates that the controller is presenting valid packet during the next clk_byte period.
rx_payload_valid_last	Output	This data is the last of the packet, active high signals last cycle of rx_payload_valid.
rx_cmd_valid	Output	Packet header data is valid on the packet header ports below when this signal is asserted.
rx_cmd_vc[1:0]	Output	Packet virtual channel number, valid when rx_cmd_valid is asserted.
rx_cmd_data_type[5:0]	Output	Packet data type, valid when rx_cmd_valid is asserted. See the MIPI_DSI_HOST-2 specification for a definition of possible values.
rx_cmd_byte_count[15:0]	Output	Packet Word Count (byte count). Contains the number of bytes of data in the received packet. Valid when rx_cmd_valid is asserted.
ecc_one_bit_error	Output	Single bit error in the packet header was detected and corrected. Active high. Valid when rx_cmd_valid is high.
ecc_two_bit_error	Output	Two packet header bit errors were detected and not corrected, active high. Valid when rx_cmd_valid is high.
ecc_one_bit_error_pos[4: 0]	Output	Position of the corrected single bit error in the packet header. Valid when ecc_one_bit_error is high.
crc_err	Output	Asserts high when the CRC calculated on the received data does not reach the end of the packet.

48.3.3 Status Interface Ports

The Status Interface Core provides for reading the Status of the DSI Host Controller Core via a simple interface. All signals in the Status Interface are synchronous to the `clk_byte` clock. The Status Interface ports are listed in the table below.

NOTE

Only a subset of the Packet Interface signals are accessible by user application through `DSI_HOST_APB_PKT_IFx` registers and `DSI_HOST` device registers:

- `status_out`
- `hs_tx_timeout`
- `lp_rx_timeout`

Refer to `DSI_HOST_APB_PKT_IFx` and `DSI_HOST` device sections in [MIPI DSI Host register descriptions](#) for details.

Table 489. DSI Host Controller Status Interface Port

PORT	TYPE	DESCRIPTION
<code>status_rd</code>	Input	Initiates a read of the status register. Asserting this input high will result in the contents of the status register being presented on <code>status_out[15:0]</code> along with <code>status_out_valid</code> asserting high to indicate that <code>status_out[15:0]</code> contains a valid value.
<code>status_addr[3:0]</code>	Input	When <code>status_rd</code> is asserted, indicates which status register is to be returned. Address 0x0 is a valid address while 0x1-0xF are reserved.
<code>status_out[31:0]</code>	Output	Contains the contents of the addressed status register.
<code>status_out_valid</code>	Output	When asserted high, indicates that the data on <code>status_out</code> is valid.
<code>tx_active</code>	Output	<code>tx_active</code> asserts high when the Host Controller is actively transmitting data or when it has accepted a request from the user but has not yet started transmitting.
<code>hs_tx_timeout</code>	Output	Asserts high for one <code>clk_byte</code> period to indicate that a High Speed transmit has timed out.
<code>lp_rx_timeout</code>	Output	Asserts high for one <code>clk_byte</code> period to indicate that a Low Power RX occurred.

48.3.4 Control and Status Ports

The Control and Status Packet Interface allows the user to control and receive the status of the underlying link between the DSI Host and DSI Peripheral. It works on the same `clk_byte` clock as the Host Transmit Packet Interface. The Packet Interface Control and Status Interface ports are listed in the table below.

NOTE

Only a subset of the Packet Interface signals may be accessible by user application through DSI_HOST_APB_PKT_IFx registers and DSI_HOST device registers:

- dphy_direction
- tx_hs_mode
- dphy_turnaround

Please refer to DSI_HOST_APB_PKT_IFx and DSI_HOST device sections in [MIPI DSI Host register descriptions](#) for details.

Table 490. DSI Host Controller Core Control And Status Packet Interface

PORT	TYPE	DESCRIPTION
tx_hs_mode	Input	Switches the DPHY into High Speed Data Transfer mode or Low Power Data Transfer mode. 1'b1 = request HS mode 1'b0 = request LP mode. The Packet interface will not acknowledge packet commands or data while switching modes.
dphy_turnaround	Input	Requests bus turnaround. 1'b1 = Request reverse direction LP mode, from Host TX to Host RX. 1'b0 = No effect. This signal is ignored if the bus is already in Reverse (Host RX) direction.
dphy_direction	Output	Reports the current bus direction. 1'b1 = Bus is in Reverse direction (Host RX). 1'b0 = Bus is in Forward direction (Host TX).

48.3.5 Interface Cores

There are several interface cores for the DSI Host Controller Core. These interface cores provide industry standard interfaces to the DSI Host Controller Core to simplify interfacing and integration the DSI Host Controller Core to your application. When selected, these interface cores provide additional ports to the DSI Host Controller core.

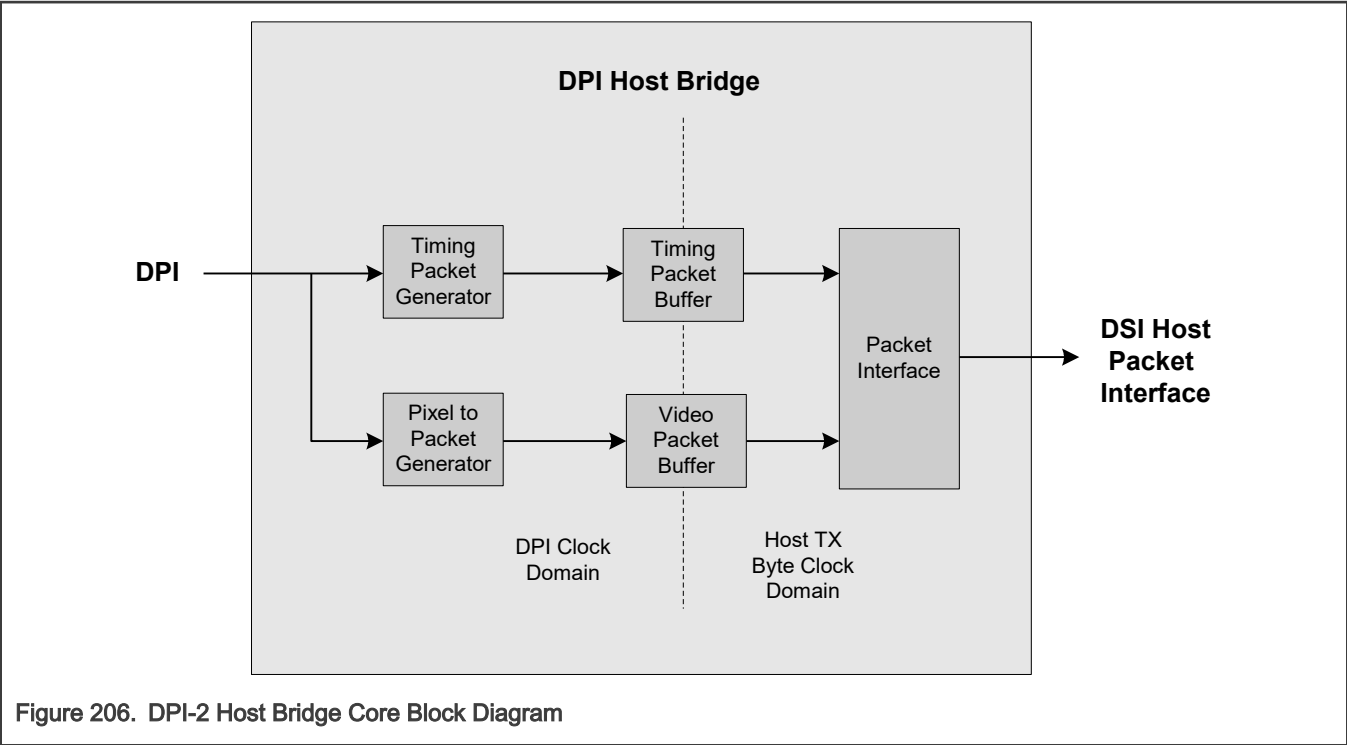
48.3.5.1 DPI-2 Interface Host Bridge Core

The DPI-2 Interface Host Bridge Core provides a DPI-2 compliant interface to the DSI Host controller. The DPI-2 Interface Host Bridge Core handles converting the DPI-2 interface into a packet format that is compatible with the DSI Host Controller Core's Packet Interface, and that adheres to the DSI specification. The DPI-2 Interface Core handles all pixel-to-byte packing for all DPI data types.

The DSI Host Controller DPI-2 Interface Core provides the following features:

- Support for Type 2,3, and 4 displays
- Support for 16-, 18- and 24- bit Pixel data and all alignment configurations
- Support for 16, 18, 24, and 18 bit loosely packed DSI data types
- Supports DSI video modes
 - Non-Burst Mode with Sync Pulses
 - Non-Burst Mode with Sync Events
 - Burst Mode
- Supports normal or inverted HSYNC and VSYNC signals
- Handles clock domain crossing from DPI Pixel clock to the Host controller TX Byte clock
- Interfaces directly to the Host Controller's DSI Packet Interface
- Comes already integrated with the DSI Host Controller

A block diagram of the DPI-2 Interface Host Bridge Core is shown below. The DPI-2 Host Bridge Core accepts a MIPI compliant DPI-2 set of signals (vsync, hsync, pixel data), creates DSI packets for the video timing events and video data and transmits those packets via the DSI Host Controller’s Packet Interface.



48.3.5.1.1 DPI-2 Interface Ports

The DPI-2 Interface module provides a MIPI compliant DPI-2 interface to the DSI Host Controller Core. In addition to the standard DPI-2 interface signals (prefixed with a dpi_) there are several configuration and status ports that allow the user to adjust the behavior of the DPI-2 Interface module to better suit a particular application.

NOTE

The DPI configuration signals herein described are accessible to User Application through chip-specific registers (please refer to chip-specific MIPI-DSI information section for details) and DSI_HOST_DPI_INTFC device registers (please refer to [MIPI_DSI_HOST Memory Map/Register](#) section for details).

Table 491. DSI Host Controller Core Receive Packet interface

PORT	TYPE	DESCRIPTION
dpi_sd	Input	Shut Down – Control to shutdown display (type 4 only) 1'b1= Send shutdown command. 1'b0= No effect
dpi_cm	Input	Color Mode control. 1'b0== Normal Mode 1'b1== Low-color Mode
cfg_dpi_pixel_payload_size [15:0]	Input	Maximum number of pixels that should send as one DSI packet. Recommended to be evenly divisible by the line size (in pixels).
cfg_dpi_pixel_fifo_send_level[15:0]	Input	In order to optimize DSI utility, the DPI bridge buffers a certain number of DPI pixels before initiating a DSI packet. This configuration port controls the level at which the DPI Host bridge begins sending pixels.

Table continues on the next page...

Table 491. DSI Host Controller Core Receive Packet interface (continued)

PORT	TYPE	DESCRIPTION
cfg_dpi_interface_color_coding[2:0]	Input	Sets the distribution of RGB bits within the 24-bit d bus, as specified by the DPI specification. 0= 16-bit Configuration 1 1= 16-bit Configuration 2 2= 16-bit Configuration 3 3= 18-bit Configuration 1 4= 18-bit Configuration 2 5= 24-bit
cfg_dpi_pixel_format[1:0]	Input	Sets the DSI packet type of the pixels. 0= 16-bit 1= 18-bit 2= 18-bit loosely packed, 3= 24-bit
dpi_host_underrun_err	Output	During DSI Host transmission of DPI data insufficient DPI data was received. This may indicate that DPI_CLK is too slow, or that the cfg_dpi_* parameters are incorrectly set.
cfg_dpi_vsync_polarity	Input	Sets polarity of dpi_vsync input, 0 – active low, 1 active high
cfg_dpi_hsync_polarity	Input	Sets Polarity of dpi_hsync input, 0 – active low, 1 – active high
cfg_host_dpi_video_mode[1:0]	Input	Select DSI video mode that the host DPI module should generate packets for. 2'b00 – Non-Burst mode with Sync Pulses 2'b01 – Non-Burst mode with Sync Events 2'b10 – Burst mode 2'b11 – Reserved, not valid
cfg_host_dpi_hfp[15:0]	Input	Sets the DSI packet payload size, in bytes, of the horizontal front porch blanking packet.
cfg_host_dpi_hbp[15:0]	Input	Sets the DSI packet payload size, in bytes, of the horizontal back porch blanking packet.
cfg_host_dpi_hsa[15:0]	Input	Sets the DSI packet payload size, in bytes, of the horizontal sync active packet.
cfg_enable_mult_packets	Input	Enable Multiple packets per video line. When enabled, cfg_dpi_pixel_payload_size must be set to exactly half the size of the video line. 0 – Video Line is sent in a single packet 1 – Video Line is sent in two packets
cfg_bllp_mode	Input	Optimize bllp periods to Low Power mode when possible 0 – blanking packets are sent during BLLP periods 1 – LP mode is used for BLLP periods

48.3.5.1.2 DPI-2 Video Timing Recreation over DSI

The DSI Host DPI-2 module is designed to preserve the receive video timing by properly spacing the DSI packets that carry the DPI video information and by inserting blanking packets into the packet stream. In order to achieve this goal, the DSI Host controller needs to have the cfg_host_dpi_hfp, cfg_host_dpi_hbp, and cfg_host_dpi_hsa ports properly set. The values of these ports reflect the size of the hfp, hbp, and has in terms of DSI packet bytes, values such that the number of DSI bytes when transmitted takes approximately the same amount of absolute time as the video event took on the DPI interface.

To relate pixel sizes on the DPI interface to DSI packet bytes, use the following relationship:

$$\text{Time of DPI event} = \text{time to transmit} \times \text{number of bytes on the DSI interface}$$

$$\text{dpi\_event\_size} \times \text{dpi\_pclk\_period} = \text{number\_of\_dsi\_bytes} \times 8 \times \text{dsi\_hs\_bit\_period} / \text{cfg\_num\_lanes}$$

Solving for number_of_dsi_bytes:

$$\text{number\_of\_dsi\_bytes} = \text{dpi\_event\_size} \times \text{dpi\_pclk\_period} \times \text{cfg\_num\_lanes} / (8 \times \text{dsi\_hs\_bit\_period})$$

number_of_dsi_bytes	value to set the cfg_host_dpi_hfp (hbp, or hsa) port to
dpi_event_size	size of the video event in pixels (hfp, hbp, hsa)
dpi_pclk_period	period of dpi_pclk
dsi_hs_bit_period	bit period of the data lanes in High Speed mode
cfg_num_lanes	host controller setting for number of active lanes (0 = 1 lane, 1 = 2 lanes, 2 = 3 lanes, 3 = 4 lanes)

48.3.6 RAM Usage

The DSI Host Controller Core requires on-chip RAM arrays. These RAM arrays and their corresponding sizes are listed in the table below.

All RAM arrays used in this core are configured as two-port, having the following attributes:

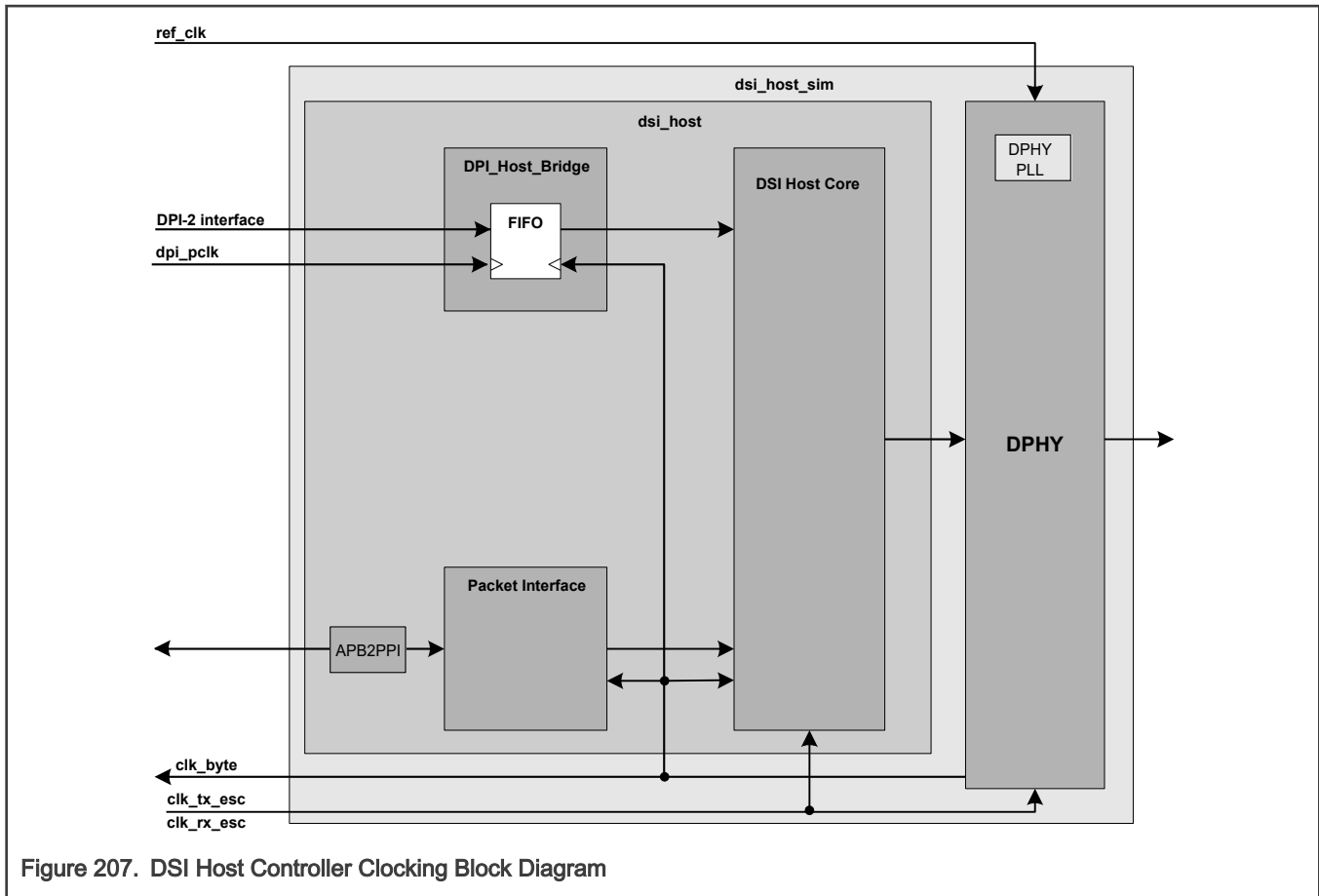
- Two address buses, one for the write port and one for read port
- Two data buses, one for write data on the write port and one for read data on the read port
- Two clock inputs, one for the write port and one for the read port. (Note: In some cases, the core logic will connect the same clock to both the read and write clock inputs of the RAM.)

Ram Purpose	Size (Depth x Width)
DPI Core Pixel Data FIFO	2048 x 32
APB to Packet Interface TX FIFO	64 x 32
APB to Packet Interface RX FIFO	64 x 32

48.3.7 DSI Host Clocking

The DSI Host Controller requires the following clocks:

- Reference clock (ref_clk) - used by the DPHY (DPHY PLL generates byte clock)
- Byte clock (clk_byte) - generated by the DPHY, used for packet generation and transfer to the DPHY
- TX escape mode clock (clk_tx_esc) - for internal LPDT state machines and low-power transmissions;
- Pixel clock (dpi_pclk)



ref_clk

The reference clock is used by the DPHY (DPHY PLL) to generate the High Speed MIPI clock and High Speed Data Lane signaling. The frequency of the resulting clock generated from DPHY PLL is typically equal to the high speed bit rate of the data lanes. The DPHY will also use this clock to generate the `clk_byte` clock output used by the Host DSI Controller and user application logic.

The MIPI DPHY PLL must be setup through the MIPI_DSI_HOST_FSL_IP1_DPHY INTFCx registers, prior to starting DSI Host Controller and DPHY operation. For details, please refer to MIPI_DSI_HOST IP1 DPHY INTFC Memory section.

clk_byte

The `clk_byte` clock is generated by the DPHY (DPHY PLL) and the frequency is 1/8th the data rate of the DPHY Data Lane High Speed data rate. A DPHY configured to generate High Speed data at 1Gbps/lane would generate a 125MHz `clk_byte` clock.

The `clk_byte` clock is used by the DSI Host Controller to interface to the MIPI Tx DPHY High Speed PPI interface. The DSI Host Controller's Packet Interface is synchronous to the `clk_byte` clock.

dpi_pclk

The `dpi_pclk` clock is used on the Host DPI-2 interface. All of the Host DPI-2 signals are synchronous to this clock. The DSI Host Controller's DPI Bridge module handles transferring video data received in the `dpi_pclk` clock domain over to the `clk_byte` clock domain.

The `dpi_pclk` and `clk_byte` frequencies are related by the following formula: $\text{clk\_byte\_freq} \geq \text{dpi\_pclk\_freq} * \text{DPI\_pixel\_size} / (8 * (\text{cfg\_num\_lanes} + 1))$

cfg_num_lanes = the configuration port setting that selects the number of active MIPI DPHY data lanes
clk_byte_freq = frequency of clk_byte which is 1/8th the High Speed data lane rate.
dpi_pclk_freq = frequency of the dpi_pclk clock on the DPI-2 interface.
DPI_pixel_size = size of pixels, in bits, on the DPI-2 interface

If the clk_byte frequency does not meet this requirement, the MIPI interface will not be able to keep up with the video stream on the DPI-2 interface resulting in dropped video lines.

clk_tx_esc

The clk_tx_esc clock is used by the MIPI Tx DPHY for state control and low power data transmission. The DSI Host Controller also uses clk_tx_esc for the portion of controller logic that interfaces to the MIPI Tx DPHY that are synchronous to clk_tx_esc. The frequency of clk_tx_esc is defined by the requirements of the MIPI interface and the MIPI Tx DPHY. For details about the clk_tx_esc source, please refer to the chip-specific MIPI-DSI clocking section.

48.3.8 Reset and Initialization

Reset and Initialization procedure for the MIPI_DSI_HOST Controller is as follows:

1. Assert all resets through the chip-specific registers provided for this purpose:
 - reset_dpi_n
 - reset_byte_n
 - reset_esc_n
2. Setup MIPI_DSI_HOST related clocks through chip-specific registers provided for this purpose.
3. Setup MIPI_DSI_HOST application by programming all MIPI_DSI_HOST Controller and DSI DPHY required parameters (refer to [MIPI_DSI_HOST Memory Map/Register](#) for details).
4. Deassert all remaining resets through the Chip-specific registers provided for this purpose:
 - reset_dpi_n
 - reset_byte_n
 - reset_esc_n
5. DSI Host Controller should be ready for use

48.4 DSI D-PHY

The figure below shows the block diagram of the MIPI_DSI_HOST D-PHY. It consists of a clock lane module and two data lane modules. Each of these PHY Lane Modules communicates through a differential line to a complementary PHY at the other side of the lane interconnect.

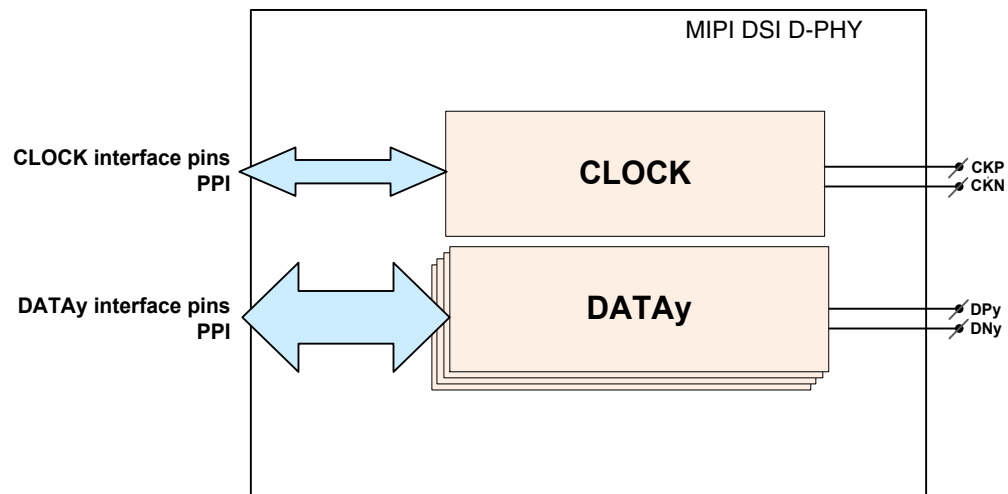


Figure 208. MIPI D-PHY Block Diagram

The figure below shows the D-PHY Lane components:

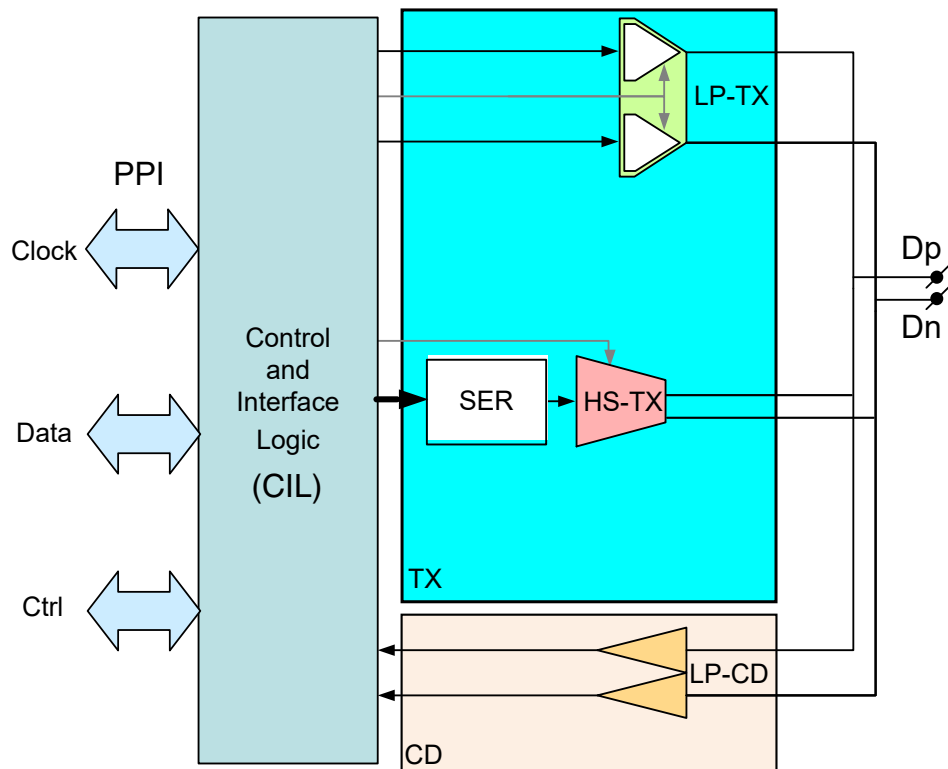


Figure 209. D-PHY Lane Diagram

The DATAy are bidirectional lanes and consist of High-Speed Transmitter (HS-TX), Low-Power Transmitter (LP-TX), Serializer (SER), and Low- Power Contention Detector (LP-CD).

The Clock lane is a bidirectional lane and consists of High-Speed Transmitter (HS-TX), and Low-Power Transmitter (LP-TX).

The Control and Interface Logic (CIL) module interfaces with the Protocol and determines the global operation of the Lane Module. The interface between the D-PHY and the protocol is called the PHY-Protocol Interface (PPI).

During normal operation, a Lane switches between Low-Power and High-Speed mode. Bidirectional Lanes can also switch communication direction. The change of operating mode or direction requires enabling and disabling of certain electrical functions. These enable and disable events do not cause glitches on the Lines that would result in a detection of an incorrect signal level. Therefore, all mode and direction changes are smooth to always ensure a proper detection of the Line signals.

48.4.1 High-Speed Transmitter (HS-TX)

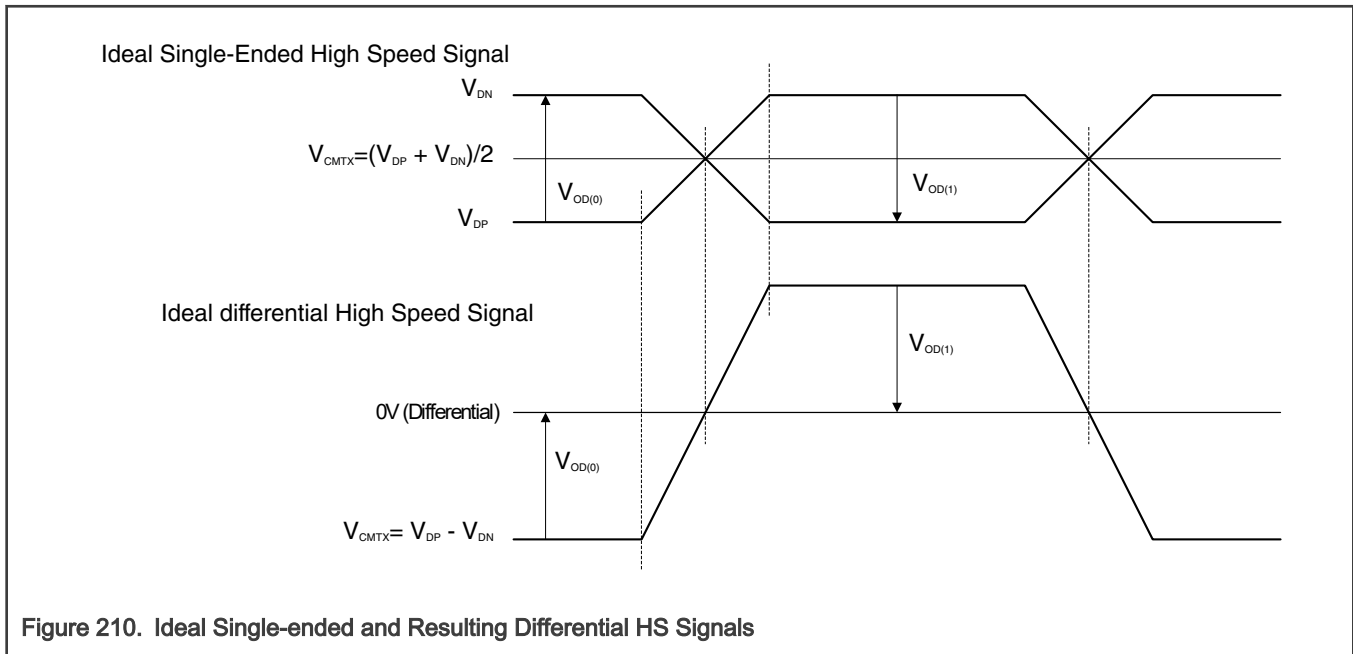
The differential output voltage V_{OD} is defined as the difference of the voltages V_{DP} and V_{DN} at the Dp and Dn pins, respectively.

$$V_{OD} = V_{DP} - V_{DN}$$

The output voltages V_{DP} and V_{DN} at the Dp and Dn pins will not exceed the high-speed output high voltage V_{OHHS} . The common-mode voltage V_{CMTX} is defined as the arithmetic mean value of the voltages at the Dp and Dn pins:

$$V_{CMTX} = (V_{DP} + V_{DN})/2$$

V_{OD} and V_{CMTX} are graphically shown in Figure 210 for ideal HS signals. Figure 211 shows single ended HS signals with the possible kinds of distortion of the differential output and common mode voltages. V_{OD} and V_{CMTX} may be slightly different for driving a Differential-1 or a Differential-0 on the pins.



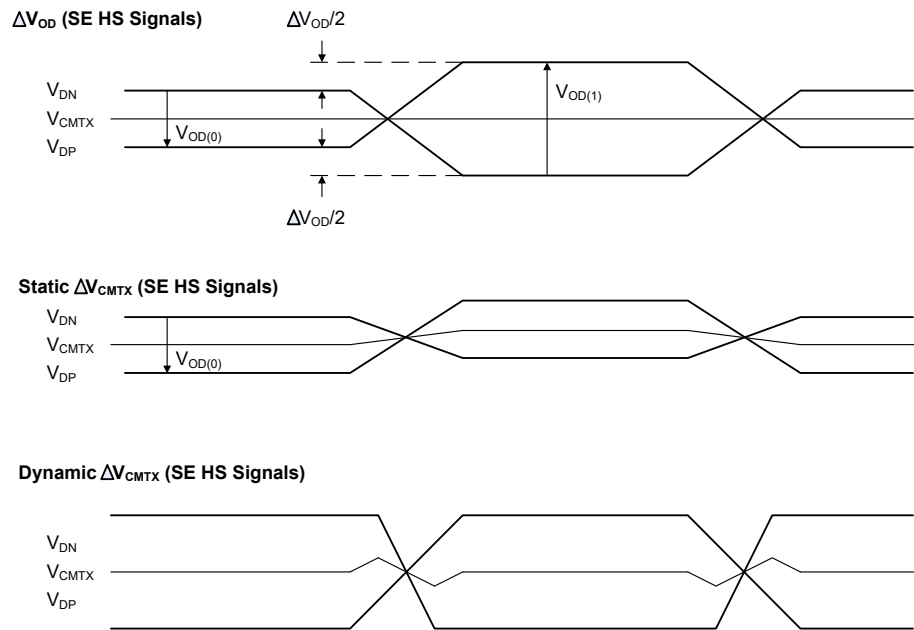


Figure 211. Possible ΔV_{CMTX} and ΔV_{OD} Distortions of the Single-Ended HS Signals

The output differential voltage mismatch ΔV_{OD} is defined as the difference of the absolute values of the differential output voltage in the Differential-1 state $V_{OD(1)}$ and the differential output voltage in the Differential-0 state $V_{OD(0)}$. This is expressed by:

$$\Delta V = |V_{OD(1)}| - |V_{OD(0)}|$$

If $V_{CMTX(1)}$ and $V_{CMTX(0)}$ are the common-mode voltages for static Differential - 1 and Differential - 0 states respectively, then the common-mode reference voltage is defined by:

$$V_{CMTX, REF} = (V_{CMTX(1)} + V_{CMTX(0)})/2$$

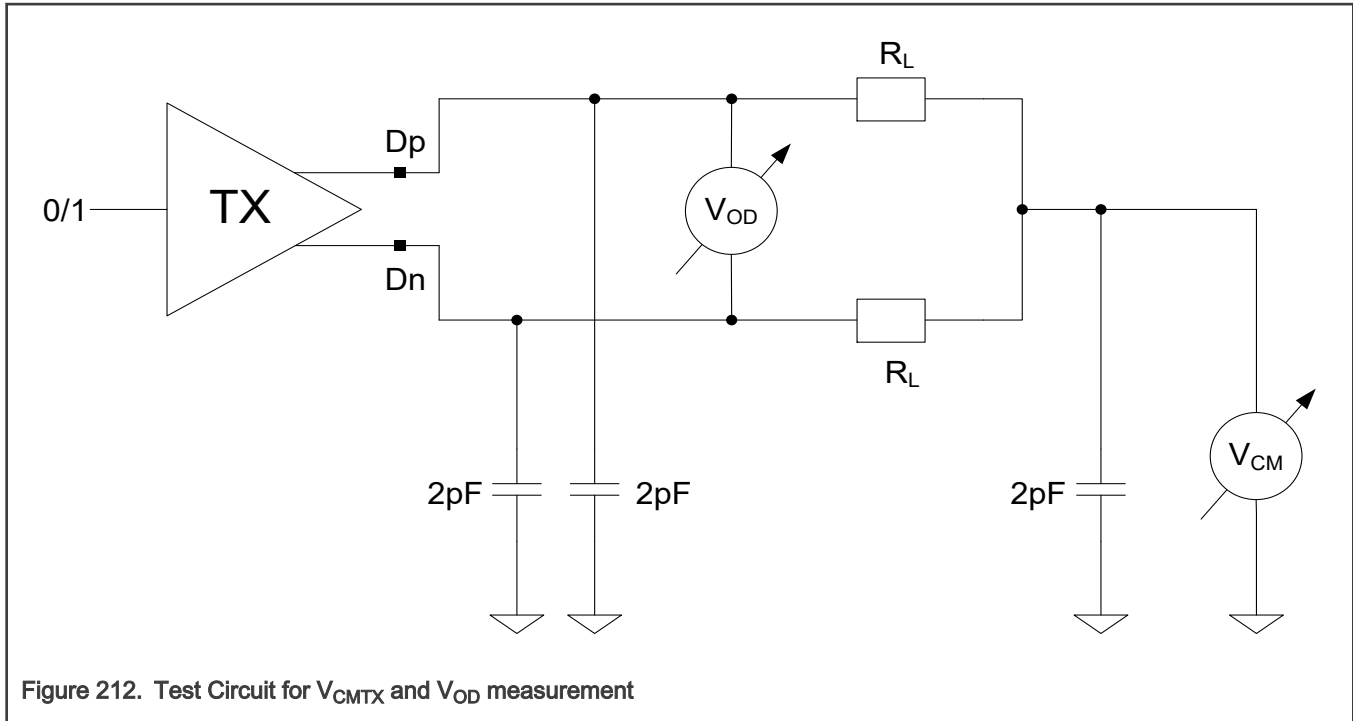
The transient common-mode voltage variation is defined by:

$$\Delta V_{CMTX(t)} = V_{CMTX(t)} - V_{CMTX, REF(t)}$$

The static common-mode voltage mismatch between the Differential-1 and Differential-0 state is given by:

$$\Delta V_{CMTX(1,0)} = (V_{CMTX(1)} - V_{CMTX(0)})/2$$

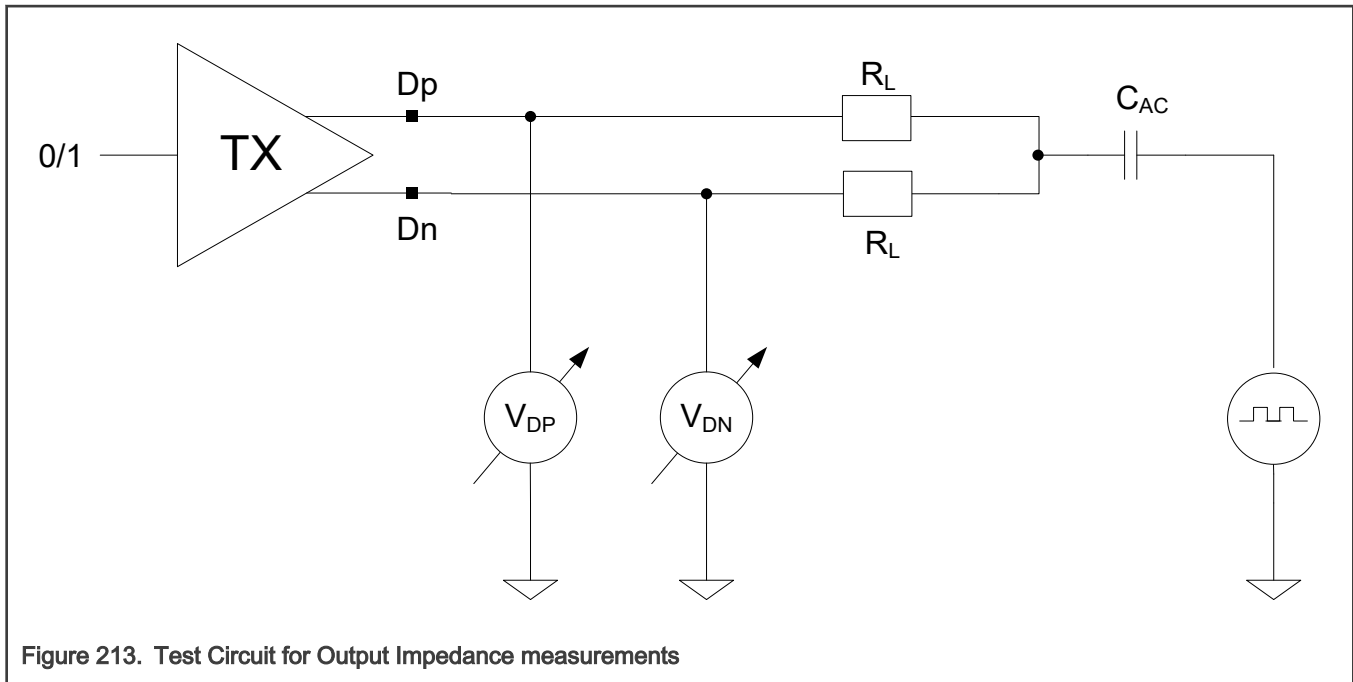
A test circuit for the measurement of V_{OD} and V_{CMTX} is shown:



The single-ended output impedance of the transmitter at both the Dp and Dn pins is denoted by Z_{OS} . ΔZ_{OS} is the mismatch of the single ended output impedances at the Dp and Dn pins, denoted by Z_{OSDP} and Z_{OSDN} respectively. This mismatch is defined as the ratio of the absolute value of the difference of Z_{OSDP} and Z_{OSDN} and the average of those impedances:

$$\Delta Z_{OS} = (2 * |Z_{OSDP} - Z_{OSDN}|) / (Z_{OSDP} + Z_{OSDN})$$

The output impedance Z_{OS} and the output impedance mismatch ΔZ_{OS} is compliant with High Speed Transmitter DC Specifications for both the Differential-0 and Differential-1 states for all allowed loading conditions. It is recommended the design keep the output impedance during state transitions as close as possible to the steady state value. The output impedance Z_{OS} can be determined by injecting an AC current into the Dp and Dn pins and measuring the peak-to-peak voltage amplitude. The test circuit for such a measurement is shown in the following figure, where R_L are load resistors and $V_{TEST,PP}$ is a test signal applied at the common-mode node via an AC coupling capacitor CAC. The frequency of the test signal is f_{TEST} .



Assuming negligible effect of the coupling capacitor, the example circuit gives the following relation between peak-to-peak pin voltage $V_{PIN,PP}$ and Z_{OS} . This equation can be used for both Dn and Dp pins individually.

$$Z_{OS} = (V_{PIN,PP} * R_L) / (V_{TEST,PP} - Z_{PIN,PP})$$

Where $V_{PIN,PP}$ is the peak-to-peak voltage at the pin Dp or Dn.

48.4.2 Low-Power Transmitter (LP-TX)

The Low-Power transmitter is a slew-rate controlled push-pull driver. It is used for driving the Lines in all Low-Power operating modes. The LP transmitter has been optimized to minimize static power consumption. The slew-rate of signal transitions is bounded in order to keep EMI low. The LP transmitter will not drive the pad pin potential statically beyond the maximum value of V_{OH} . V_{OH} is the thevenin output, high-level voltage in the high-level state, when the pad pin is not loaded. V_{OL} is the thevenin output, low-level voltage in the LP transmit mode. This is the voltage at an unloaded pad pin in the low-level state.

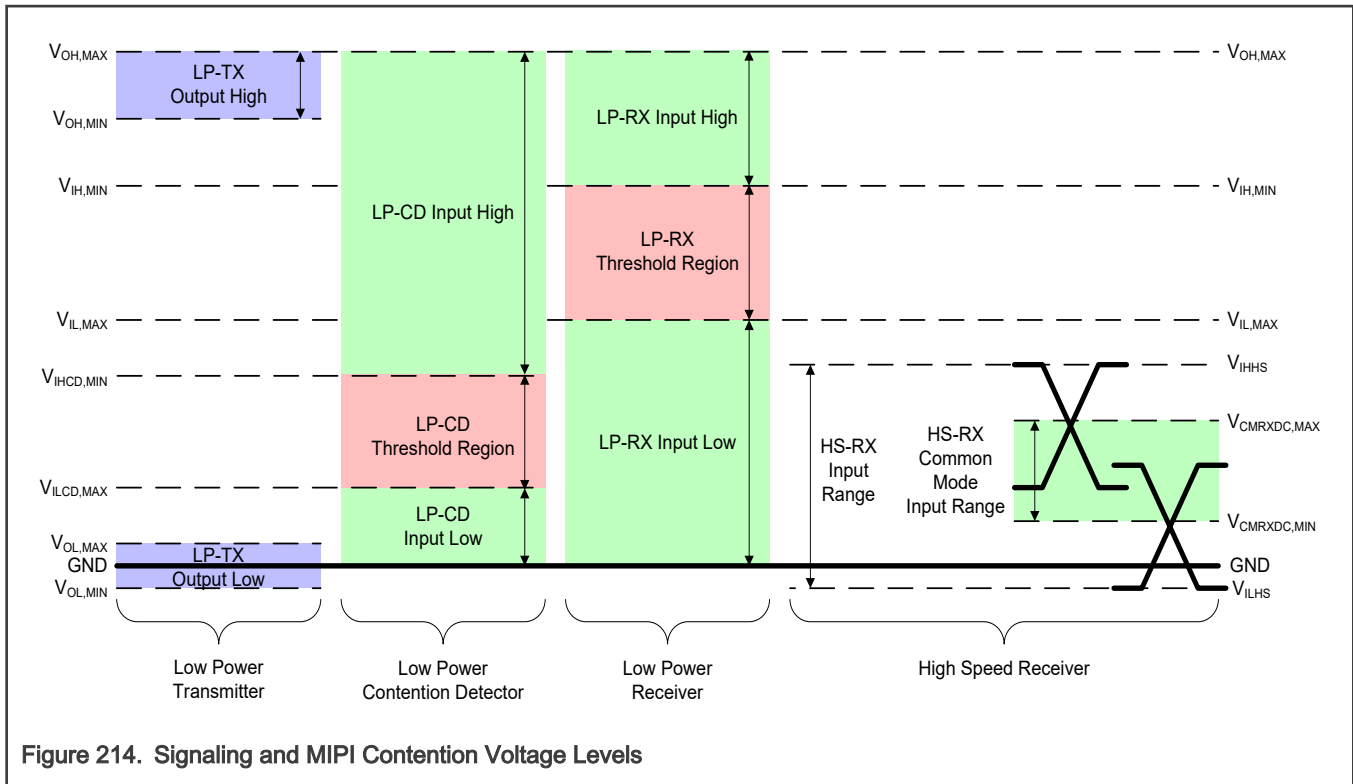
48.4.3 Low-Power Contention Detector (LP-CD)

Contention can be inferred from any of the following conditions:

- LP high fault is detected when the LP transmitter is driving high and the pin voltage is less than V_{IL} .
- LP low fault is detected when the LP transmitter is driving low and the pad pin voltage is greater than V_{IHCD} . An LP low fault will not be detected when the pin voltage is less than V_{ILCD} .

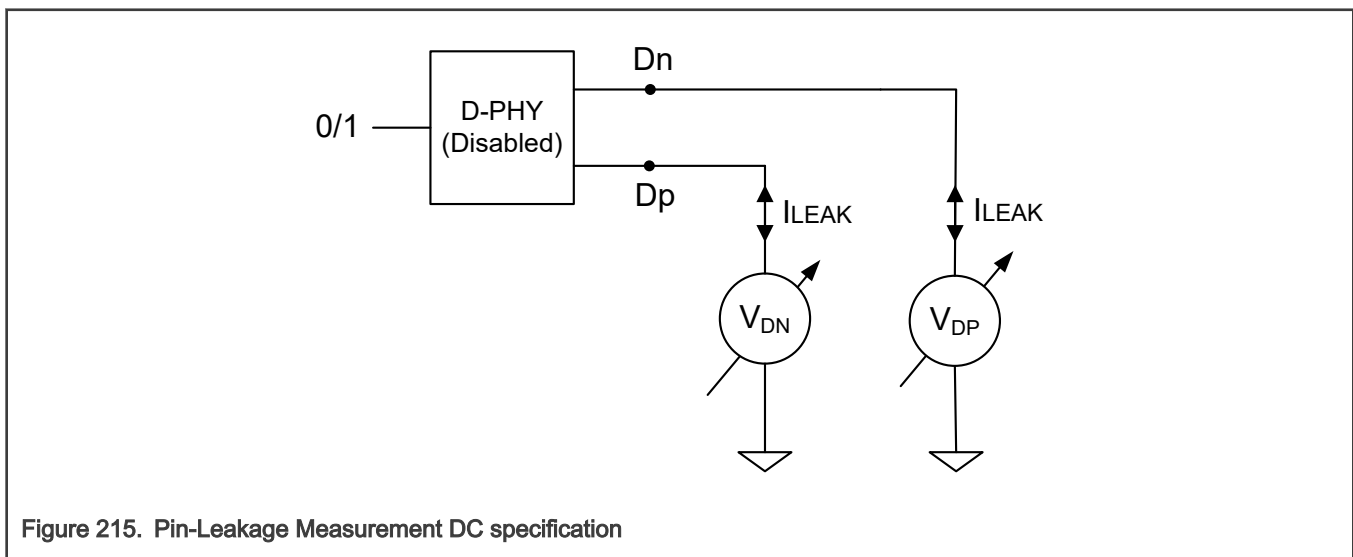
The general operation of a contention detector is similar to that of an LP receiver with lower threshold voltages. Although the DC specifications differ, the AC specifications of the LP-CD are defined to match those of the LP receiver. The LP-CD sufficiently filters the input signal to avoid false triggering on short events

The LP-CD threshold voltages (V_{ILCD} , V_{IHCD}) are shown along with the normal signaling voltages:



48.4.4 Input Characteristics

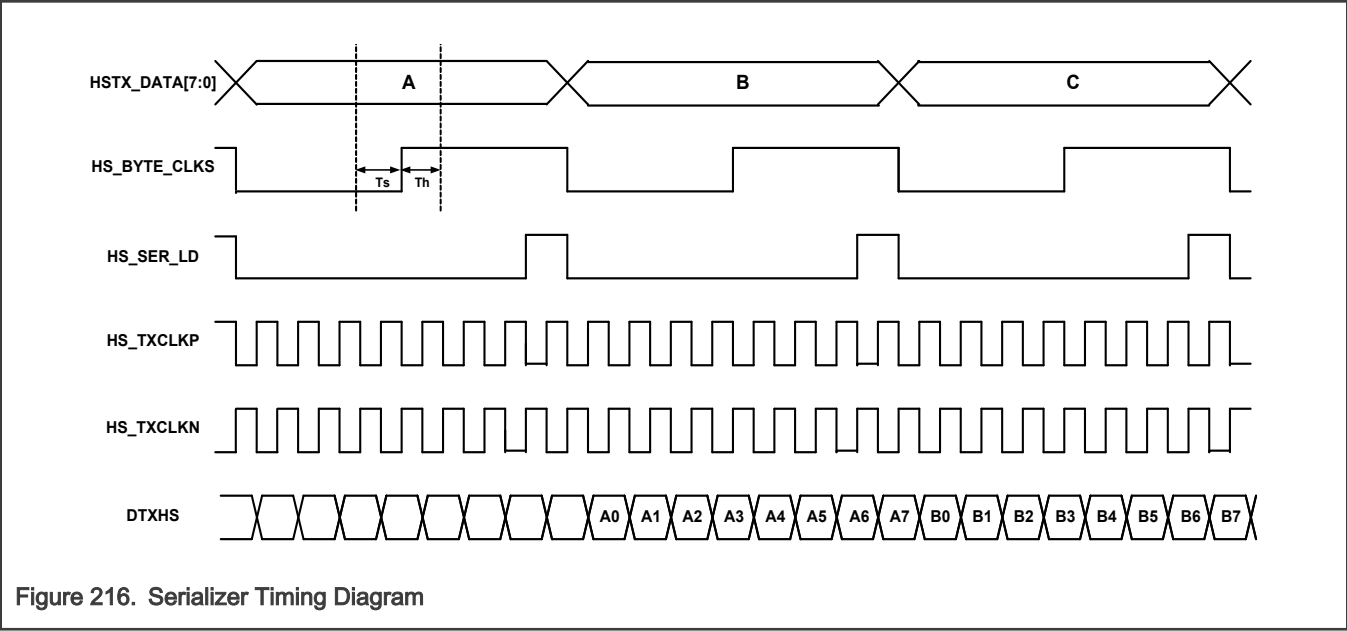
No structure within the PHY may be damaged when a DC signal that is within the signal voltage range V_{PIN} is applied to a pad pin for an indefinite period of time. $V_{PIN(ABSMAX)}$ is the maximum transient output voltage at the transmitter pin. The voltage on the transmitter's output pin will not exceed $V_{PIN,MAX}$ for a period greater than $T_{VPIN(ABSMAX)}$. When the PHY is in the low-power receive mode the pad pin leakage current will be I_{LEAK} when the pad signal voltage is within the signal voltage range of V_{PIN} . The specification of I_{LEAK} assures interoperability of any PHY in the LP mode by restricting the maximum load current of an LP transmitter. The test circuit for leakage current measurement is shown:



48.4.5 High-Speed Serializer

The HS-Serializer converts 8-bits parallel data into 1 bit data stream to be transmitted by HSTX module. The CIL engine should clock out 8-bits parallel data, HSTX_DATA[7:0], on D0_HS_BYTE_CLKS rising edges. Then the Serializer will capture HSTX_DATA[7:0] on D0_HS_BYTE_CLKS rising edges and the signal connected to HSTX_DATA[0] is transmitted first. There is setup timing constraint and hold timing constraint designated to the rising edge of D0_HS_BYTE_CLKS as constrained in the liberty view provided.

The figure shows the relationship between HS_TXCLKP, HS_TXCLKN, HS_SER_LD, HS_BYTE_CLKS and HSTX_DATA[7:0]. It also shows how the 8-bits of parallel bits are mapped into the serial link:



48.4.6 DSI D-PHY Signals

This section details the interface signals for MIPI DSI D-PHY. For detailed signal spec, please refer to the chip data sheet.

Table 492. DSI DPHY PPI Signals

Signal	Type	Description
TxCkEsc	Clock	Escape mode Transmit Clock. This clock is directly used to generate escape sequences. The period of this clock determines the symbol time for low power signals. It is therefore constrained by the normative part of the DPHY specification. NOTE The max frequency of TxCkEsc is 20 MHz NOTE The Min frequency of TxCkEsc is 12 MHz.

Table continues on the next page...

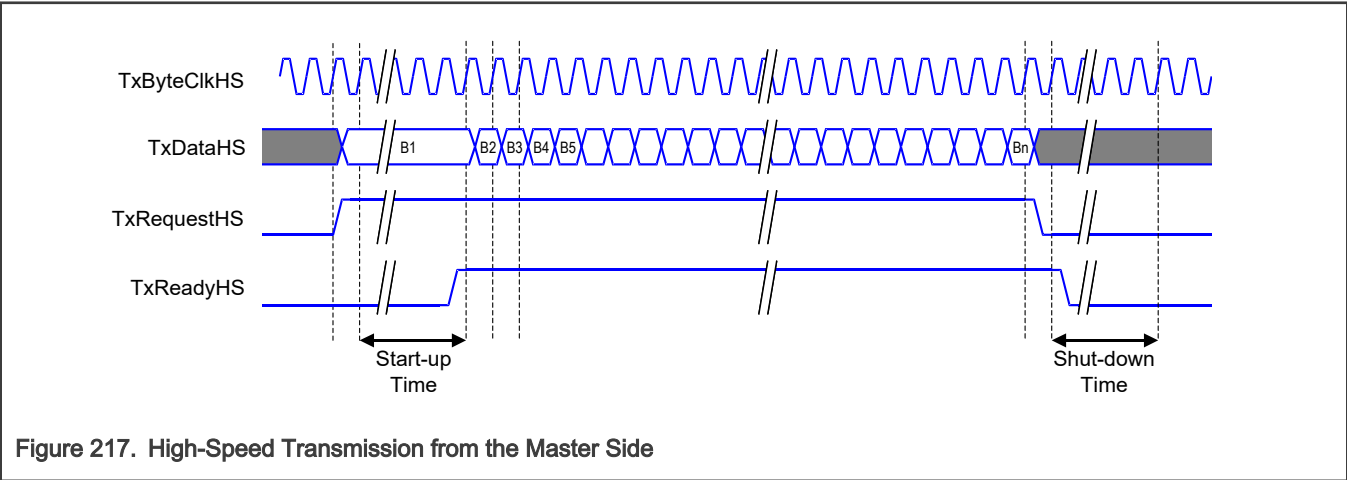
Table 492. DSI DPHY PPI Signals (continued)

Signal	Type	Description
TxByteClkHS	Clock	High-Speed Transmit Byte Clock. This is used to synchronize PPI signals in the High-Speed transmit clock domain. It is recommended that all transmitting Data Lane Modules share on TxByteClkHS signal. The frequency of TxByteClkHS is exactly 1/8 the High-Speed bit rate.

48.4.7 Operating Modes

48.4.7.1 High-Speed Transmit from the Master Side

The figure shows an example of a High-Speed transmission on the Master side. While TxRequestHS is low, the Lane Module ignores the value of TxDataHS. To begin transmission, the protocol drives TxDataHS with the first byte of data and asserts TxRequestHS. This data byte is accepted by the PHY on the first rising edge of TxByteClkHS with TxReadyHS also asserted. At this point, the protocol logic drives the next data byte onto TxDataHS. After every rising clock cycle with TxReadyHS active, the protocol supplies a new valid data byte or ends the transmission. After the last data byte has been transferred to the Lane Module, TxRequestHS is driven low to cause the Lane Module to stop the transmission and enter Stop state. The minimum number of bytes transmitted could be as small as one.



48.4.7.2 Low-Power Data Transmission

For Low-Power data transmission the TxClkEsc is used. The Protocol directs the Data Lane to enter Low-Power data transmission Escape mode by asserting TxRequestEsc with TxLpdtEsc high. The Low-Power transmit data is transferred on the TxDataEsc lines when TxValidEsc and TxReadyEsc are both active at a rising edge of TxClkEsc. The byte is transmitted in the time after the TxDataEsc is accepted by the Lane Module (TxValidEsc = TxReadyEsc = high). The Protocol knows the byte transmission is finished when TxReadyEsc is asserted. After the last byte has been transmitted, the protocol de-asserts TxRequestEsc to end the Low-Power data transmission. This causes TxReadyEsc to return low. Whenever TxRequestEsc transitions from high-to-low, it always remains in the low state for a minimum of two TxClkEsc clock cycles. The figure shows an example Low-Power data transmission operation.

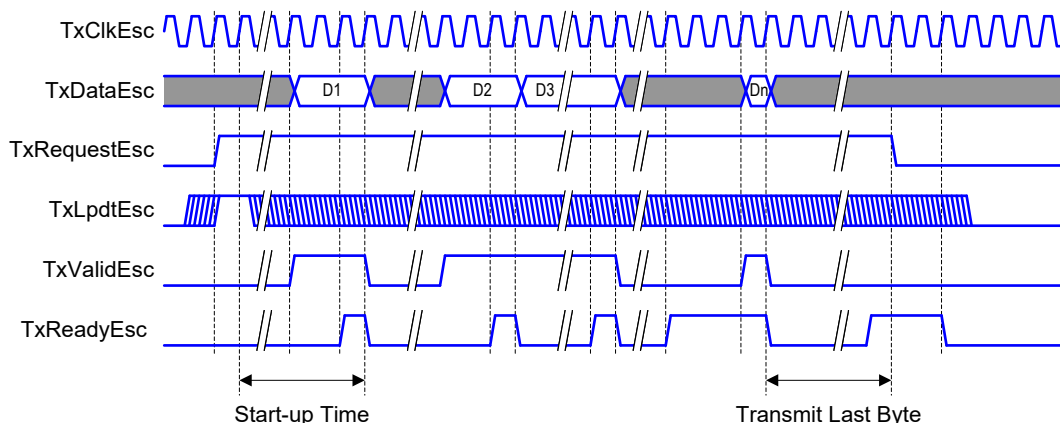


Figure 218. Low-Power Data Transmission

48.4.7.3 Turn-around

If the Master side and Slave side Lane Modules are both bi-directional, it is possible to turn around the Link for Escape mode signaling. Which side is allowed to transmit is determined by passing a “token” back and forth. That is, the side currently transmitting passes the token to the receiving side. If the receiving side acknowledges the turn-around request, as indicated by driving the appropriate line state, the direction is switched. The figure shows an example of two turn-around events. At the beginning, the local side is the transmitter, as shown by Direction=0. When the protocol on this side wishes to turn the Lane around (i.e. give the token to the other side), it asserts TurnRequest for at least one cycle of TxClkEsc. This initiates the turn around procedure. The remote side acknowledges the turn-around request by driving the appropriate states on the Lines. When this happens, the local Direction signal changes from transmit (0) to receive (1). The remote side initiates a turn-around request, passing the token back to the local side. When this happens, the local Direction signal changes back to transmit (0). The transmitter is in control of the Link direction and decides when to turn the Link around, passing control to the receiver. If the remote side does not acknowledge the turnaround request, the Direction signal does not change.

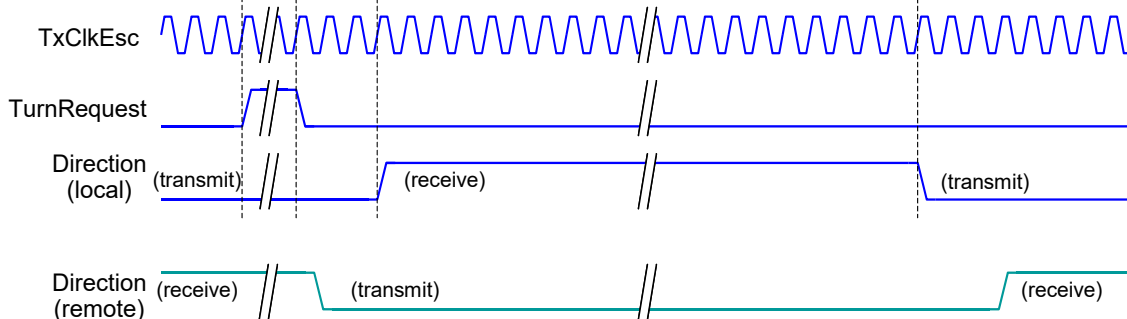


Figure 219. Turn-around Actions Transmit-to-Receive and Back to Transmit

48.5 DPHY Programming

The following sections detail the programming of the MIPI_DSI_HOST DPHY high-speed timers.

High-Speed Prepare Timer ($T_{HS-PREPARE}$)

Calculate the value for clock lane high-speed prepare timer (mc_PRG_HS_PREPARE) using the formula below:

Assuming TxClkEsc = 20 MHz ($T_{per} = 50$ ns)

When `mc_PRG_HS_PREPARE = 0`, $T_{CLK-PREPARE} = 1 * T_{TxClkEsc} \text{ Period} = 50 \text{ ns}$

When `mc_PRG_HS_PREPARE = 1`, $T_{CLK-PREPARE} = 1.5 * T_{TxClkEsc} \text{ Period} = 75 \text{ ns}$

Calculate the value for data lane high-speed prepare timer (`m_PRG_HS_PREPARE`) using the formula below:

Assuming $T_{TxClkEsc} \text{ (TLPX)} = 20 \text{ MHz}$ ($T_{per} = 50 \text{ ns}$)

When `m_PRG_HS_PREPARE = 00`, $T_{HS-PREPARE} = 1 * T_{TxClkEsc} \text{ Period} = 50 \text{ ns}$

When `m_PRG_HS_PREPARE = 01`, $T_{HS-PREPARE} = 1.5 * T_{TxClkEsc} \text{ Period} = 75 \text{ ns}$

When `m_PRG_HS_PREPARE = 10`, $T_{HS-PREPARE} = 2 * T_{TxClkEsc} \text{ Period} = 100 \text{ ns}$

When `m_PRG_HS_PREPARE = 11`, $T_{HS-PREPARE} = 2.5 * T_{TxClkEsc} \text{ Period} = 125 \text{ ns}$

High-Speed Zero Timer($T_{HS-ZERO}$)

Calculate the value for clock lane high-speed zero timer (`mc_PRG_HS_ZERO`) using the formula below:

$T_{CLK-ZERO} = (\text{mc\_PRG\_HS\_ZERO} + 3) * (T_{TxByteClkHS} \text{ Period})$

$T_{TxByteClkHS} = 1/8 \text{ Data Rate}$

When `mc_PRG_HS_ZERO = 000110 @ 250 Mb/s`, $T_{CLK-ZERO} = (6+3) * 32 \text{ ns} = 288 \text{ ns}$

When `mc_PRG_HS_ZERO = 100000 @ 1 Gb/s`, $T_{CLK-ZERO} = (32+3) * 8 \text{ ns} = 280 \text{ ns}$

Calculate the value for data lane high-speed zero timer (`m_PRG_HS_ZERO`) using the formula below:

$T_{HS-ZERO} = (\text{m\_PRG\_HS\_ZERO} + 6) * (T_{TxByteClkHS} \text{ Period})$

$T_{TxByteClkHS} = 1/8 \text{ Data Rate}$

When `m_PRG_HS_ZERO = 00001 @ 250 Mb/s`, $T_{HS-ZERO} = (1+6) * 32 \text{ ns} = 224 \text{ ns}$

When `m_PRG_HS_ZERO = 01001 @ 1 Gb/s`, $T_{HS-ZERO} = (9+6) * 8 \text{ ns} = 120 \text{ ns}$

High-Speed Trail Timer ($T_{HS-TRAIL}$)

Calculate the value for the high-speed trail timer (`PRG_HS_TRAIL`) using the formula below:

$T_{HS-TRAIL} = (\text{PRG\_HS\_TRAIL}) * (T_{TxByteClkHS} \text{ Period})$

$T_{TxByteClkHS} = 1/8 \text{ Data Rate}$

When `PRG_HS_TRAIL = 0100 @ 250 Mb/s`, $T_{HS-TRAIL} = 4 * 32 \text{ ns} = 128 \text{ ns}$

When `PRG_HS_TRAIL = 1100 @ 1 Gb/s`, $T_{HS-TRAIL} = 12 * 8 \text{ ns} = 96 \text{ ns}$

48.6 Packet Data and Pixel Formats

The DSI Host Controller Core and the external DSI Peripheral explicitly handle and exchanges 32-bit words encapsulated as packet data. The mapping between pixel formats and pixel data must be done on both sides by user application on both sides of the DSI link. The DSI spec gives many examples of common pixel formats. The DSI Host Controller Core and external DSI Peripheral have no preference or limitation on which pixel formats it will handle. The IP which provides video source packet data through DPI-2 interface (e.g. LCDIF) onto DSI should also be configured by the user application to match the pixel formats expected by the DSI Host Controller.

The DPI-2 interface core handles pixel to byte packing and byte to pixel unpacking according to the MIPI_DSI_HOST specification.

48.7 MIPI_DSI_HOST Memory Map/Register Definition

MIPI_DSI_HOST Controller Register Groups	Offset
DSI_HOST	0x00
DBI_INTFC	0x100
DPI_INTFC	0x200
APB_PKT_IF	0x280
DPHY_INTFC	0x300

48.7.1 MIPI DSI Host register descriptions

48.7.1.1 MIPI_DSI_HOST memory map

MIPI_DSI_HOST base address: 4003_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Configure Number of Lanes (CFG_NUM_LANES)	32	RW	0000_0000h
4h	Configure Non-continuous Clock (CFG_NONCONTINUOUS_CLK)	32	RW	0000_0000h
8h	Configure Pre Clock Periods (CFG_T_PRE)	32	RW	0000_0000h
Ch	Configure Post Clock Periods (CFG_T_POST)	32	RW	0000_0000h
10h	Configure Gap Clock Periods (CFG_TX_GAP)	32	RW	0000_0000h
14h	Configure Autoinsert EOTP (CFG_AUTOINSERT_EOTP)	32	RW	0000_0000h
18h	Configure Extra Commands after EOTP (CFG_EXTRA_CMDS_AFTER_EOTP)	32	RW	0000_0000h
1Ch	Configure High Speed Tx Timeout Count (CFG_HTX_TO_COUNT)	32	RW	0000_0000h
20h	Configure Low Power Rx Timeout Count (CFG_LRX_H_TO_COUNT)	32	RW	0000_0000h
24h	Configure Bus Turn Around Timeout Count (CFG_BTA_H_TO_COUNT)	32	RW	0000_0000h
28h	Configure Twakeup (CFG_TWAKEUP)	32	RW	0000_0000h
2Ch	Configure Status Out (CFG_STATUS_OUT)	32	R	0000_0000h
30h	Receive Error Status (RX_ERROR_STATUS)	32	R	0000_0000h
100h	Pixel Payload Size (CFG_DBI_PIXEL_PAYLOAD_SIZE)	32	RW	0000_0000h
104h	Configure DBI Pixel FIFO Send Level (CFG_DBI_PIXEL_FIFO_SEND_LEVEL)	32	RW	0000_0000h
200h	Configure DPI Pixel Payload Size (CFG_DPI_PIXEL_PAYLOAD_SIZE)	32	RW	0000_0000h
204h	Configure DPI Pixel FIFO Send Level (CFG_DPI_PIXEL_FIFO_SEND_LEVEL)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
208h	Configure DPI Interface Color Coding (CFG_DPI_INTERFACE_COLOR_CODING)	32	RW	0000_0000h
20Ch	Configure DPI Pixel Format (CFG_DPI_PIXEL_FORMAT)	32	RW	0000_0000h
210h	Configure DPI vsync Polarity (CFG_DPI_VSYNC_POLARITY)	32	RW	0000_0000h
214h	Configure DPI hsync Polarity (CFG_DPI_HSYNC_POLARITY)	32	RW	0000_0000h
218h	Configure DPI Video Mode (CFG_DPI_VIDEO_MODE)	32	RW	0000_0000h
21Ch	Configure DPI horizontal front porch (CFG_DPI_HFP)	32	RW	0000_0000h
220h	Configure DPI Horizontal Back Porch (CFG_DPI_HBP)	32	RW	0000_0000h
224h	Configure DPI Horizontal Sync Width (CFG_DPI_HSA)	32	RW	0000_0000h
228h	Enable Multiple Packets (CFG_DPI_ENABLE_MULT_PKTS)	32	RW	0000_0000h
22Ch	Configure DPI Vertical Back Porch (CFG_DPI_VBP)	32	RW	0000_0000h
230h	Configure DPI Vertical Front Porch (CFG_DPI_VFP)	32	RW	0000_0000h
234h	Configure DPI BLLP Mode (CFG_DPI_BLLP_MODE)	32	RW	0000_0000h
238h	Configure DPI Blank Packet in BLLP (CFG_DPI_USE_NULL_PKT_BLLP)	32	RW	0000_0000h
23Ch	Configure DPI Vertical Active (CFG_DPI_VACTIVE)	32	RW	0000_0000h
240h	Configure DPI Virtual Channel (CFG_DPI_VC)	32	RW	0000_0000h
280h	Transmit Payload (TX_PAYLOAD)	32	RW	0000_0000h
284h	Packet Control (PKT_CONTROL)	32	RW	0000_0000h
288h	Send Packet (SEND_PACKET)	32	RW	0000_0000h
28Ch	Packet Status (PKT_STATUS)	32	R	0000_0000h
290h	Packet FIFO Write (PKT_FIFO_WR_LEVEL)	32	R	0000_0000h
294h	Packet FIFO Read (PKT_FIFO_RD_LEVEL)	32	R	0000_0000h
298h	Packet Rx Payload (PKT_RX_PAYLOAD)	32	R	0000_0000h
29Ch	Packet Rx Packet Header (PKT_RX_PKT_HEADER)	32	R	0000_0000h
2A0h	Interrupt Status (IRQ_STATUS)	32	R	0000_0000h
2A4h	Interrupt Status 2 (IRQ_STATUS2)	32	R	0000_0000h
2A8h	Mask Interrupt (IRQ_MASK)	32	RW	0000_0000h
2ACh	Interrupt Mask 2 (IRQ_MASK2)	32	RW	0000_0000h
300h	DPHY Power Down (PD_DPHY)	32	RW	0000_0001h
304h	Program T_HS_PREPARE (M_PRG_HS_PREPARE)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
308h	Program HS T_CLK_PREPARE (MC_PRG_HS_PREPARE)	32	RW	0000_0000h
30Ch	Program T_HS_ZERO (M_PRG_HS_ZERO)	32	RW	0000_0000h
310h	Program T_CLK_ZERO (MC_PRG_HS_ZERO)	32	RW	0000_0000h
314h	Program T_HS_TRAIL (M_PRG_HS_TRAIL)	32	RW	0000_0000h
318h	Program T_CLK_TRAIL (MC_PRG_HS_TRAIL)	32	RW	0000_0000h
31Ch	DPHY TST Input (TST)	32	RW	0000_0025h
320h	RTERM Select (RTERM_SEL)	32	RW	0000_0000h
324h	Power Down Auto Enable (AUTO_PD_EN)	32	RW	0000_0000h
328h	DPHY RXLPRP Input (RXLPRP)	32	RW	0000_0000h
32Ch	DPHY RXCDRP Input (RXCDRP)	32	RW	0000_0000h

48.7.1.2 Configure Number of Lanes (CFG_NUM_LANES)

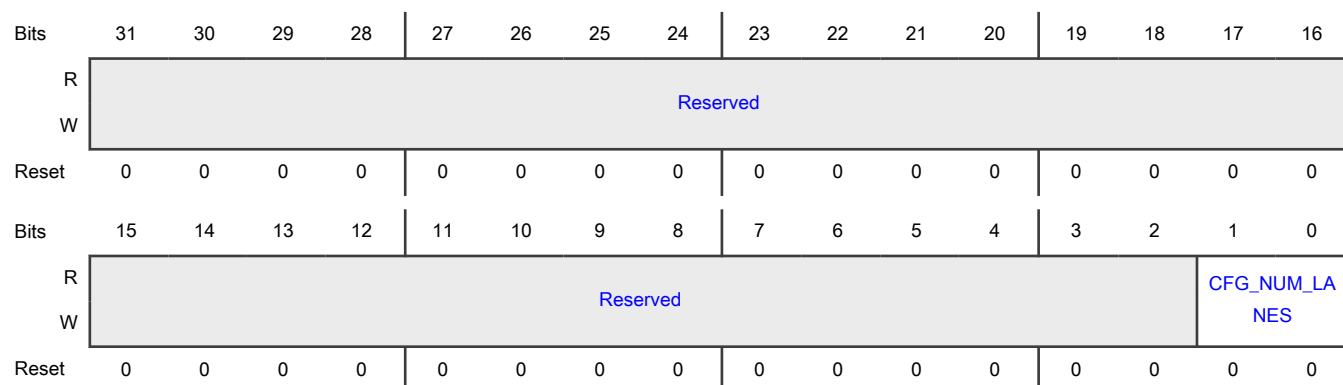
Offset

Register	Offset
CFG_NUM_LANES	0h

Function

Sets the number of active lanes used for transmitting data.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 CFG_NUM_LANES	Configure Number of Lanes Sets the number of active lanes used for transmitting data. 00b - 1 Lane 01b - 2 Lanes 10b - 3 Lanes 11b - 4 Lanes

48.7.1.3 Configure Non-continuous Clock (CFG_NONCONTINUOUS_CLK)**Offset**

Register	Offset
CFG_NONCONTINUOUS_CLK	4h

Function

Sets the Host Controller into non-continuous MIPI clock mode.

Diagram

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

Fields

Field	Function
31-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
0	Configure Non-continuous Clock
CFG_NONCONTINUOUS_CLK	When in non-continuous clock mode, the high speed clock will transition into low power mode between transmissions. 0b - Continuous high speed clock 1b - Non-Continuous high speed clock

48.7.1.4 Configure Pre Clock Periods (CFG_T_PRE)

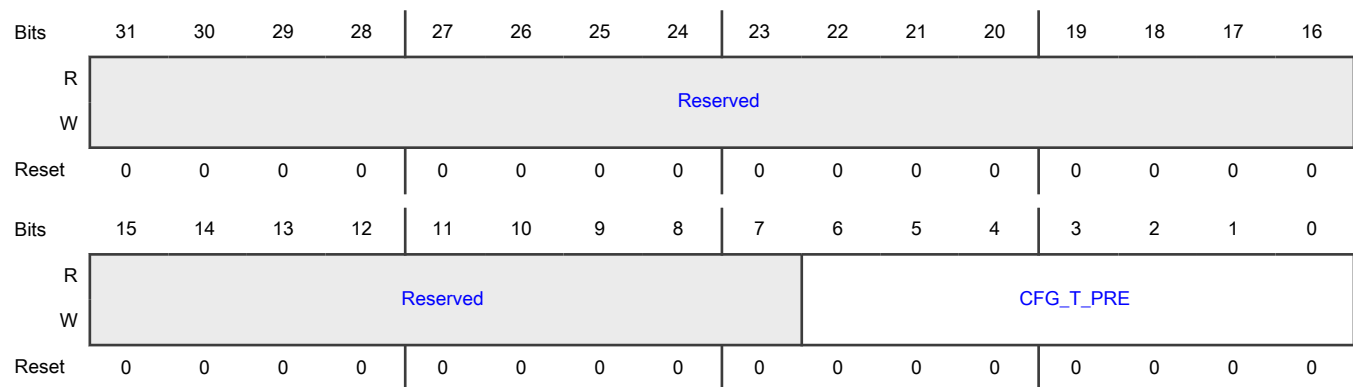
Offset

Register	Offset
CFG_T_PRE	8h

Function

Sets the number of byte clock periods pre clock.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 CFG_T_PRE	Configure Pre Clock Periods Sets the number of byte clock periods ('clk_byte' input) that the controller will wait after enabling the clock lane for HS operation before enabling the data lanes for HS operation.

48.7.1.5 Configure Post Clock Periods (CFG_T_POST)

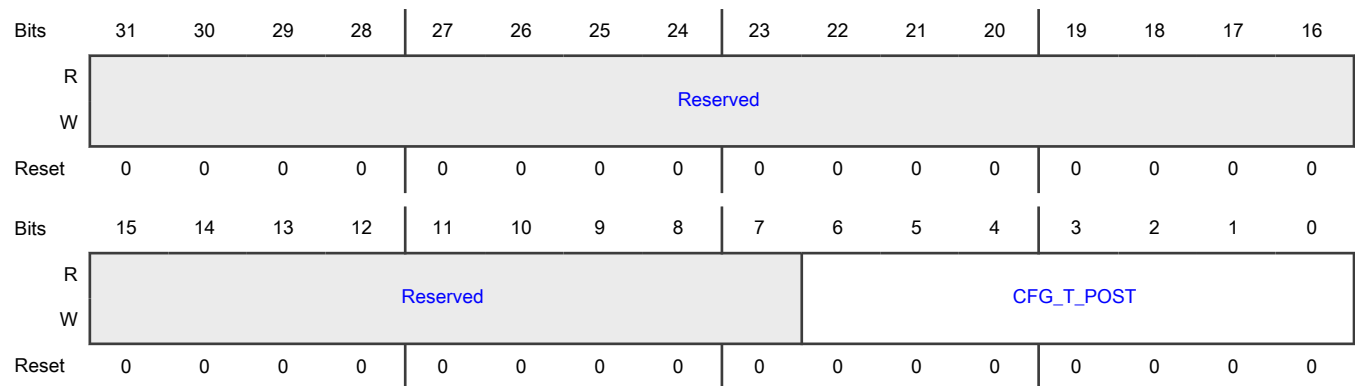
Offset

Register	Offset
CFG_T_POST	Ch

Function

Sets the number of byte clock periods post clock.

Diagram



Fields

Field	Function
31-7 —	Reserved
6-0 CFG_T_POST	Configure Post Clock Periods Sets the number of byte clock periods ('clk_byte' input) to wait before putting the clock lane into LP mode after the data lanes have been detected to be in Stop State.

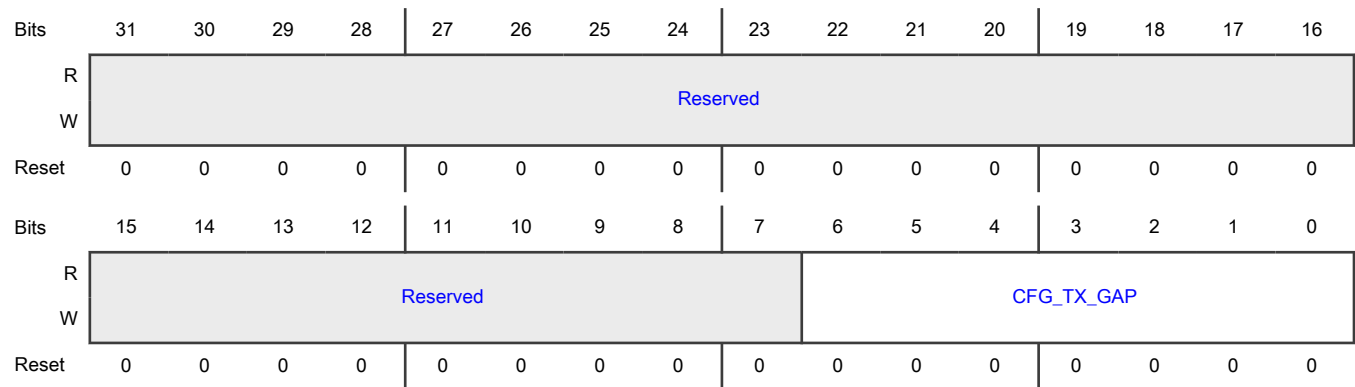
48.7.1.6 Configure Gap Clock Periods (CFG_TX_GAP)

Offset

Register	Offset
CFG_TX_GAP	10h

Function

Sets the number of byte clock periods ('clk_byte' input) that the controller will wait after the clock lane has been put into LP mode before enabling the clock lane for HS mode again.

Diagram**Fields**

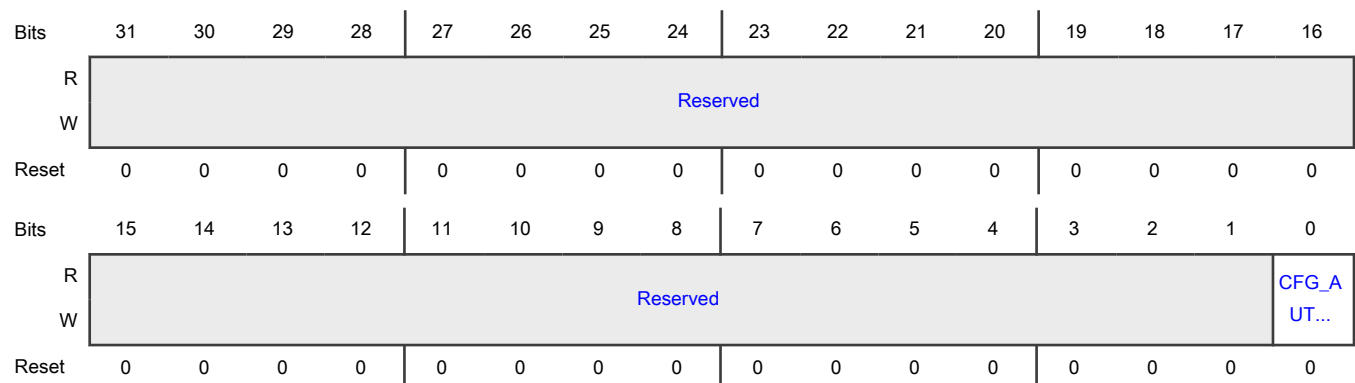
Field	Function
31-7 —	Reserved
6-0 CFG_TX_GAP	Configure Gap Clock Periods This setting represents the THS-EXIT DPHY timing parameter. The minimum value for this port is 1.

48.7.1.7 Configure Autoinsert EOTP (CFG_AUTOINSERT_EOTP)**Offset**

Register	Offset
CFG_AUTOINSERT_EOTP	14h

Function

Enables the Host Controller to automatically insert an EOTP short packet when switching from HS to LP mode.

Diagram

Fields

Field	Function
31-1 —	Reserved
0 CFG_AUTOINSERT_EOTP	Configure Autoinsert EOTP 0b - EOTP is not automatically inserted 1b - EOTP is automatically inserted

48.7.1.8 Configure Extra Commands after EOTP (CFG_EXTRA_CMDS_AFTER_EOTP)

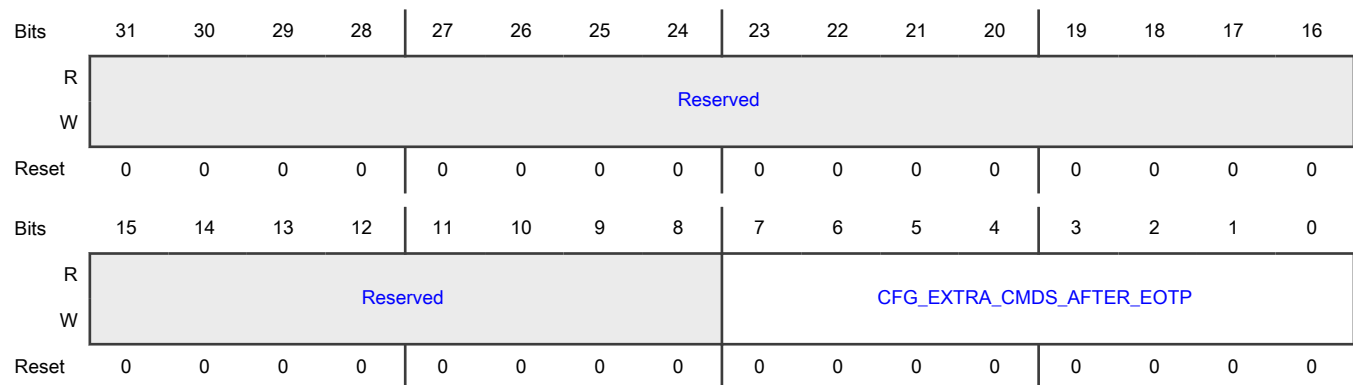
Offset

Register	Offset
CFG_EXTRA_CMDS_AFTER_EOTP	18h

Function

Configures the DSI Host Controller to send extra End Of Transmission Packets after the end of a packet.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 CFG_EXTRA_CMDS_AFTER_EOTP	Configure Extra Commands after EOTP The value is the number of extra EOTP packets sent.

48.7.1.9 Configure High Speed Tx Timeout Count (CFG_HTX_TO_COUNT)

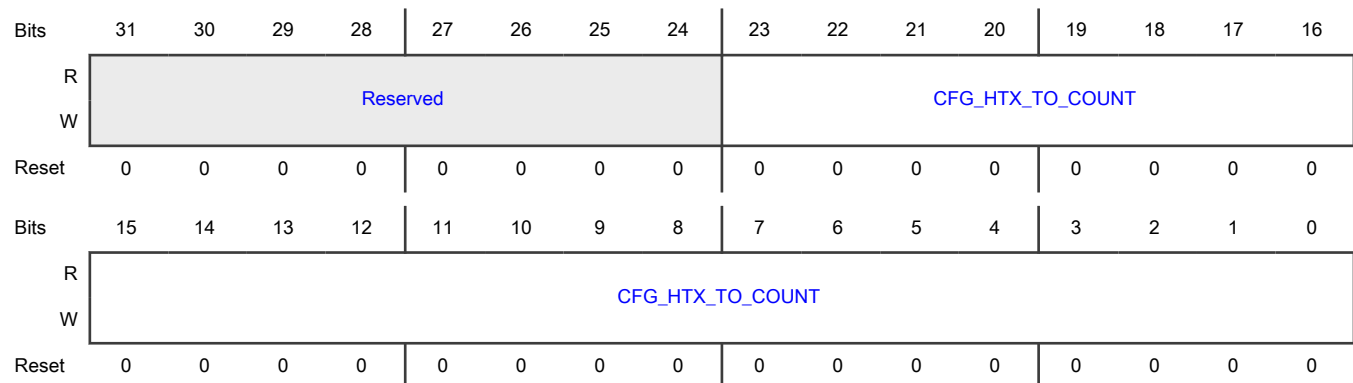
Offset

Register	Offset
CFG_HTX_TO_COUNT	1Ch

Function

Sets the value of the DSI Host High Speed TX timeout count.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 CFG_HTX_TO_COUNT	Configure High Speed Tx Timeout Count Sets the value of the DSI Host High Speed TX timeout count in clk_byte clock periods that once reached will initiate a timeout error and follow the recovery procedure documented in the DSI specification.

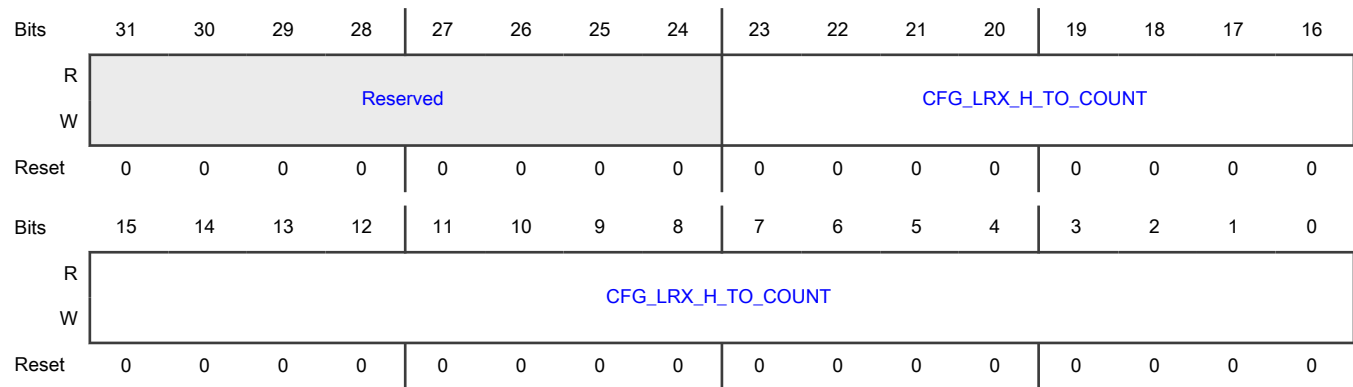
48.7.1.10 Configure Low Power Rx Timeout Count (CFG_LRX_H_TO_COUNT)

Offset

Register	Offset
CFG_LRX_H_TO_COUNT	20h

Function

Sets the value of the DSI Host low power Rx timeout count.

Diagram**Fields**

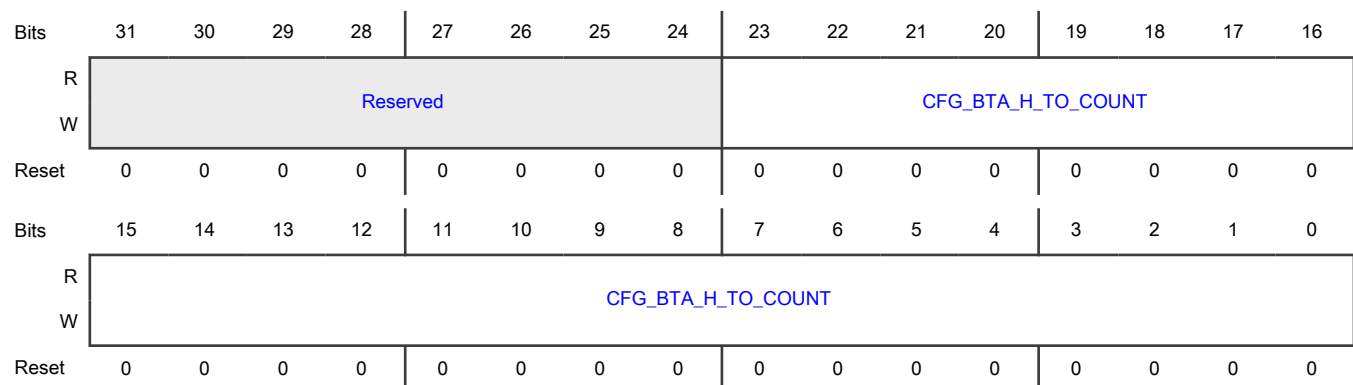
Field	Function
31-24 —	Reserved
23-0 CFG_LRX_H_TO_COUNT	Configure Low Power Rx Timeout Count Sets the value of the DSI Host low power Rx timeout count in clk_byte clock periods that once reached will initiate a timeout error and follow the recovery procedure documented in the DSI specification.

48.7.1.11 Configure Bus Turn Around Timeout Count (CFG_BTA_H_TO_COUNT)**Offset**

Register	Offset
CFG_BTA_H_TO_COUNT	24h

Function

Sets the value of the DSI Host Bus Turn Around (BTA) timeout.

Diagram

Fields

Field	Function
31-24 —	Reserved
23-0 CFG_BTA_H_T O_COUNT	Configure Bus Turn Around Timeout Count Sets the value of the DSI Host Bus Turn Around (BTA) timeout in clk_byte clock periods that once reached will initiate a timeout error.

48.7.1.12 Configure Twakeup (CFG_TWAKEUP)

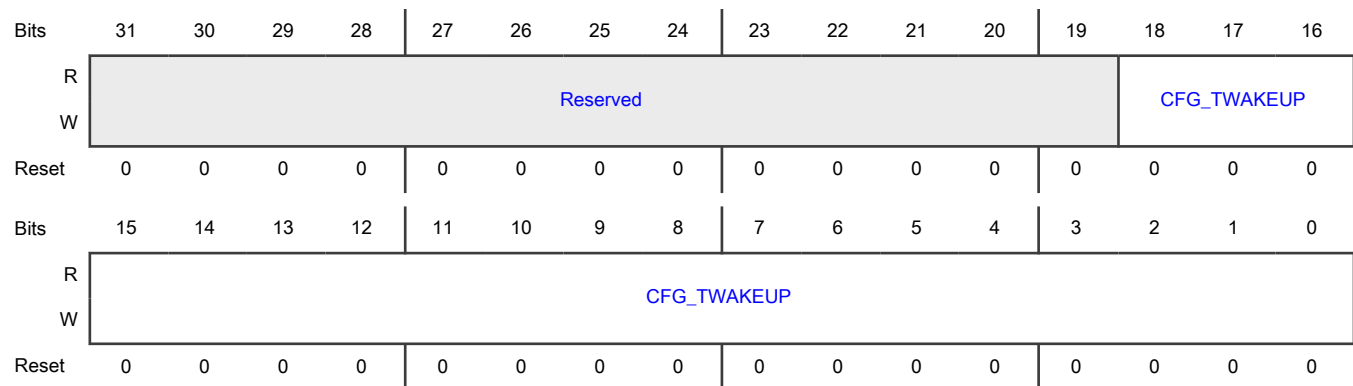
Offset

Register	Offset
CFG_TWAKEUP	28h

Function

Sets the number of clk_esc clock periods to keep a clock or data lane in Mark-1 state after exiting ULPS.

Diagram



Fields

Field	Function
31-19 —	Reserved
18-0 CFG_TWAKEU P	DPHY Twakeup Timing Parameter Sets the number of clk_esc clock periods to keep a clock or data lane in Mark-1 state after exiting ULPS. The MIPI DPHY spec requires a minimum of 1ms in Mark-1 state after leaving ULPS.

48.7.1.13 Configure Status Out (CFG_STATUS_OUT)

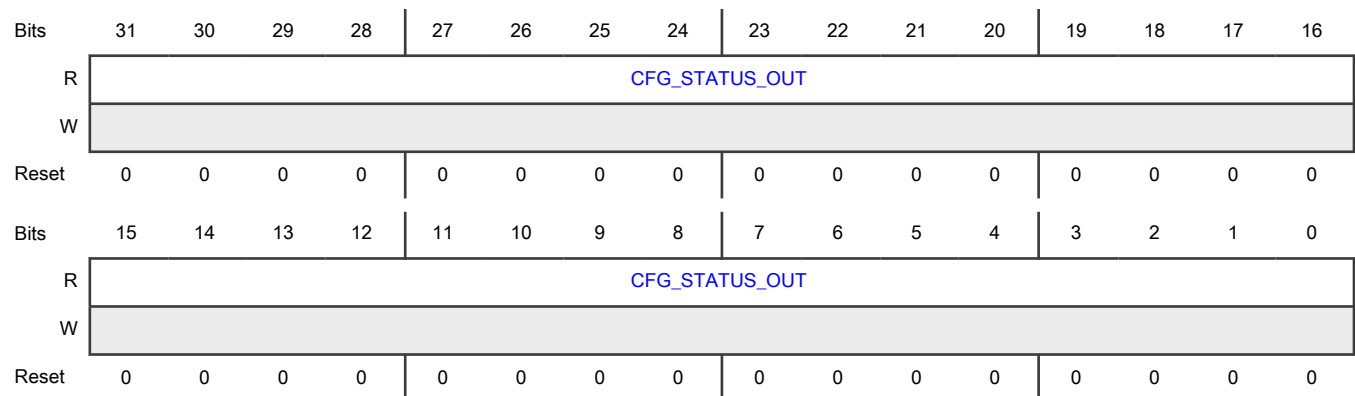
Offset

Register	Offset
CFG_STATUS_OUT	2Ch

Function

Sets status out.

Diagram



Fields

Field	Function
31-0 CFG_STATUS_OUT	Configure Status Out

48.7.1.14 Receive Error Status (RX_ERROR_STATUS)

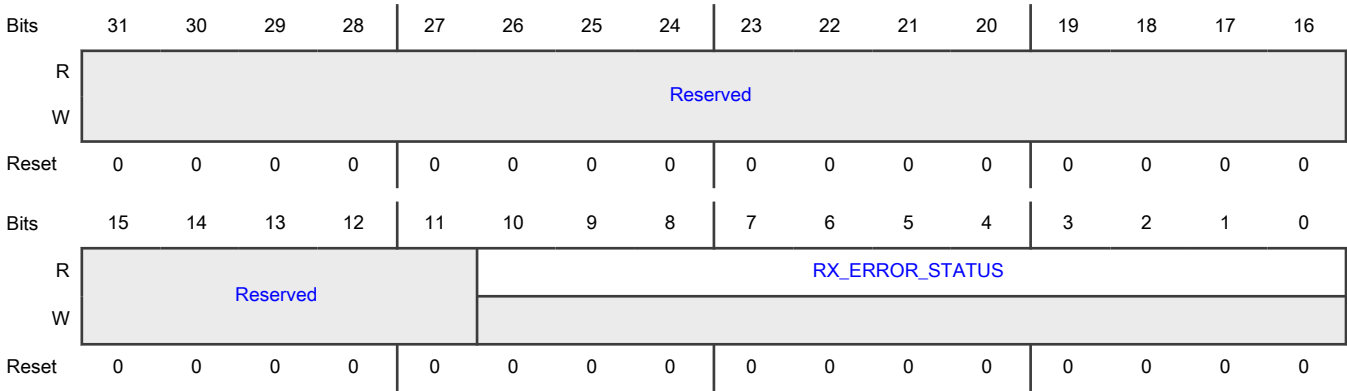
Offset

Register	Offset
RX_ERROR_STATUS	30h

Function

Status for Host receive error detection and timeout indicators.

Diagram



Fields

Field	Function
31-11 —	Reserved
10-0 RX_ERROR_S TATUS	Receive Error Status Status for Host receive error detection, ECC errors, CRC errors and for timeout indicators [0] ECC single bit error detected [1] ECC multi bit error detected [6:2] Errored bit position for single bit ECC error [7] CRC error detected [8] High Speed forward TX timeout detected [9] Reverse Low power data receive timeout detected [10] BTA timeout detected

48.7.1.15 Pixel Payload Size (CFG_DBI_PIXEL_PAYLOAD_SIZE)

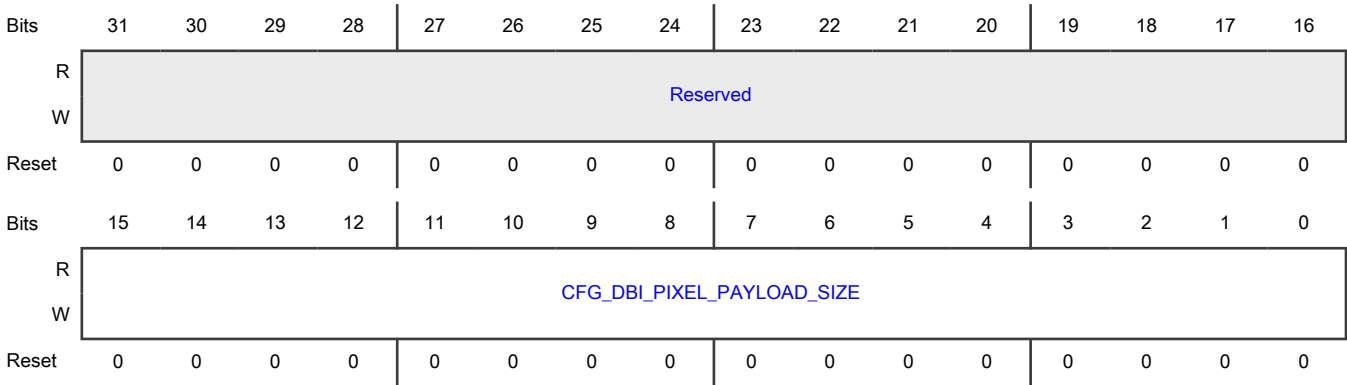
Offset

Register	Offset
CFG_DBI_PIXEL_PAYL OAD_SIZE	100h

Function

Maximum number of pixels that should be sent as one DSI packet.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DBI_PIXEL_PAYLOAD_SIZE	Pixel Payload Size Recommended to be evenly divisible by the line size (in pixels).

48.7.1.16 Configure DBI Pixel FIFO Send Level (CFG_DBI_PIXEL_FIFO_SEND_LEVEL)

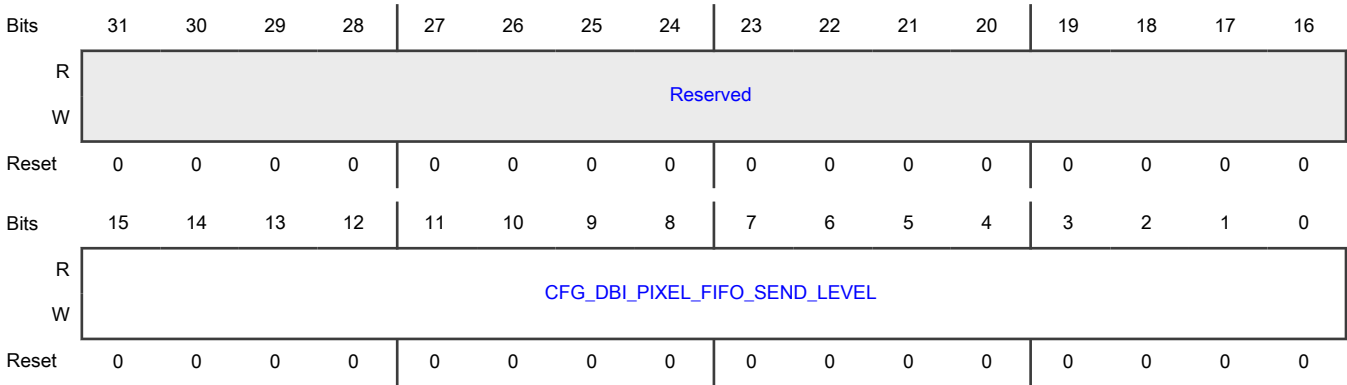
Offset

Register	Offset
CFG_DBI_PIXEL_FIFO_SEND_LEVEL	104h

Function

Controls the level at which the DBI Host bridge begins sending pixels.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DBI_PIXEL_FIFO_SEND_LEVEL	Configure DBI Pixel FIFO Send Level In order to optimize DSI utility, the DBI bridge buffers a certain number of DBI pixels before initiating a DSI packet.

48.7.1.17 Configure DPI Pixel Payload Size (CFG_DPI_PIXEL_PAYLOAD_SIZE)

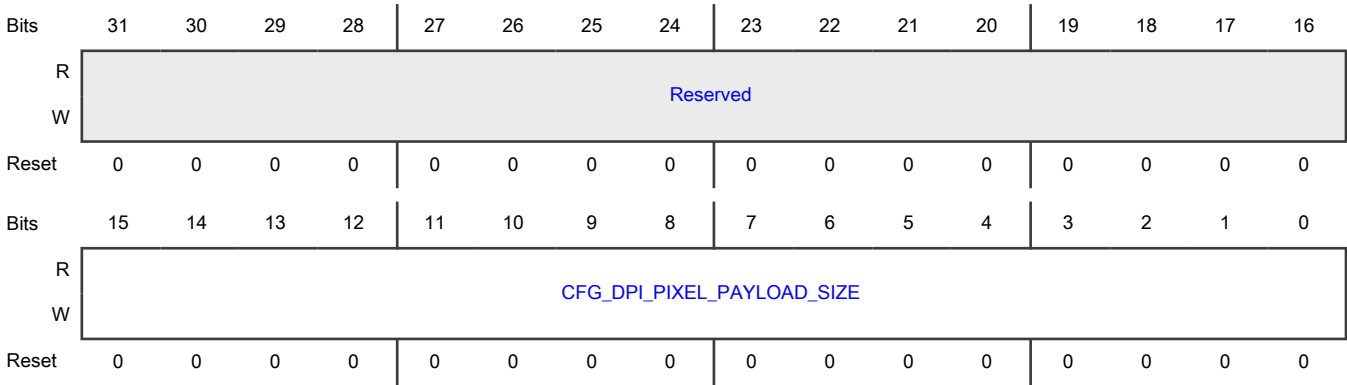
Offset

Register	Offset
CFG_DPI_PIXEL_PAYLOAD_SIZE	200h

Function

Maximum number of pixels that should be sent as one DSI packet.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DPI_PIXEL_PAYLOAD_SIZE	Configure DPI Pixel Payload Recommended to be evenly divisible by the line size (in pixels).

48.7.1.18 Configure DPI Pixel FIFO Send Level (CFG_DPI_PIXEL_FIFO_SEND_LEVEL)

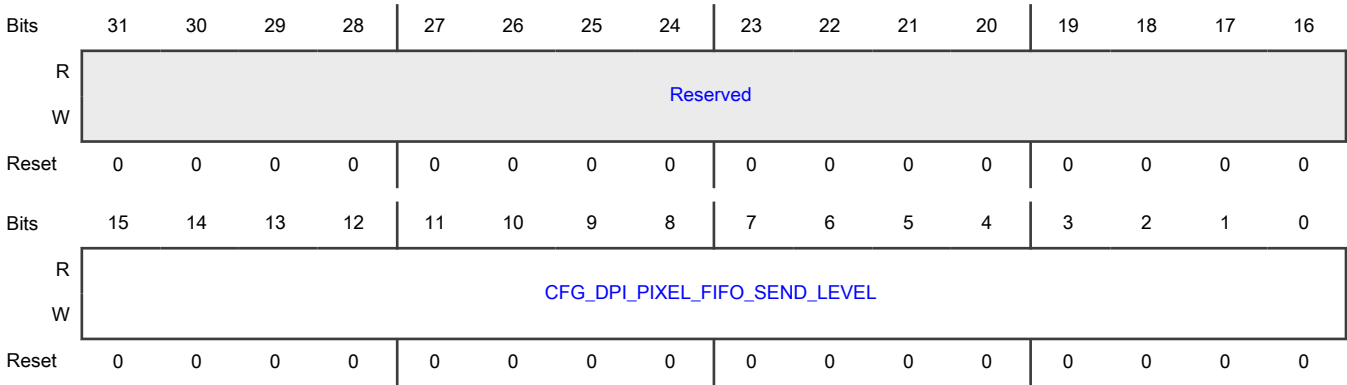
Offset

Register	Offset
CFG_DPI_PIXEL_FIFO_SEND_LEVEL	204h

Function

Controls the level at which the DPI Host bridge begins sending pixels.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DPI_PIXEL_FIFO_SEND_LEVEL	Configure DPI Pixel FIFO Send Level In order to optimize DSI utility, the DPI bridge buffers a certain number of DPI pixels before initiating a DSI packet.

48.7.1.19 Configure DPI Interface Color Coding (CFG_DPI_INTERFACE_COLOR_CODING)

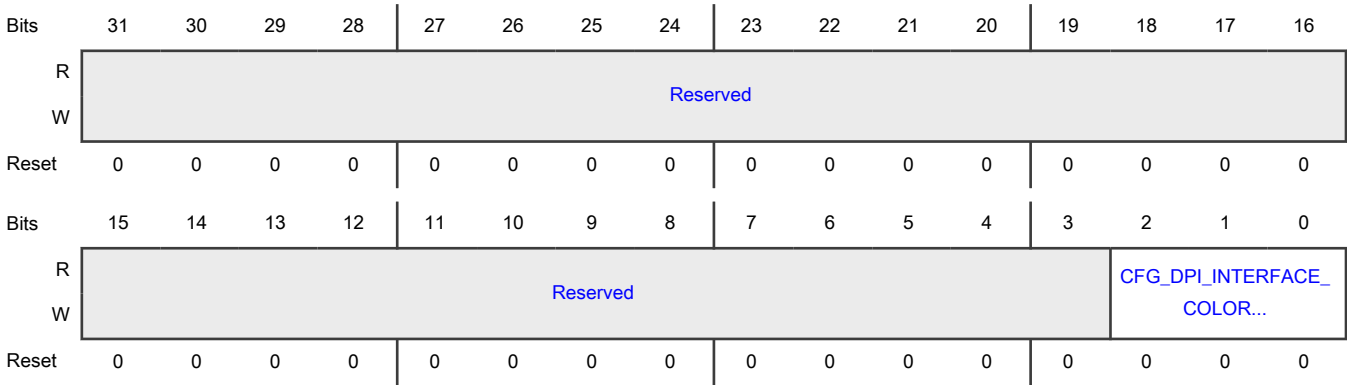
Offset

Register	Offset
CFG_DPI_INTERFACE_COLOR_CODING	208h

Function

Sets the distribution of RGB bits.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 CFG_DPI_INTE RFACE_COLO R_CODING	Configure DPI Interface Color Coding Sets the distribution of RGB bits within the 24-bit d bus, as specified by the DPI specification. 0= 16-bit Configuration 1 1= 16-bit Configuration 2 2= 16-bit Configuration 3 3= 18-bit Configuration 1 4= 18-bit Configuration 2 5= 24-bit

48.7.1.20 Configure DPI Pixel Format (CFG_DPI_PIXEL_FORMAT)

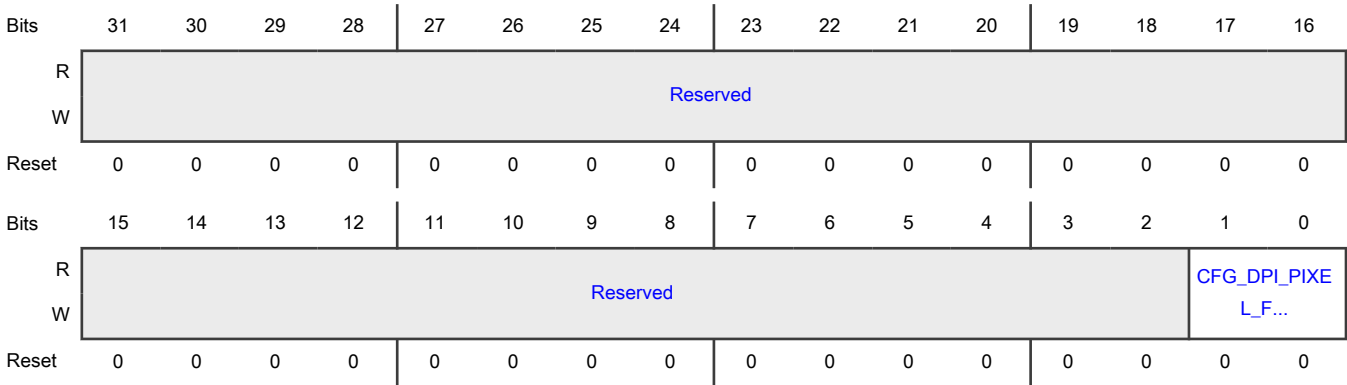
Offset

Register	Offset
CFG_DPI_PIXEL_FORM AT	20Ch

Function

Sets the DSI packet type of the pixels.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 CFG_DPI_PIXE L_FORMAT	Configure DPI Pixel Format Sets the DSI packet type of the pixels. 0 - 16 bit 1 - 18 bit 2 - 18 bit loosely packed 3 - 24 bit

48.7.1.21 Configure DPI vsync Polarity (CFG_DPI_VSYNC_POLARITY)

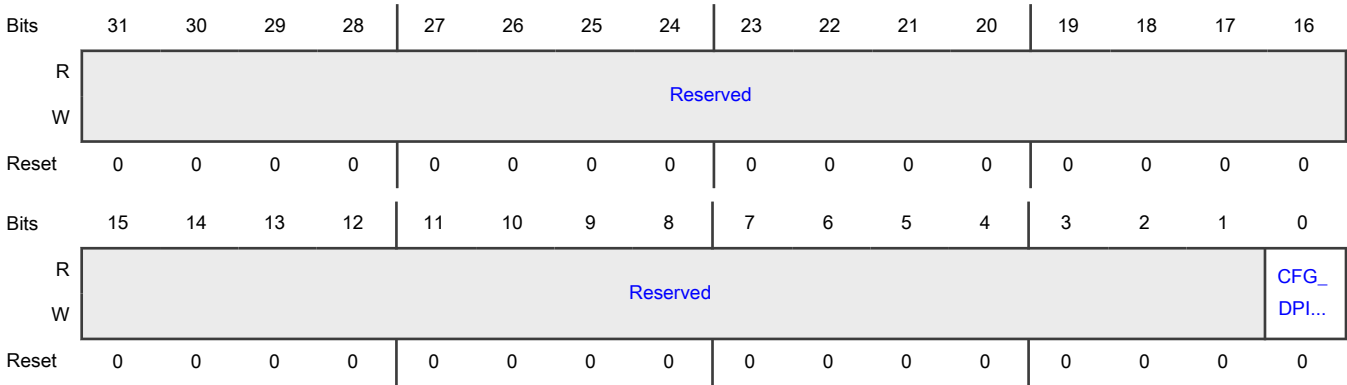
Offset

Register	Offset
CFG_DPI_VSYNC_POLARITY	210h

Function

Sets polarity of dpi_vsync_input

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CFG_DPI_VSY NC_POLARITY	Configure DPI vsync Polarity Sets polarity of dpi_vsync_input. 0 - active low 1 - active high

48.7.1.22 Configure DPI hsync Polarity (CFG_DPI_HSYNC_POLARITY)

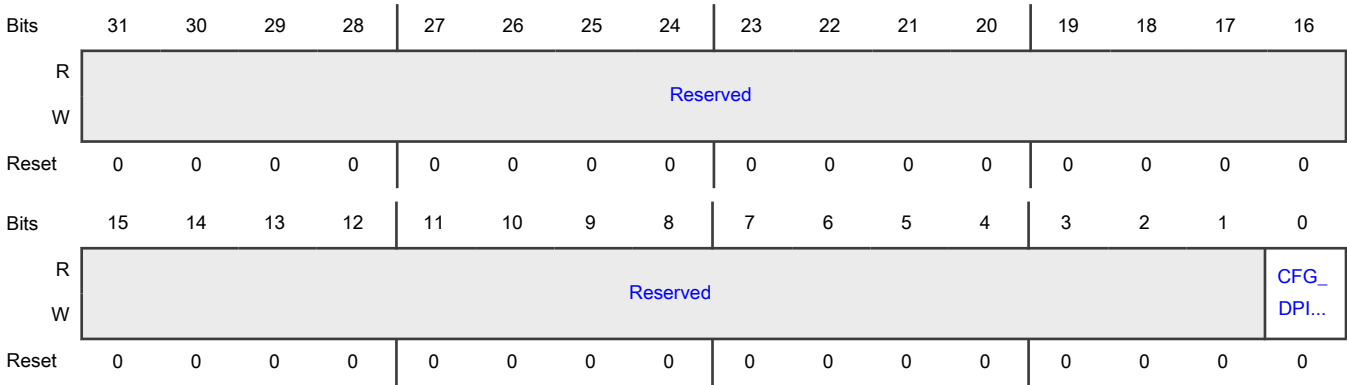
Offset

Register	Offset
CFG_DPI_HSYNC_POL ARITY	214h

Function

Sets polarity of dpi_hsync_input.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CFG_DPI_HSY NC_POLARITY	Configure DPI hsync Polarity Sets polarity of dpi_hsync_input. 0 - active low 1 - active high

48.7.1.23 Configure DPI Video Mode (CFG_DPI_VIDEO_MODE)

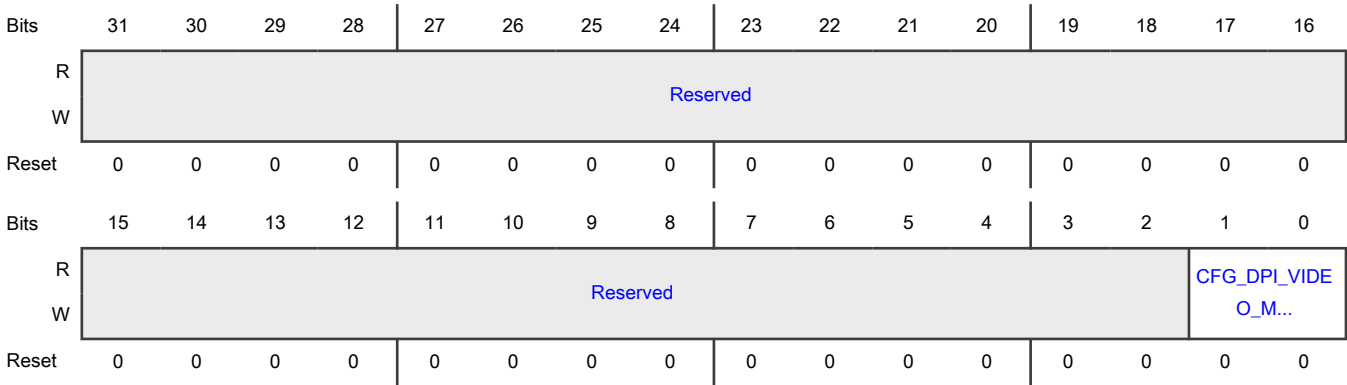
Offset

Register	Offset
CFG_DPI_VIDEO_MODE	218h

Function

Selects DSI video mode for the host DPI module generated packets.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 CFG_DPI_VIDE O_MODE	Configure DPI Video Mode Selects DSI video mode for the host DPI module generated packets. 00b - Non-Burst mode with Sync Pulses 01b - Non-Burst mode with Sync Events 10b - Burst mode 11b - Reserved, not valid

48.7.1.24 Configure DPI horizontal front porch (CFG_DPI_HFP)

Offset

Register	Offset
CFG_DPI_HFP	21Ch

Function

Sets the DSI packet payload size for the HFP blanking packet.

Diagram

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

Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DPI_HFP	Configure DPI Horizontal Front Porch Sets the DSI packet payload size, in bytes, of the horizontal front porch blanking packet.

48.7.1.25 Configure DPI Horizontal Back Porch (CFG_DPI_HBP)**Offset**

Register	Offset
CFG_DPI_HBP	220h

Function

Sets the DSI packet payload size of the HBP blanking packet.

Diagram

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

Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DPI_HBP	Configure DPI Horizontal Back Porch Sets the DSI packet payload size, in bytes, of the horizontal back porch blanking packet.

48.7.1.26 Configure DPI Horizontal Sync Width (CFG_DPI_HSA)**Offset**

Register	Offset
CFG_DPI_HSA	224h

Function

Sets the DSI packet payload size of the HSA blanking packet.

Diagram

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

Fields

Field	Function
31-16 —	Reserved
15-0 CFG_DPI_HSA	Configure DPI Horizontal Sync Width Sets the DSI packet payload size, in bytes, of the horizontal sync width filler blanking packet.

48.7.1.27 Enable Multiple Packets (CFG_DPI_ENABLE_MULT_PKTS)

Offset

Register	Offset
CFG_DPI_ENABLE_MULT_PKTS	228h

Function

Enables video line sent in two packets.

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 CFG_DPI_ENABLE_MULT_PKTS	Enable Multiple Packets Per Video Line. When enabled, <code>cfg_dpi_pixel_payload_size</code> must be set to exactly half the size of the video line. 0 - Video Line is sent in a single packet 1 - Video Line is sent in two packets

48.7.1.28 Configure DPI Vertical Back Porch (CFG_DPI_VBP)

Offset

Register	Offset
CFG_DPI_VBP	22Ch

Function

Sets the number of lines in the vertical back porch.

Diagram

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

Fields

Field	Function
31-8 —	Reserved
7-0 CFG_DPI_VBP	Configure DPI Vertical Back Porch

48.7.1.29 Configure DPI Vertical Front Porch (CFG_DPI_VFP)**Offset**

Register	Offset
CFG_DPI_VFP	230h

Function

Sets the number of lines in the vertical front porch.

Diagram

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

Fields

Field	Function
31-8 —	Reserved
7-0 CFG_DPI_VFP	Configure DPI VFP

48.7.1.30 Configure DPI BLLP Mode (CFG_DPI_BLLP_MODE)

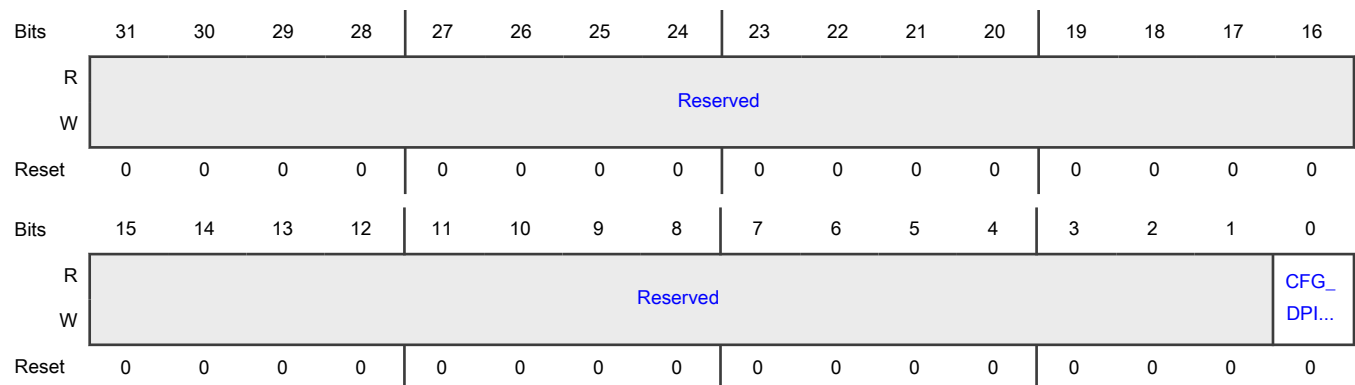
Offset

Register	Offset
CFG_DPI_BLLP_MODE	234h

Function

Used to optimize blp periods to Low Power when possible.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CFG_DPI_BLLP_MODE	Configure DPI BLLP Mode Optimize blp periods to Low Power mode when possible 0 - blanking packets are sent during BLLP periods 1 - LP mode is used for BLLP periods

48.7.1.31 Configure DPI Blank Packet in BLLP (CFG_DPI_USE_NULL_PKT_BLLP)

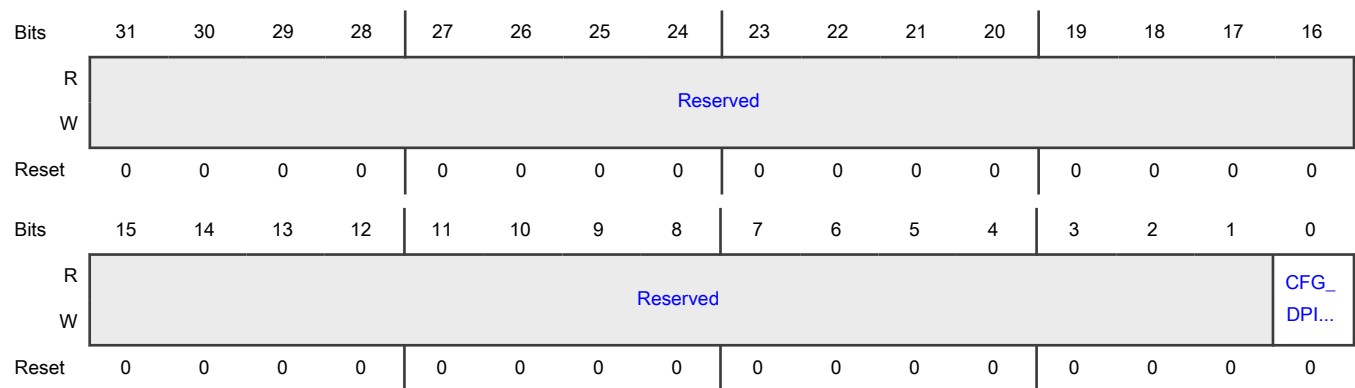
Offset

Register	Offset
CFG_DPI_USE_NULL_PKT_BLLP	238h

Function

Selects type of blanking packet to be sent during blp region.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 CFG_DPI_USE_NULL_PKT_BLLP	Configure DPI Blank Packet in BLLP Selects type of blanking packet to be sent during blp region. 0 - Blanking packet used in blp region 1 - Null packet used in blp region

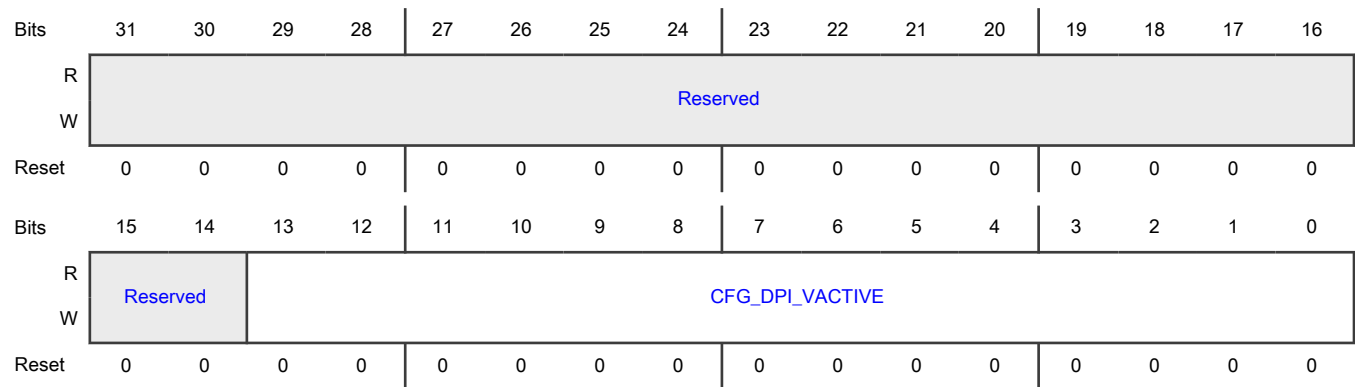
48.7.1.32 Configure DPI Vertical Active (CFG_DPI_VACTIVE)

Offset

Register	Offset
CFG_DPI_VACTIVE	23Ch

Function

Sets the number of lines in the Vertical Active.

Diagram**Fields**

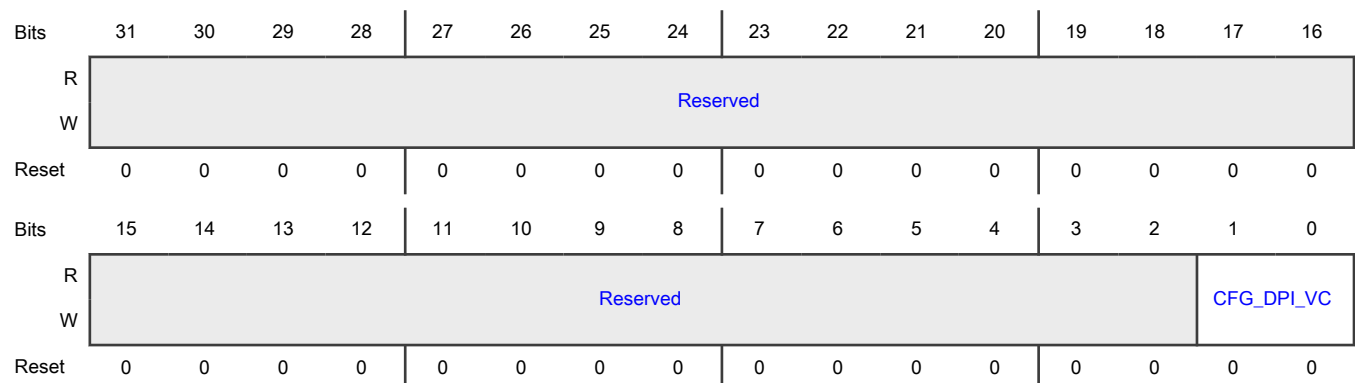
Field	Function
31-14 —	Reserved
13-0 CFG_DPI_VACTIVE	Configure DPI Vertical Active Sets the number of lines in the vertical active read.

48.7.1.33 Configure DPI Virtual Channel (CFG_DPI_VC)**Offset**

Register	Offset
CFG_DPI_VC	240h

Function

Sets the Virtual Channel (VC) of packets that will be sent to the receive packet interface.

Diagram

Fields

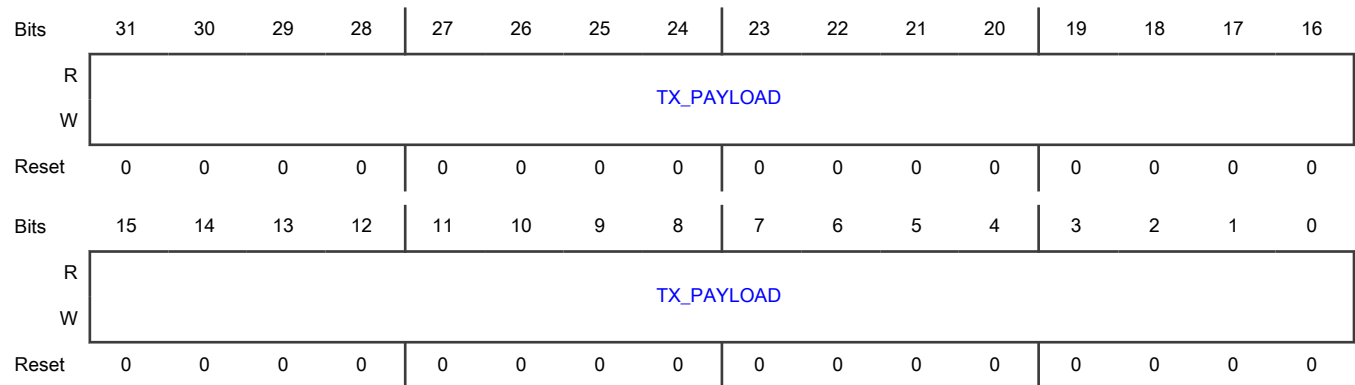
Field	Function
31-2 —	Reserved
1-0 CFG_DPI_VC	Configure DPI Virtual Channel Packets with VC not equal to this value are discarded and the "DSI VC ID Invalid" bit (bit 12) in the DSI error report is set.

48.7.1.34 Transmit Payload (TX_PAYLOAD)**Offset**

Register	Offset
TX_PAYLOAD	280h

Function

Writes to this registers load the payload fifo with 32 bit values.

Diagram**Fields**

Field	Function
31-0 TX_PAYLOAD	Tx Payload Data Write

48.7.1.35 Packet Control (PKT_CONTROL)

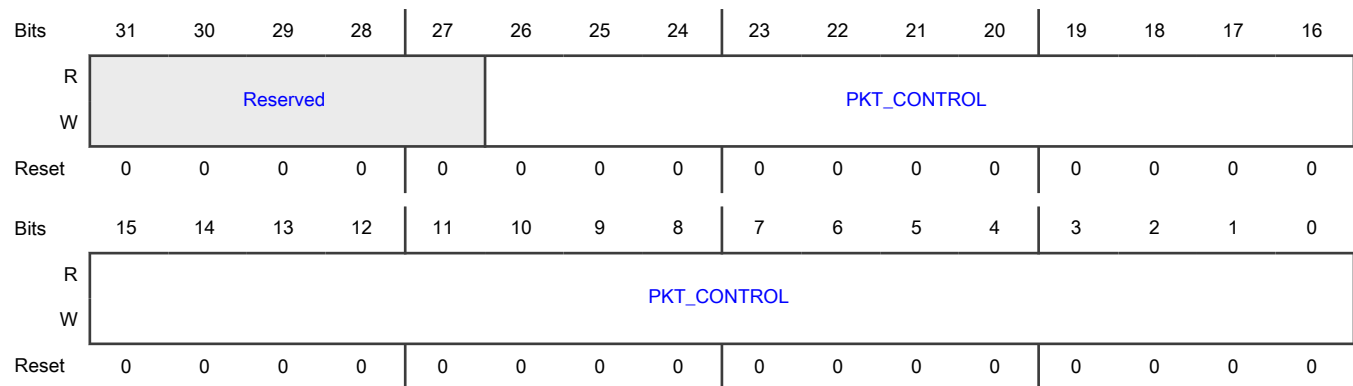
Offset

Register	Offset
PKT_CONTROL	284h

Function

Sets values for packet control.

Diagram



Fields

Field	Function
31-27 —	Reserved
26-0 PKT_CONTROL	Tx Packet Control [15:0] - Packet word count [17:16] - Packet Virtual Channel [23:18] - Packet Header DSI Data Type [24] - LP or HS select. 0 - LP mode, 1 - HS mode [25] - Perform BTA after packet is sent [26] - Perform BTA only, no packet tx

48.7.1.36 Send Packet (SEND_PACKET)

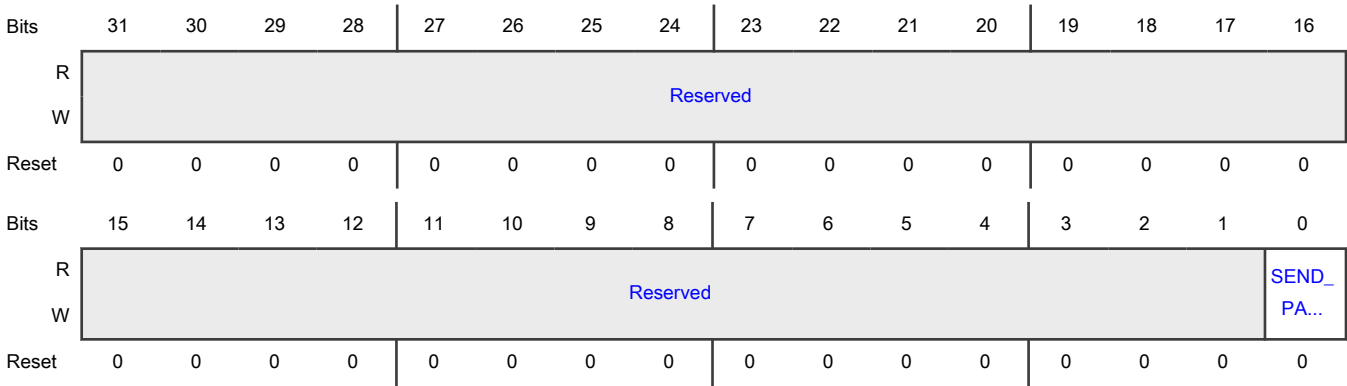
Offset

Register	Offset
SEND_PACKET	288h

Function

Tx send packet, writing to this register causes the packet described in dsi_host_pkt_control to be sent.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 SEND_PACKET	Send Packet

48.7.1.37 Packet Status (PKT_STATUS)

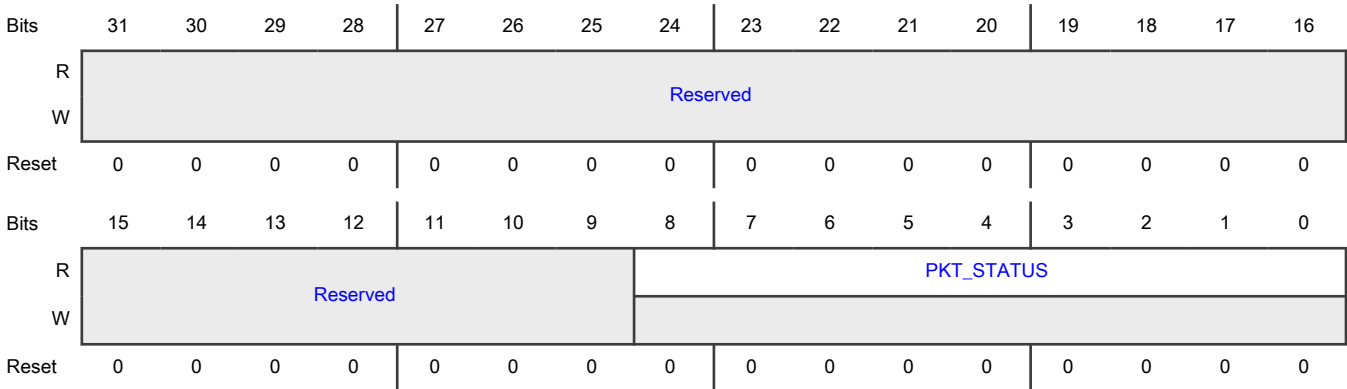
Offset

Register	Offset
PKT_STATUS	28Ch

Function

Status of APB to packet interface

Diagram



Fields

Field	Function
31-9 —	Reserved
8-0 PKT_STATUS	Packet Status [0] - state machine not idle [1] - Tx packet done [2] - dphy direction 0 - tx had control, 1 - rx has control [3] - tx fifo overflow [4] - tx fifo underflow [5] - rx fifo overflow [6] - rx fifo underflow [7] - rx packet header has been received [8] - all rx packet payload data has been received

48.7.1.38 Packet FIFO Write (PKT_FIFO_WR_LEVEL)

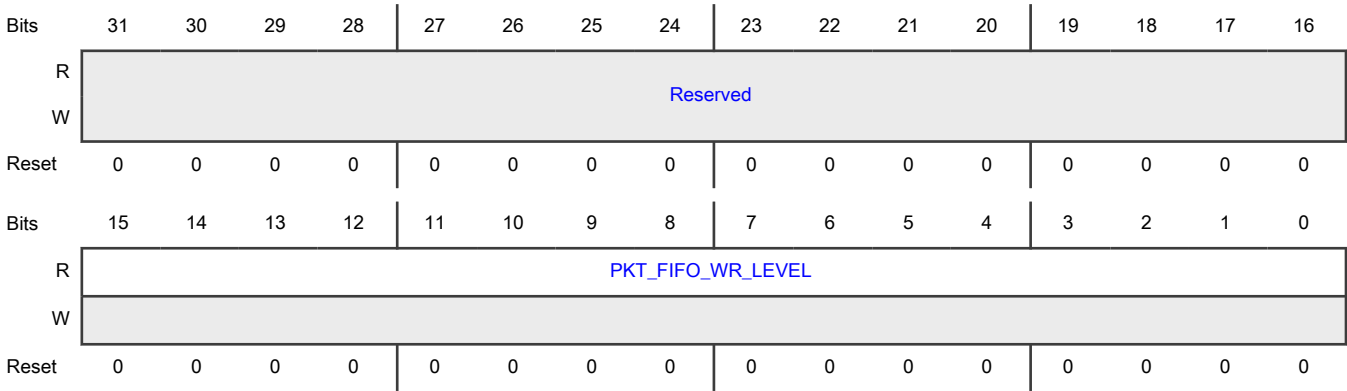
Offset

Register	Offset
PKT_FIFO_WR_LEVEL	290h

Function

Write level of APB to pkt interface fifo.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 PKT_FIFO_WR_LEVEL	Packet FIFO Write Level

48.7.1.39 Packet FIFO Read (PKT_FIFO_RD_LEVEL)

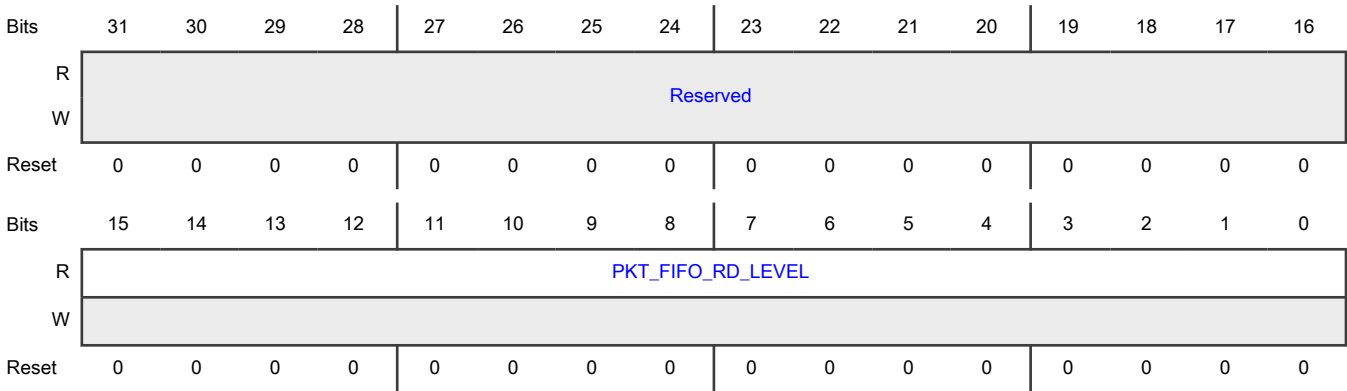
Offset

Register	Offset
PKT_FIFO_RD_LEVEL	294h

Function

Read level of APB to pkt interface fifo.

Diagram



Fields

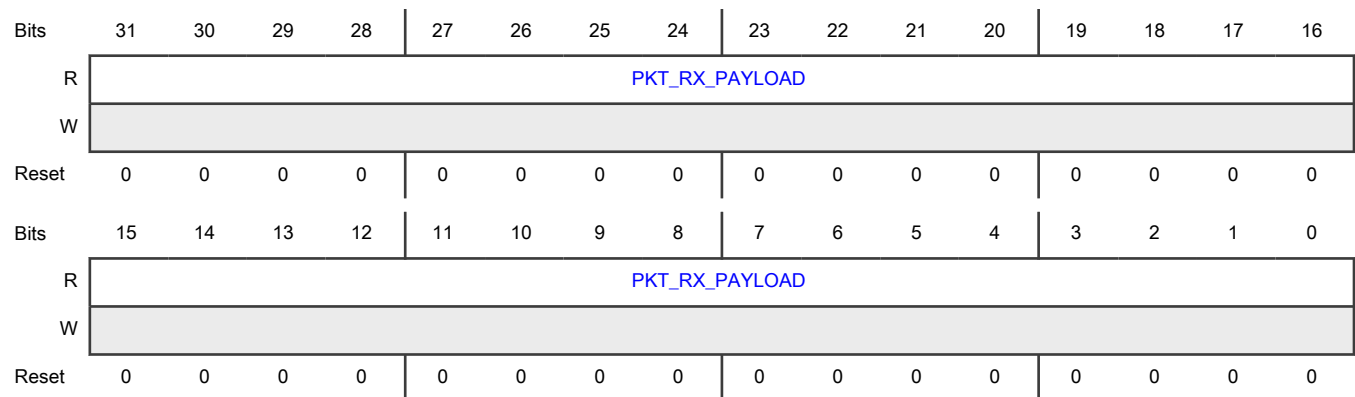
Field	Function
31-16 —	Reserved
15-0 PKT_FIFO_RD_LEVEL	Packet FIFO Read Level

48.7.1.40 Packet Rx Payload (PKT_RX_PAYLOAD)**Offset**

Register	Offset
PKT_RX_PAYLOAD	298h

Function

APB to pkt interface rx payload read.

Diagram**Fields**

Field	Function
31-0 PKT_RX_PAYL OAD	Packet Rx Payload

48.7.1.41 Packet Rx Packet Header (PKT_RX_PKT_HEADER)

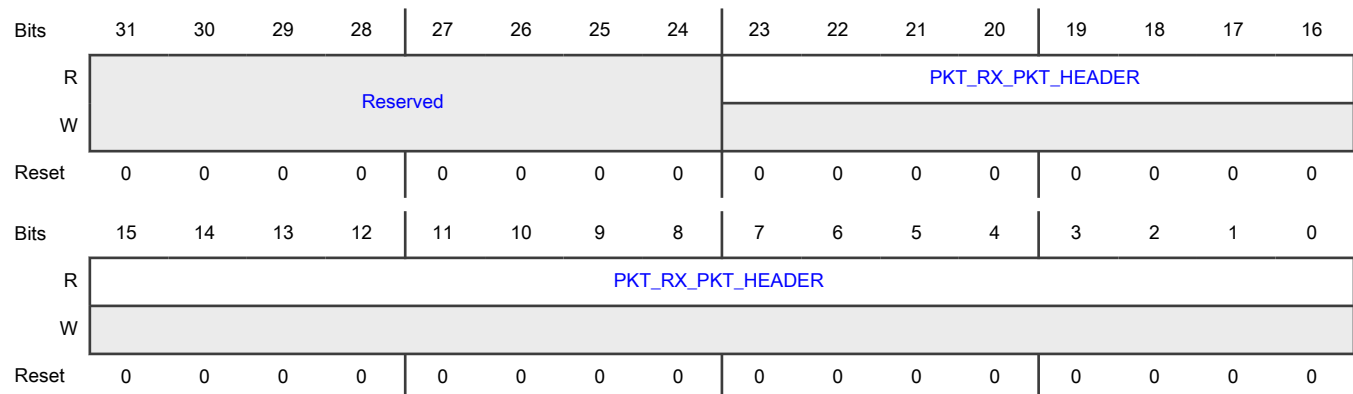
Offset

Register	Offset
PKT_RX_PKT_HEADER	29Ch

Function

APB to pkt interface rx packet header

Diagram



Fields

Field	Function
31-24 —	Reserved
23-0 PKT_RX_PKT_HEADER	Packet Rx Packet Header APB to pkt interface rx packet header [15:0] word count [21:16] data type [23:22] Virtual Channel

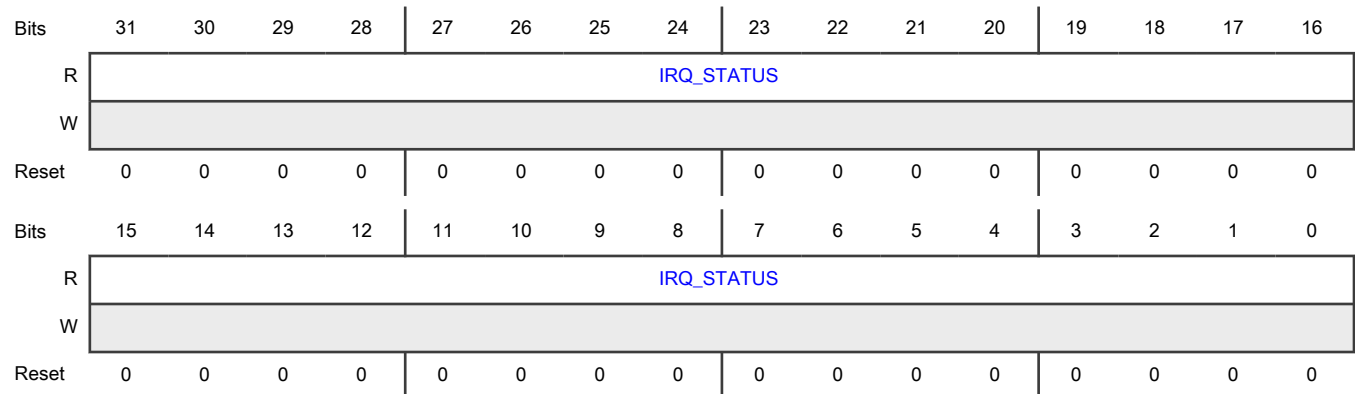
48.7.1.42 Interrupt Status (IRQ_STATUS)

Offset

Register	Offset
IRQ_STATUS	2A0h

Function

Status of APB to packet interface

Diagram**Fields**

Field	Function
31-0 IRQ_STATUS	Interrupt Status Status of APB to packet interface [0] - state machine not idle [1] - Tx packet done [2] - dphy direction 0 - tx had control, 1 - rx has control [3] - tx fifo overflow [4] - tx fifo underflow [5] - rx fifo overflow [6] - rx fifo underflow [7] - rx packet header has been received [8] - all rx packet payload data has been received [28:9] - map directory to dsi host controller status_out port bit descriptions [29] - high speed tx timeout, host controller hs_tx_timeout port [30] - low power rx timeout, host controller lp_rx_timeout port [31] - host bta timeout, host controller host_bta_timeout port

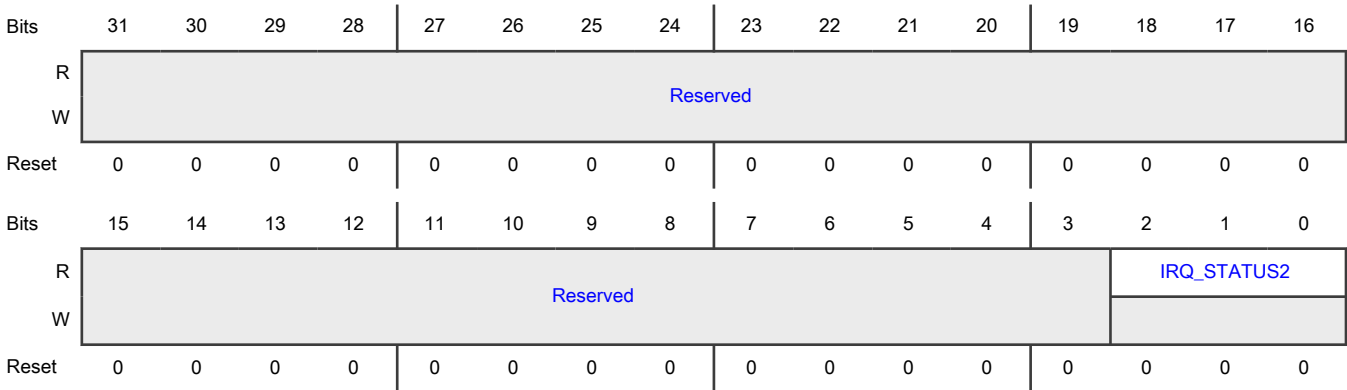
48.7.1.43 Interrupt Status 2 (IRQ_STATUS2)**Offset**

Register	Offset
IRQ_STATUS2	2A4h

Function

Status of APB to packet interface part 2, read part 2 first then dsi_host_irq_status.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 IRQ_STATUS2	Interrupt Status 2 Reading dsi_host_irq_status will clear both status and status2 [0] - single bit ecc error [1] - multi bit ecc error [2] - crc error

48.7.1.44 Mask Interrupt (IRQ_MASK)

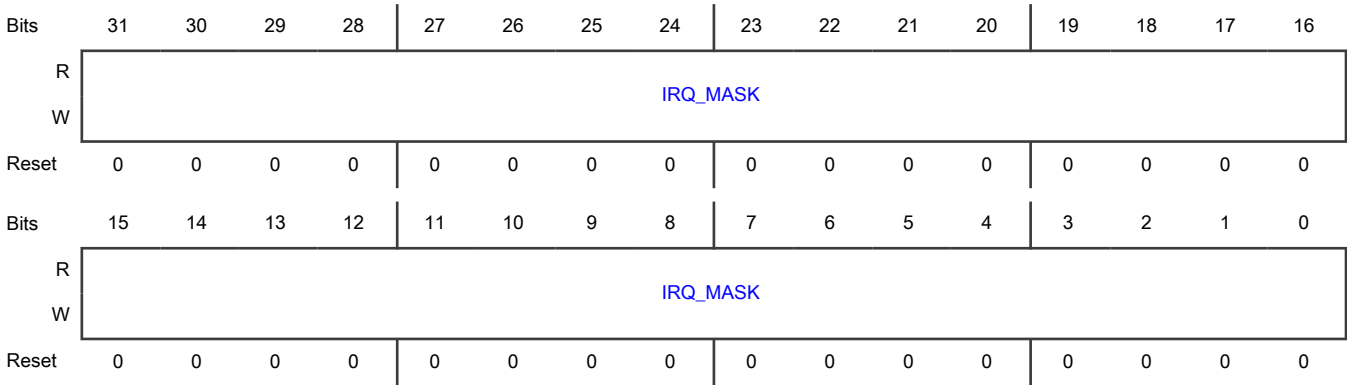
Offset

Register	Offset
IRQ_MASK	2A8h

Function

Contains various mask interrupts.

Diagram



Fields

Field	Function
31-0 IRQ_MASK	Mask Interrupt irq mask [0] - state machine not idle [1] - Tx packet done [2] - dphy direction 0 - tx had control, 1 - rx has control [3] - tx fifo overflow [4] - tx fifo underflow [5] - rx fifo overflow [6] - rx fifo underflow [7] - rx packet header has been received [8] - all rx packet payload data has been received [28:9] - map directory to dsi host controller status_out port bit descriptions [29] - high speed tx timeout, host controller hs_tx_timeout port [30] - low power rx timeout, host controller lp_rx_timeout port [31] - host bta timeout, host controller host_bta_timeout port

48.7.1.45 Interrupt Mask 2 (IRQ_MASK2)

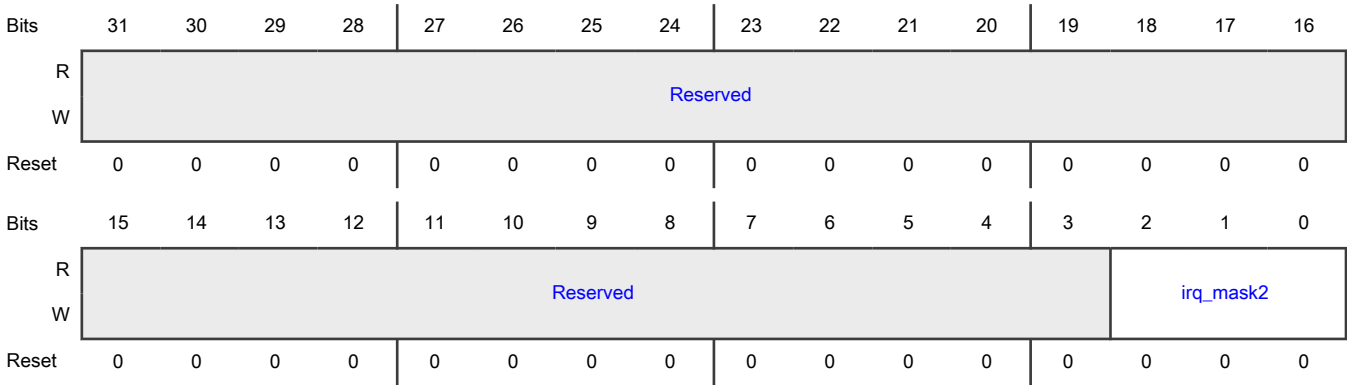
Offset

Register	Offset
IRQ_MASK2	2ACh

Function

Contains various mask interrupts.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 irq_mask2	Interrupt Mask 2 irq mask 2 [0] - single bit ecc error [1] - multi bit ecc error [2] - crc error

48.7.1.46 DPHY Power Down (PD_DPHY)

Offset

Register	Offset
PD_DPHY	300h

Function

Power Down Input for D-PHY

Diagram

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

Fields

Field	Function
31-1 —	Reserved
0 PD_DPHY	Power Down Input for D-PHY 0b - Power up 1b - Power down

48.7.1.47 Program T_HS_PREPARE (M_PRG_HS_PREPARE)**Offset**

Register	Offset
M_PRG_HS_PREPARE	304h

Function

Used to program T_HS_PREPARE time.

Diagram

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

Fields

Field	Function
31-2 —	Reserved
1-0 M_PRG_HS_P PREPARE	DPHY m_PRG_HS_PREPARE Input Bits used to program T_HS_PREPARE time in the beginning of high speed transmission mode. For more details on how to program this field, see DPHY Programming .

48.7.1.48 Program HS T_CLK_PREPARE (MC_PRG_HS_PREPARE)

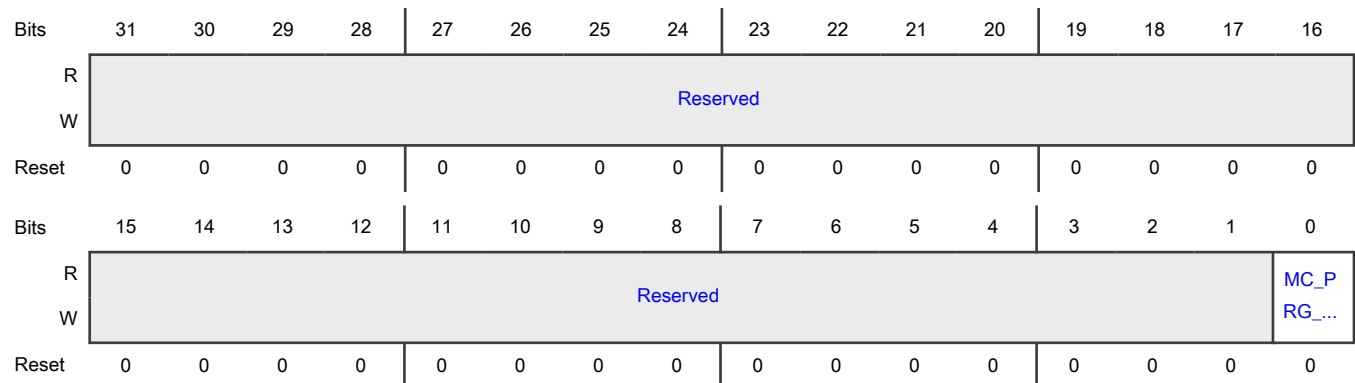
Offset

Register	Offset
MC_PRG_HS_PREPARE	308h

Function

Used to program T_CLK_PREPARE time.

Diagram



Fields

Field	Function
31-1 —	Reserved
0 MC_PRG_HS_ PREPARE	DPHY mc_PRG_HS_PREPARE input Bits used to program T_CLK_PREPARE time in the beginning of high speed transmission mode. For more details on how to program this field, see DPHY Programming .

48.7.1.49 Program T_HS_ZERO (M_PRG_HS_ZERO)

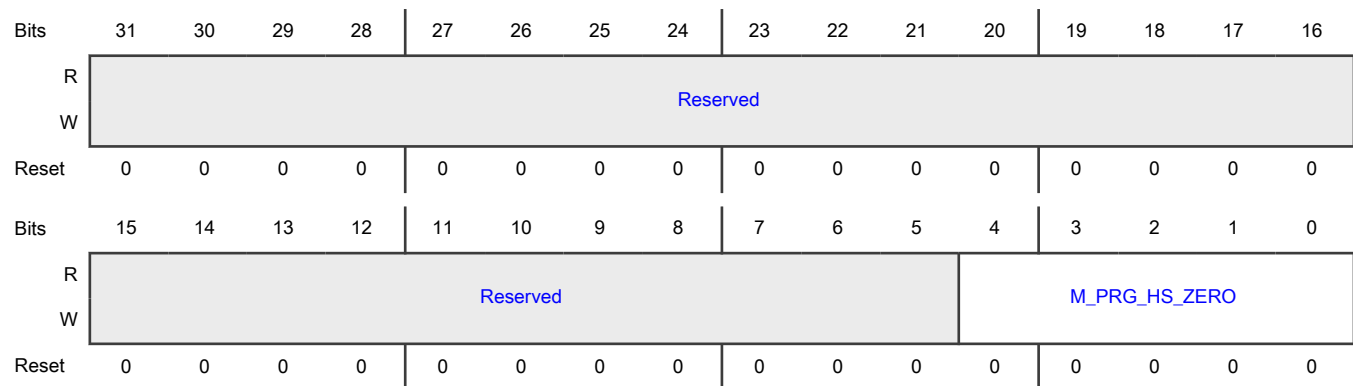
Offset

Register	Offset
M_PRG_HS_ZERO	30Ch

Function

Used to program T_HS_ZERO time.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 M_PRG_HS_ZERO	DPHY m_PRG_HS_ZERO Input Bits used to program T_HS_ZERO time in the beginning of high speed transmission mode. For more details on how to program this field, see DPHY Programming .

48.7.1.50 Program T_CLK_ZERO (MC_PRG_HS_ZERO)

Offset

Register	Offset
MC_PRG_HS_ZERO	310h

Function

Used to program T_CLK_ZERO time.

Diagram

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

Fields

Field	Function
31-6 —	Reserved
5-0 MC_PRG_HS_ZERO	DPHY mc_PRG_HS_ZERO Input Bits used to program T_CLK_ZERO time in the beginning of high speed transmission mode. For more details on how to program this field, see DPHY Programming .

48.7.1.51 Program T_HS_TRAIL (M_PRG_HS_TRAIL)**Offset**

Register	Offset
M_PRG_HS_TRAIL	314h

Function

Used to program T_HS_TRAIL time.

Diagram

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

Fields

Field	Function
31-4 —	Reserved
3-0 M_PRG_HS_TRAIL	DPHY m_PRG_HS_TRAIL Input Bits used to program T_HS_TRAIL time in the end of high speed transmission mode. For more details on how to program this field, see DPHY Programming .

48.7.1.52 Program T_CLK_TRAIL (MC_PRG_HS_TRAIL)

Offset

Register	Offset
MC_PRG_HS_TRAIL	318h

Function

Used to program T_CLK_TRAIL time.

Diagram

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

Fields

Field	Function
31-4 —	Reserved
3-0 MC_PRG_HS_TRAIL	DPHY mc_PRG_HS_TRAIL input Bits used to program T_CLK_TRAIL time in the end of high speed transmission mode.

48.7.1.53 DPHY TST Input (TST)

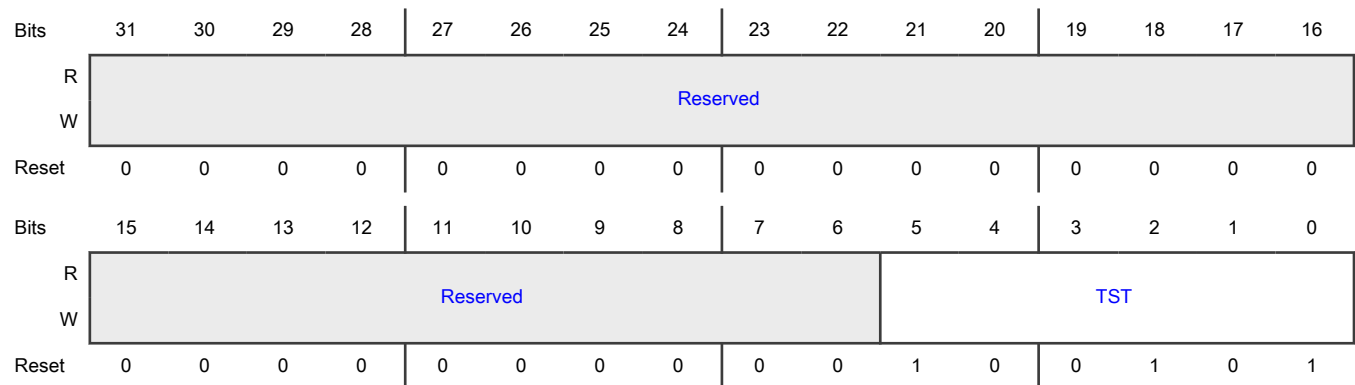
Offset

Register	Offset
TST	31Ch

Function

Set TST[5:0]=[100101] at normal operation.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-0 TST	DPHY TST Input

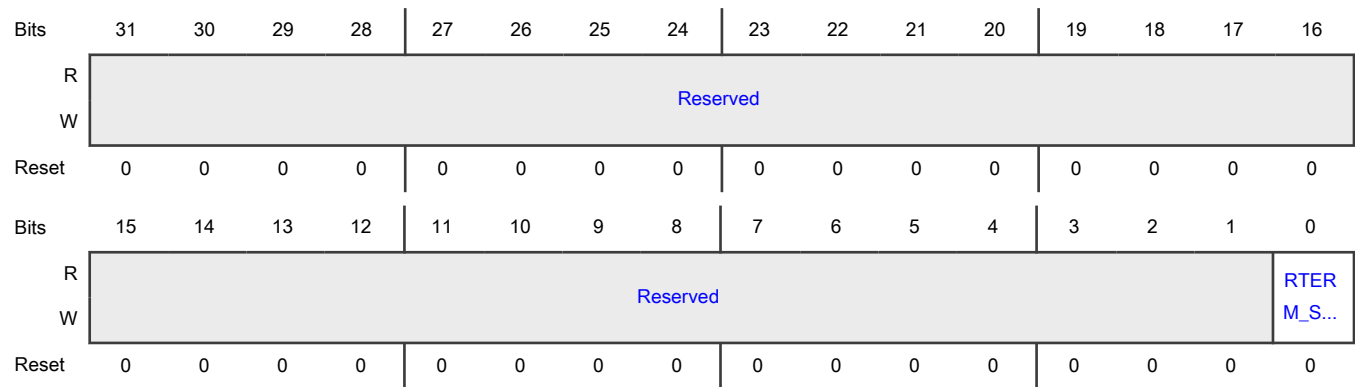
48.7.1.54 RTERM Select (RTERM_SEL)

Offset

Register	Offset
RTERM_SEL	320h

Function

RTERM_SEL selects voltage levels to enable HS termination.

Diagram**Fields**

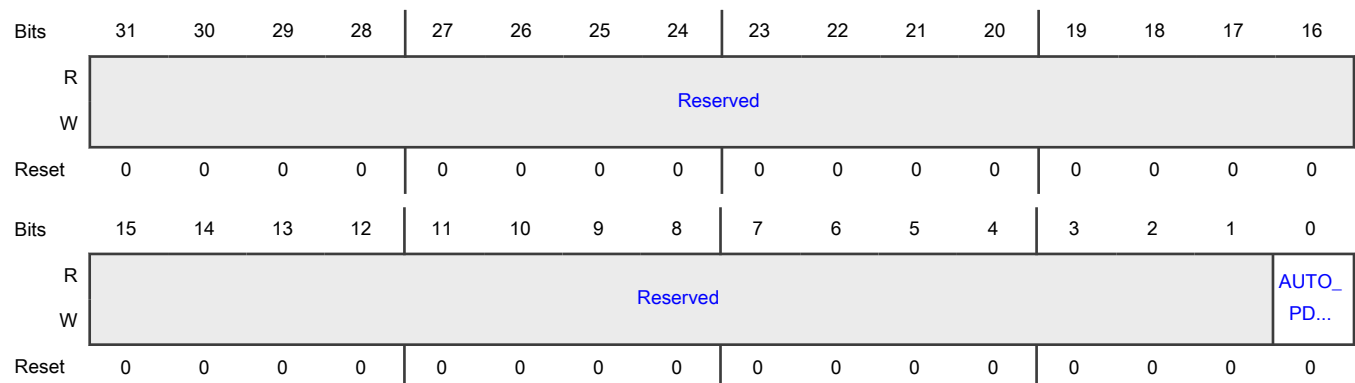
Field	Function
31-1 —	Reserved
0 RTERM_SEL	DPHY RTERM_SEL Input 0b - LPCD levels enables HS termination (VIL-CD). 1b - LPRX levels enables HS terminations (LP-VIL).

48.7.1.55 Power Down Auto Enable (AUTO_PD_EN)**Offset**

Register	Offset
AUTO_PD_EN	324h

Function

Powers down inactive lanes reported by CFG_NUM_LANES input bus.

Diagram

Fields

Field	Function
31-1 —	Reserved
0 AUTO_PD_EN	Power Down Auto Enable Input Powers down inactive lanes. 0b - Inactive lanes are powered up and driving LP11. 1b - Inactive lanes are powered down.

48.7.1.56 DPHY RXLPRP Input (RXLPRP)

Offset

Register	Offset
RXLPRP	328h

Function

RXLPRP adjusts the threshold voltage of LP-RX.

Diagram

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

Fields

Field	Function
31-2 —	Reserved
1-0 RXLPRP	DPHY RXLPRP Input Default setting 2'b01.

48.7.1.57 DPHY RXCDRP Input (RXCDRP)

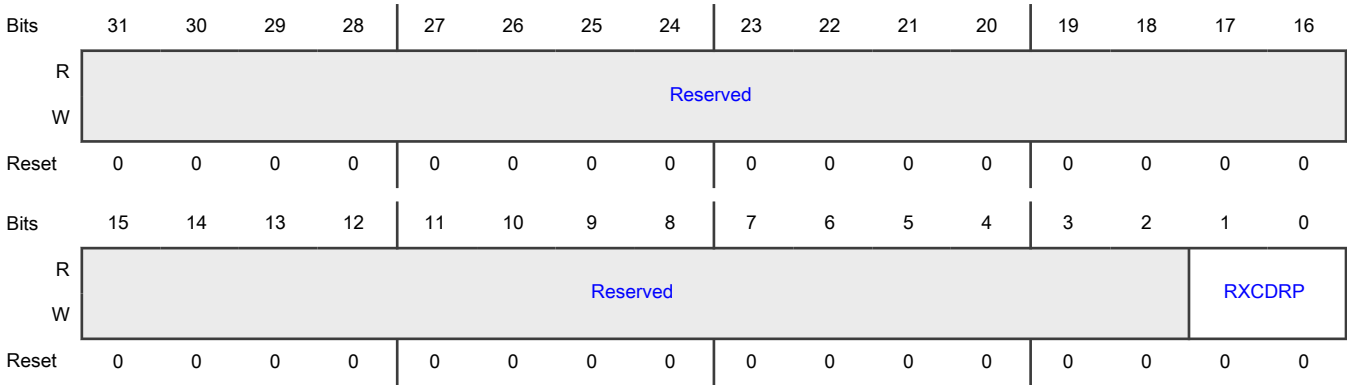
Offset

Register	Offset
RXCDRP	32Ch

Function

RXCDRP adjusts the threshold voltage of LP-CD.

Diagram



Fields

Field	Function
31-2 —	Reserved
1-0 RXCDRP	DPHY RXCDRP Input Default setting 2'b01.

Chapter 49

DMIC Subsystem

49.1 Chip-specific DMIC information

This section shows how instances of the DMIC module connect with other modules in the device.

Table 493. Reference links to related information

Topic	Related module	Reference
Full description	DMIC	Overview
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
DMA Controller	DMA	DMA Controller

49.1.1 Module instances

This device contains one instance of DMIC.

49.1.2 Initialization

Initial configuration of the DMIC can be accomplished as follows:

- Enable the clock to the DMIC in the PSCCTL0 register in CLKCTL1 register section. This enables the register interface and the peripheral function clock.
- Select a clock source for the DMIC using the DMIC0FCLKSEL register in CLKCTL1 register section.
- Select a clock divide for the DMIC using the DMIC0FCLKDIV register in CLKCTL1 register section.
- Clear the DMIC peripheral reset in the PRSTCTL0 register by writing to the PRSTCTL0_CLR register in RSTCTL1 registers.
- The DMIC provides interrupts to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable the interrupts in the STARTEN0 register.
- Use the IOPCTL registers to connect the DMIC inputs and outputs to external pins.
- The DMIC provides DMA requests to the DMA controller.
- PDM internal setup:
 - Enable DMIC PDM channels via the EN_CH0/1 bits in the CHANEN register.
 - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers.
 - If interrupts are used with this peripheral, enable them in the NVIC.
 - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. DMA must also be enabled in the FIFOCTRL[DMAEN] field for each channel using DMA.
 - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
 - The HWVAD is active when the DMIC interface is active.
 - Reset the filters with a '1' pulse of the HWVADRSTT[RSTT] field.

- Wait for a few milliseconds to let the filter converge.
- Enable the HWVAD interrupt with the appropriate NVIC bit.
- Start the HWVAD process by toggling the HWVADST10[ST10] field from '0' to '1' and back. This also clears the interrupt flag inside the HWVAD block.
- In the HWVAD interrupt service routine take appropriate action.
- Restart the HWVAD.

Wake-up for DMA only

The device can optionally be woken up from deep-sleep only far enough to perform needed DMA before returning to deep-sleep mode without the CPU waking up at all. This applies only if some interrupt requests DMA in deep-sleep mode. See the description of the HWWAKE register in SYSCTL.

- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related DMIC interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable WAKEDMA in the HWWAKE register in SYSCTL.

49.1.3 Chip-specific Signals information

The DMIC subsystem PDM signals description table gives a brief summary of each of the PDM signals used by the DMIC subsystem.

NOTE

In order to output 8 channels of PDM Data, PDM_CLK01 must be used.

Table 494. DMIC subsystem PDM signals description

Signal	I/O	Description
PDM_CLK01, PDM_CLK23, PDM_CLK45, PDM_CLK67	O	Clock output to digital microphone(s) on the related PDM interface.
PDM_DATA01, PDM_DATA23, PDM_DATA45, PDM_DATA67	I	Data input from digital microphone(s) on the related PDM interface.

Recommended IOPCTL settings are shown here. Refer to [IOPCTL](#) chapter.

Table 495. Suggested PDM signal settings for the audio input

IOPCTL bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).

Table continues on the next page...

Table 495. Suggested PDM signal settings for the audio input (continued)

IOPCTL bit(s)	Name	Comment
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless pseudo open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

49.1.4 Clocking and Outputs

The source for the DMIC base clock can be set in the DMIC0FCLKSEL register in CLKCTL1. Note that all of these clock sources may be divided by a factor of up to 256 by the DMIC clock divider, controlled by the DMIC0FCLKDIV register in CLKCTL1.

- FRO Clock
- Audio PLL Clock
- Master Clock In
- LPOSC
- 32 KHz Wake Clock

For the DMIC clock, the base clock divider values can be set in DIVHFCLK registers.

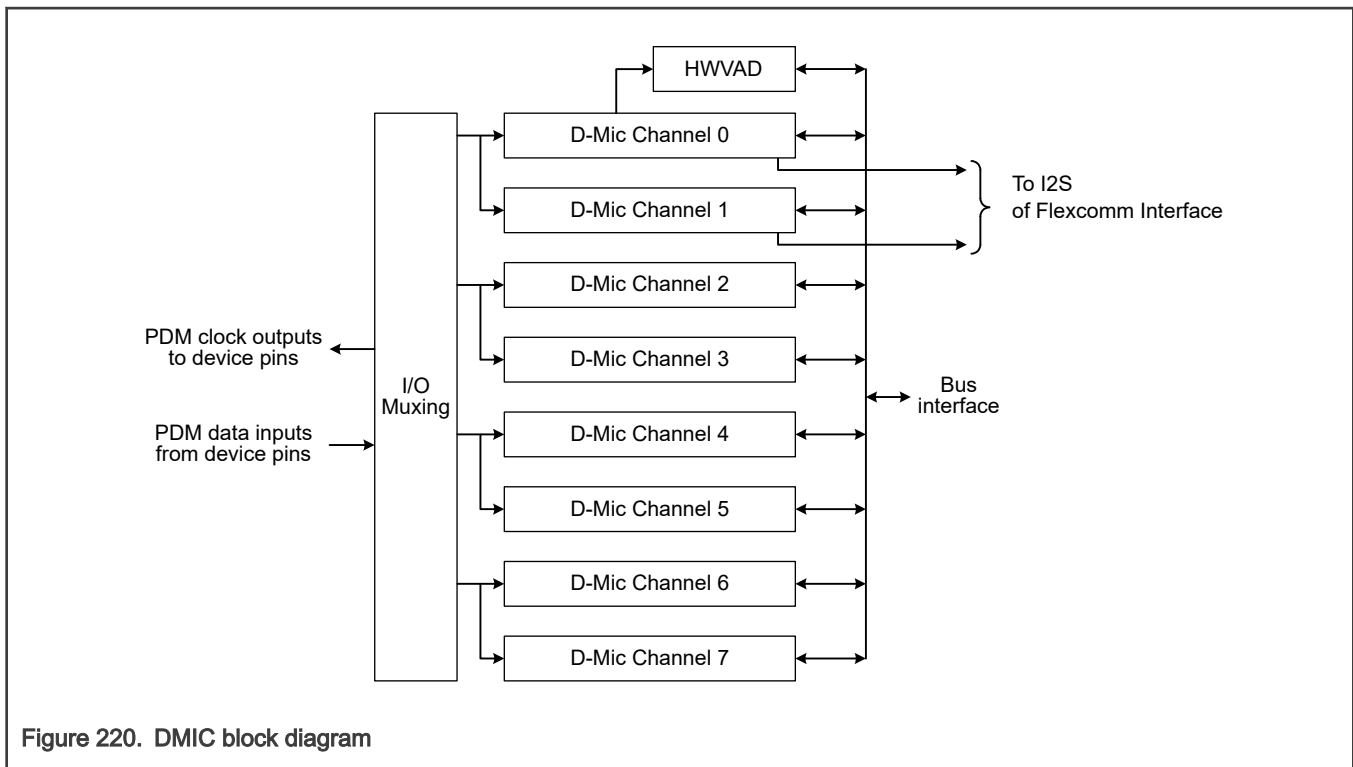
However, for power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example using register DMIC0FCLKDIV register in CLKCTL1 block).

Data from DMIC channels 0 and 1 can optionally be sent directly to the Flexcomm0 I2S function for output.

49.2 Overview

The DMIC subsystem, includes the dual-channel digital PDM microphone interface (DMIC) and hardware voice activity detector (HWVAD).

49.2.1 Block diagram



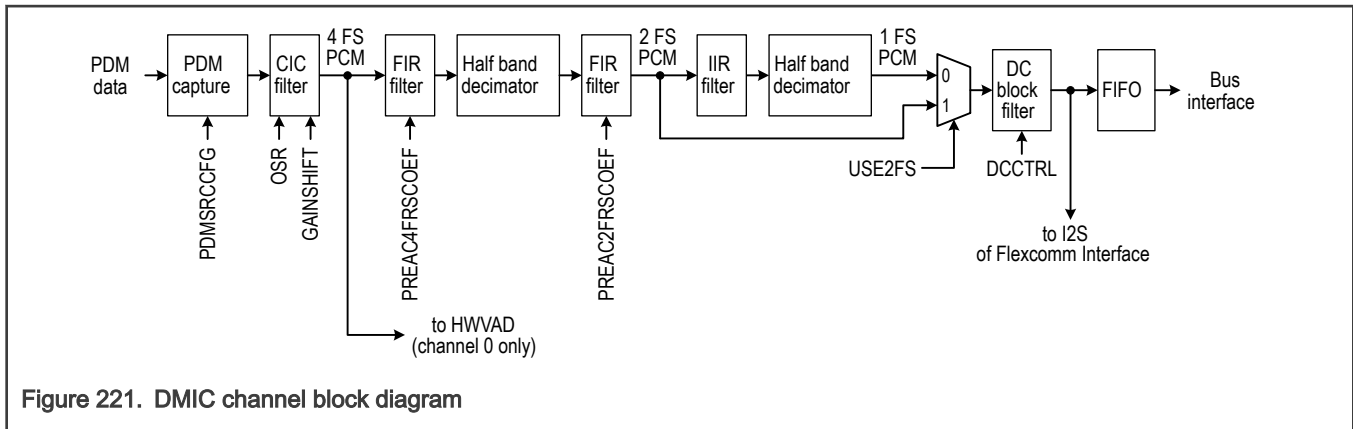
49.2.2 Features

- DMIC (dual/stereo digital microphone interface)
 - PDM (Pulse-Density Modulation) data input for up to 8 total channels on 4 data lines.
 - Flexible decimation.
 - 16 entry FIFO for each channel
 - DC blocking or unaltered DC bias can be selected
 - DMA supported on a channel-by-channel basis
 - Data can be transferred using DMA from Deep-sleep mode without waking up the CPU, then automatically returning to Deep-sleep mode.
 - Data from DMIC channels 0 and 1 can optionally be sent directly to a Flexcomm I²S function for output.
- HWVAD (Hardware-based voice activity detector):
 - Optimized for PCM signals with 16 kHz sampling frequency
 - Configurable detection levels
 - Noise envelope estimator register output for further software analysis

49.3 Functional description

49.3.1 DMIC

The DMIC interface receives PDM data from multiple digital microphones and processes it to produce 24-bit PCM data. This data can be read by the CPU or DMA, and/or can be sent to a Flexcomm I²S module for output (see chip-specific information). Many aspects of DMIC operation can be controlled. A block diagram of one DMIC channel is shown below.

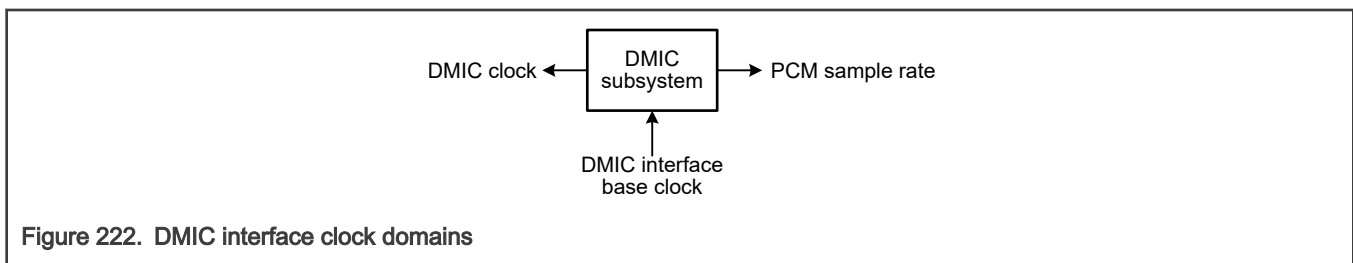


See chip-specific information.

49.3.1.1 Clocking and DMIC data rates

The DMIC interface operation is determined by 3 clock domains:

- DMIC interface base clock: supply clock for the peripheral block
- DMIC clock: sample clock for the digital microphone
- PCM sample rate: sample rate of the PCM data resulting from the PDM to PCM conversion



For the DMIC clock, the base clock divider values can be set in DIVHFCLKn registers.

DIVHFCLKn divides the clock to PDM from the original clock supplied to the DMIC subsystem block.

If the supplied clock speed is 1.536MHz, a DIVHFCLKn register PDM clock divider value = 2, gives 768kHz on the microphone.

For example, to use the microphone in Low Power mode: Use 1FS mode, DMIC clock (1.536Mhz), DIVHFCLKn register PDM clock divider value = 2 (PDM clock :768Khz), sample rate = 16khz (768/(OSR=12)/ 4 (1FS mode)).

NOTE

For power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example, using the system configuration block registers).

The DMIC peripheral block is designed to run at a DMIC clock speed no faster than 6.144 MHz and with an input frequency no faster than $4 \times 6.144 \text{ MHz} = 24.576 \text{ MHz}$. With regards to power consumption the lowest possible frequency should be selected. This frequency very much depends on the application requirements.

- For a good quality voice tag recognition the DMIC should be clocked at least with 800 kHz.

Depending on the current operating mode of the application, the clocks can be set dynamically from one sample rate to the other. For a glitch free reduction of the DMIC clock rate by factor 2 the DMIC interface contains dedicated circuitry. By setting PHY_CTRLn[PHY_HALF] = 1, the DMIC clock is divided to half the frequency internally used for the filters. This enables an on-the-fly switching of the DMIC clock without affecting the operation of the filters. As long as the sample quality on half of the frequency is good enough for the application (for example in listening mode only) this reduction in frequency helps to decrease the power consumption of the external digital microphone.

If the PDM interface operates during Deep-sleep mode (always listening), then the presence of the clock source in this mode must be taken into account as well. For example, the PLL output is not present during Deep-sleep mode, but the 12 MHz FRO is there.

In general, other clocks such as the 48 MHz FRO or the watchdog oscillator are available in Deep-sleep mode. The clock choice depends on the use case and whether a faster or slower clock provides any advantage to the system. At high PDM data rates, for example at 6 MHz, a 48 MHz clock shortens the 'awake' periods compared to 12 MHz operation. A trade-off between the sleep and the active periods, and the internal voltage required at the chosen clock rate, that determines which of the clocks perform better in terms of average power consumption.

The watchdog oscillator low power operation can help to reduce the power consumption in simple voice detection setups. By running the DMIC interface on the slow watchdog oscillator frequency, the HWVAD feature can provide a first audio detection trigger signal to the system. After wake-up, the sample rate as well as the processor performance is increased in order to run more sophisticated voice detection and/or voice recognition algorithms.

49.3.1.2 PDM to PCM conversion

The filter block for PDM to PCM conversion consists of four stages. For more details on filters, refer to AN12590 *PDM MEMS Microphone Audio Path Optimal Settings*.

1. It begins with a CIC (Cascaded-Integrator Comb) filter which is an optimized finite impulse response (FIR) filter combined with a decimator. The CIC filter converts the PDM stream from the digital microphone into PCM data with a given oversampling rate, set in the OSR registers for each channel.
2. The second block performs a decimation by 2 and compensates for a roll-off at the upper limit of the audio band.
3. The third block decimates the signal again by half, resulting in a PCM signal with the desired sample rate.
4. A final DC filter removes any unwanted DC component in the audio signal.

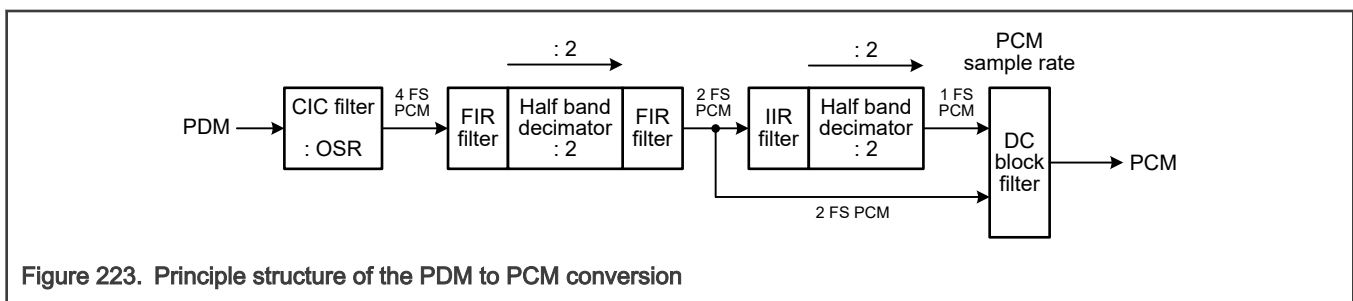


Figure 223. Principle structure of the PDM to PCM conversion

Choosing 2 FS reduces the required DMIC base clock by a factor of 2. Set this in the USE2FS[USE2FS] bit.

The PDM to PCM conversion block is designed for providing best results for PCM output signals with a 16 kHz sample rate, covering the enhanced 8 kHz speech band widely used in communication systems. However, other sample rates can be realized as well.

The final relation between the DMIC clock rate and the PCM audio sample rate is:

Table 496. DMIC input and output clock rates

2 FS mode	1 FS mode
PCM sample rate = DMIC clock rate / (2 x OSR)	PCM sample rate = DMIC clock rate / (4 x OSR)

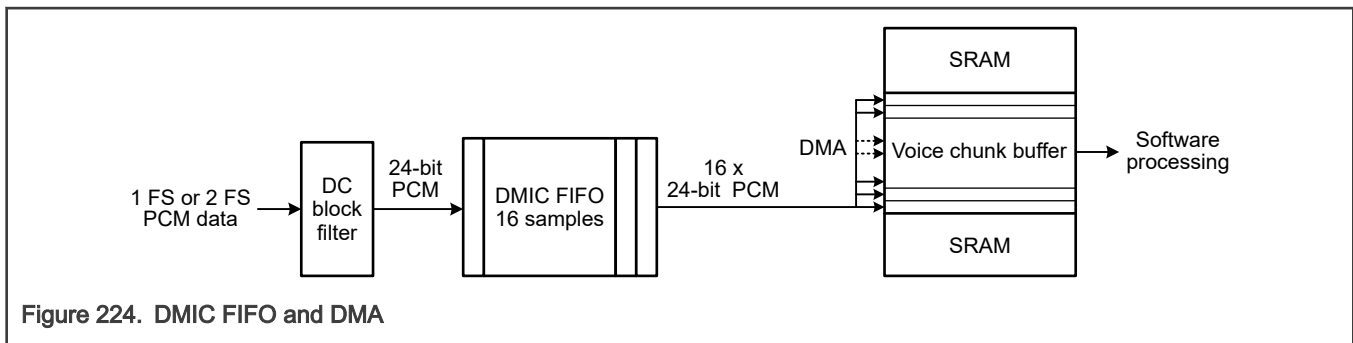
Example: DMIC clock = 800 kHz, 2 FS mode used, OSR = 25

PCM sample rate = 800 kHz / (2 x 25)

PCM sample rate = 16 kHz

The 800 kHz DMIC data is downsampled by 25 times to 32 kHz. With the following half-band filter (2 FS) the final PCM sample rate is 16 kHz.

49.3.1.3 FIFO and DMA operation



The 24-bit wide FIFO of the DMIC interface consists of 16 entries for each of the channels. The trigger level for the FIFO can be set in FIFO_CTRLn register individually for each channel.

The trigger level interrupt for the DMIC interface needs to be enabled in the NVIC, and with the related FIFO_CTRLn[INTEN] bit. The FIFO_CTRLn[DMAEN] bit enables DMA operation. With each FIFO trigger level event the DMA performs a copy of the data from the FIFO into SRAM. This data batching works without contribution of the core. When reaching the defined chunk buffer size, the DMA issues an interrupt to the Arm core for further processing of the data.

This also works when the device is in Deep-sleep mode, as the FIFO event is able to wake up the required part of the hardware. After the DMA finishes the job, the device will return to Deep-sleep mode. See the DMA chapter for DMIC channel DMA requests and trigger muxes.

Since each DMIC channel provides a separate DMA request, the most obvious configuration of DMA is to have left and right data in separate memory buffers. However, it is possible to configure the DMA controller to interleave left and right data if that is preferable in the application. To do this, the DMA is set up with a data size of halfword, but the next address written to is a word address distance away. The two descriptors would be started on consecutive halfwords. Data is delivered by the DMIC as left channel followed by right channel for each PCM stereo sample.

If more history data is required for a software algorithm, another DMA request can be set up, which copies the current chunk into a larger ring buffer structure. For algorithms like voice detection or voice recognition this is key, in order to converge the software filters to the current background noise situation.

For operation without DMA the dedicated DMIC FIFO interrupt can be enabled in order to inform the Arm core about the FIFO status.

Example:

- PCM output sample rate is 16 kHz
- The FIFO gets full every 1 ms
- The DMA copies every 1 ms the content of the DMIC FIFO to SRAM
- The DMA is configured to move 512 bytes (= 256 PCM samples) from the DMIC FIFO to SRAM before issuing a DMA interrupt
- Every 16 ms the 256 PCM samples are processed by the Arm core

The FIFO_CTRLn[DMAEN] field can be used to synchronize the start of multiple channels, allowing a single DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

49.3.1.4 Usage of the DMIC interface in reduced power modes

The DMIC interface can batch the serial PDM stream from a digital microphone in Deep-sleep mode. This requires an appropriate base clock which is active in the respective power saving mode. The best fit for this base clock is the 12 MHz FRO, which provides a good trade-off between power consumption and performance. For lower power operation the watchdog oscillator can be used, taking into account that the clock is relatively inaccurate and the DMIC sample rate is rather low. At any time an external low power clock, connected to pin MCLK, can be used.

In combination with the HWVAD this provides lowest power consumption in listening mode. Except for the short periods with DMA activity, the MCU can remain in Deep-sleep mode until the wave envelope detector of the HWVAD identifies an energy change event and issues an interrupt. With the DMA set to larger transfer sizes (maximum is 1024 transfers), there is history data available for any type of software-based analysis of the data causing the HWVAD event.

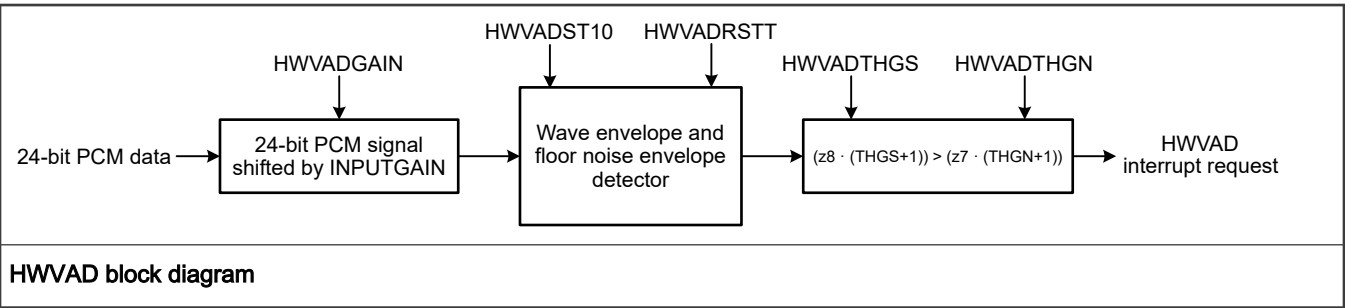
This history data also enables the system to realize different strategies for dealing with an HWVAD event. A concrete analysis of the data could for example just be started when the HWVAD detected events over a longer time frame. Analyzing the data to to understand known spurious noise could lead to strategies to avoid activating the Arm core for unnecessary events. In case the decision has been taken to take the next step in data analysis, the history buffer still contains the complete PCM data sampled since the first event, nothing is lost. The system can stay in power save mode longer if spurious events can be filtered out.

49.3.2 HWVAD

The hardware voice activity detector (HWVAD) implements a wave envelope detector and a floor noise envelope detector. It provides an interrupt when the delta between the two detectors is larger than a predefined value. The input signal for the HWVAD can come from DMIC channel 0. The basic detection of a voice activity can be the starting point for a more sophisticated task like voice recognition. As with the DMA for the DMIC subsystem, the HWVAD can be active during Deep-sleep mode and provide lower power operation, compared with a software based implementation. The DMIC receives PDM data and produces a data stream that can be read by the CPU or DMA.

The hardware voice activity detector (HWVAD) analyzes the PCM data from the DMIC channel 0 by means of a filter block. Both the noise floor and the signal wave are examined and result in separate filter outputs. The HWVAD interrupt is issued when a specific delta between the signal and the noise result is detected

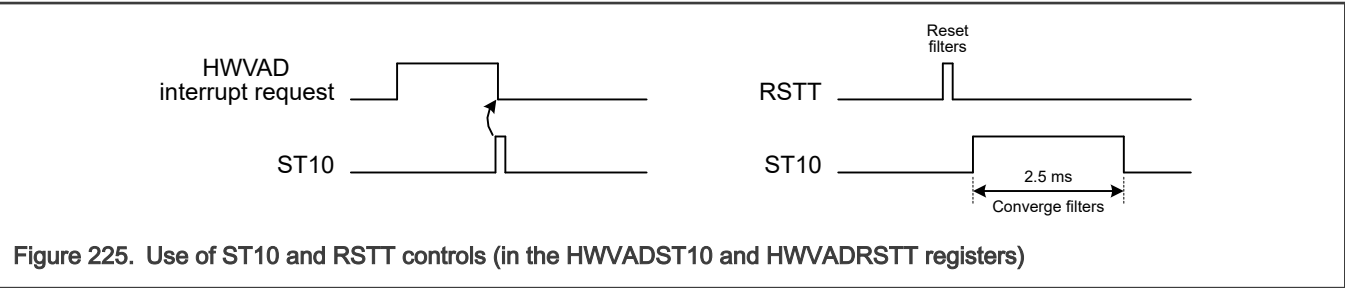
Gain levels for the input signal as well as for the signal and noise filter outputs can be set independently from each other, in order to adapt the HWVAD to different acoustic situations.



Because of the non-uniqueness of the input signal, which includes noise and voice with various frequency components and different volumes, there is no one-and-only operation mode for the HWVAD. A few parameters as well as the chronology can play an important role for a good performance.

49.3.2.1 Basic operations

There are some basic operations for the HWVAD, which can be combined in various ways to achieve different behavior.



- 1. The internal HWVAD interrupt flag is reset with a short pulse on ST10
- 2. A pulse on RSTT resets all detection filters

3. Keeping ST10 high for a period causes a special filter convergence. The 2.5 ms period is valid for 800 kHz DMIC sample rate. For 1 MHz sample rate the period is 2 ms.

Use the HWVADST10[ST10] bit to prepare HWVAD for an interrupt.

The internal flag is reset with the rising edge of ST10 and HWVAD waits for the next event. In case the application involves some post-processing after a HWVAD event (outside of the interrupt service routine), the flag should only be cleared at the end of this processing. The interrupt status on NVIC level is not affected by this bit setting.

Use the HWVADRSTT[RSTT] bit to reset all filters.

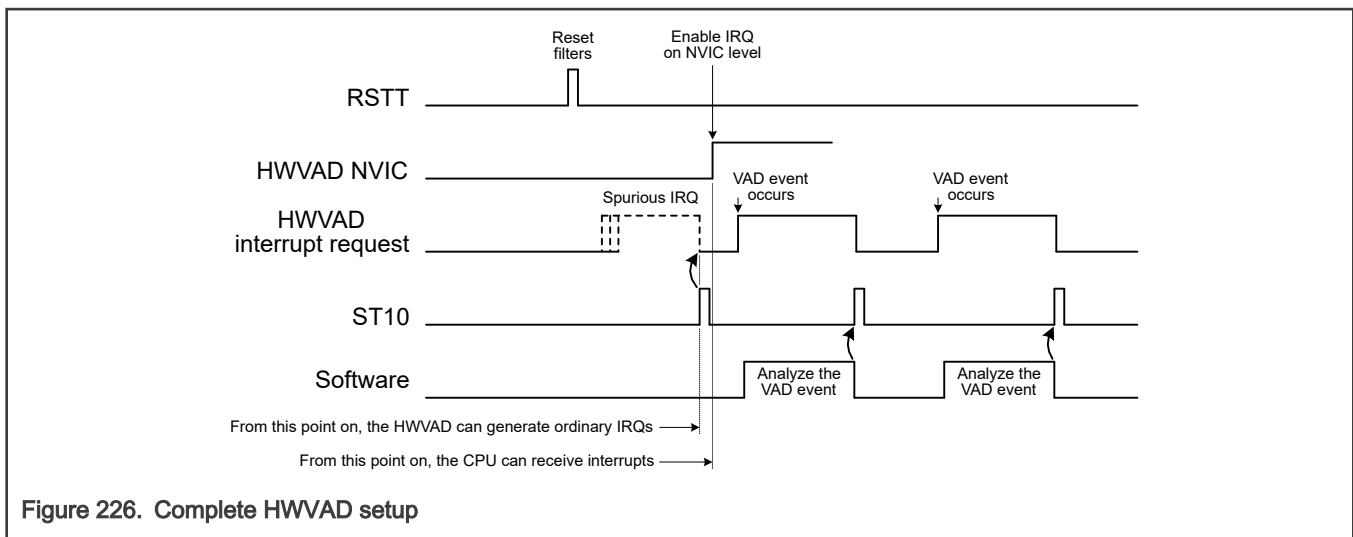
After this reset the HWVAD filters need to converge, so for the first few milliseconds the result is not reliable. The HWVAD interrupt should be masked on the NVIC level during this time frame. The wait period depends on the sample rate of the incoming data.

- For a 1 MHz DMIC sample rate the filters need about 2ms to converge.
- For an 800 kHz DMIC sample rate the filters need about 2.5ms to converge.

Depending on the use case, it might make sense to reset the filters before starting into a new detection process. For a voice application the filters can adapt continuously to the background environment, between the voice events there is normally enough time to let the filters converge to a changed background noise situation.

Keeping ST10 on high level during the convergence period enables a special mode. If the filters should adapt to a current background noise floor (without voice), then this can be done during this period. With HWVADST10[ST10] returning to a low level, the filter calculation is then based on a different filter pre-setting. This could be an advantage in special types of applications, where the signal is not continuously delivered to the HWVAD. In a DMIC system with continuous sampling, this convergence period is not required. In that case, the HWVADST10[ST10] bit is just used to clear the interrupt flag.

A complete setup sequence for standard operation looks like this:



1. Reset filters with the HWVADRSTT[RSTT] bit and provide some time to let the filter converge to the signal conditions.
2. Pulse bit HWVADST10[ST10] to clear any spurious interrupts which were generated during bad filter conditions.
3. Enable HWVAD interrupts on the NVIC level.
4. Process the VAD event in case of an interrupt, when finished clear the interrupt flag with a high pulse of HWVADST10[ST10].

49.3.2.2 Extended operation

There are a few parameters which can be set to influence the behavior of the HWVAD. There is also an intermediate filter result value available, which can be used for proprietary software-based analysis.

49.3.2.2.1 Input gain setting

The 24-bit PCM input signal can be shifted left or right with the gain setting in the HWVADGAIN register. This increases or decreases the volume of the input signal for the HWVAD processing. Note that the reset value 0x05 equals a gain factor of 1, where the signal is not shifted in either direction.

49.3.2.2.2 Filter result gain setting

The output values for the final equation can also have a gain factor.

$$[z8 \times (THGS + 1)] > [z7 \times (THGN + 1)]$$

These gain factors determine the proportion between the results of the signal and the noise filters. The values depend on the audio signal and noise environment.

- For low noise environments, the reset values THGN = 0 and THGS = 4 are more suitable.
- For noisy environments, the gain for THGN and THGS needs to be increased.
- In a typical voice recognition application, THGN = 3 and THGS = 6 is a good starting point.

49.3.2.2.3 High pass filter setting

The setting in the HWVADHPFS register can be used to adapt the filters to different background noise situations. In order to find the best setting, software could perform a rough spectral analysis of the audio signal.

- For a background with more low-frequency content, HWVADHPFS[HPFS] should be set to 0x1. This is the standard use case.
- For environments where the low-frequency content is small, HWVADHPFS[HPFS] can be set to 0x2.

49.3.2.2.4 Noise floor evaluation

The register HWVADLOWZ contains 2 bytes of the output of filter stage z7, which computes the noise floor. The characteristic of the filter block for voice applications is best for a value of 500 - 1000 in LOWZ. Software can tune the input gain to get the HWVADLOWZ[LOWZ] value into this region.

NOTE

For power saving reasons the HWVADLOWZ register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read is not in the middle of a transition.

49.4 Signals

This table describes the PDM signals used by the DMIC module.

Signal	I/O	Description
PDM_CLKn	O	Clock output to digital microphone(s) on the related PDM interface.
PDM_DATAn	I	Data input from digital microphone(s) on the related PDM interface.

The PDM interface provides options to support 2 single-channel microphones or a single stereo microphone. The general connections are shown in the DMIC subsystem signal multiplexing figure.

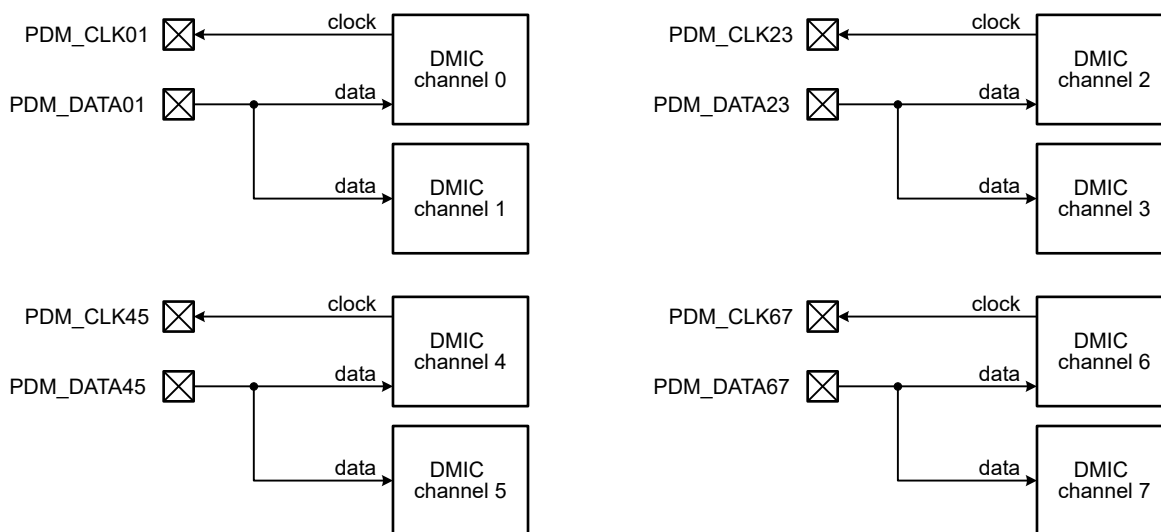


Figure 227. DMIC subsystem signal multiplexing

These figures show specific use case examples.

The first example is the simple case of one independent microphone.

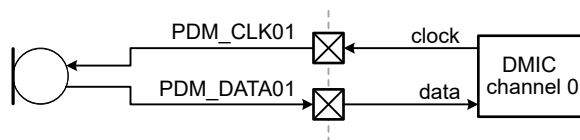


Figure 228. Example connection for a single independent microphone

This example shows two independent microphones.

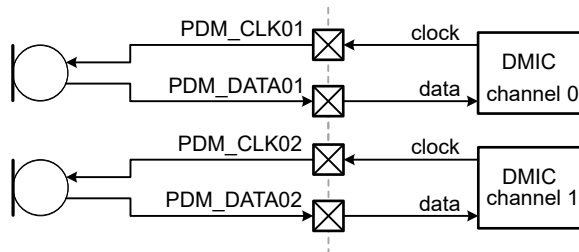


Figure 229. Example connection to two independent microphones

For the example with two microphones sharing the same clock and data lines, the DMIC channel 0 and DMIC channel 1 are configured to use the same source clock and frequency. Both microphones use the same clock from channel 0.

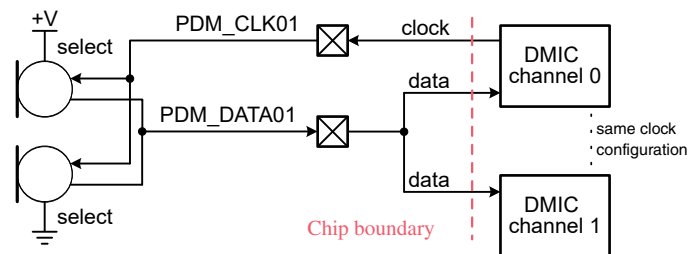


Figure 230. Example connection to two microphones sharing clock and data lines

49.5 Initialization

This section describes the general initialization of the DMIC module.

- Enable the clock to the DMIC by in the clock control registers. This enables the register interface and the peripheral function clock.
- Select a clock source for the DMIC in the clock control registers.
- Select a clock divide for the DMIC in the clock control registers.
- Clear the DMIC peripheral reset in the reset registers.
- The DMIC provides interrupts to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable interrupts in the system configuration registers.
- Use the IO Pad controller (IOPCTL) registers to connect the DMIC inputs and outputs to external pins.
- The DMIC provides DMA requests to the DMA controller.
- PDM internal setup:
 - Enable DMIC PDM channels via the CHANEN register.
 - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers.
 - If interrupts will be used with this peripheral, enable them in the NVIC.
 - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. DMA must also be enabled in the FIFO_CTRLn[DMAEN] field for each channel using DMA.
 - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
 - The HWVAD is active when the DMIC interface is active.
 - Reset the filters with '1' pulse of HWVADRSTT[RSTT].
 - Wait for a few milliseconds to let the filter converge.
 - Enable the HWVAD interrupt with the appropriate NVIC bit.
 - Start the HWVAD process by toggling the HWVADST10[ST10] field from '0' to '1' and back. This also clears the interrupt flag inside the HWVAD block.
 - In the HWVAD interrupt service routine take appropriate action.
 - Restart the HWVAD.

Wake-up for DMA only

The device can optionally wake-up from Deep-sleep to perform a needed DMA before returning to Deep-sleep mode without the CPU. This applies only if some interrupt requests DMA in Deep-sleep mode.

- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related DMIC interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable the associated hardware wake-up register in the system configuration registers.

49.6 Memory map and register definition

This section includes the DMIC module memory map and detailed descriptions of all registers.

Each of the 8 DMIC channels has a set of configuration registers with the channel noted with a suffix 0 - 7. The n at the end of the register name stands for the channel 0 - 7.

- OSRn
- DIVHFCLKn
- PREAC2FSCOEFn
- PREAC4FSCOEFn
- GAINSHIFTn
- FIFO_CTRLn
- FIFO_STATUsn
- FIFO_DATA n
- PHY_CTRLn
- DC_CTRLn

Other DMIC registers control the HWVAD, FS output choice (USE2FS), and multiple channel enable (CHANEN).

49.6.1 DMIC register descriptions

49.6.1.1 DMIC memory map

DMIC0 base address: 4012_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Oversample Rate (OSR0)	32	RW	0000_0000h
4h	DMIC Clock (DIVHFCLK0)	32	RW	0000_0000h
8h	Compensation Filter for 2 FS (PREAC2FSCOEf0)	32	RW	0000_0000h
Ch	Compensation Filter for 4 FS (PREAC4FSCOEf0)	32	RW	0000_0000h
10h	Decimator Gain Shift (GAINSHIFT0)	32	RW	0000_0000h
80h	FIFO Control (FIFO_CTRL0)	32	RW	0000_0000h
84h	FIFO Status (FIFO_STATUS0)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
88h	FIFO Data (FIFO_DATA0)	32	R	0000_0000h
8Ch	Physical Control (PHY_CTRL0)	32	RW	0000_0000h
90h	DC Filter Control (DC_CTRL0)	32	RW	0000_0000h
100h	Oversample Rate (OSR1)	32	RW	0000_0000h
104h	DMIC Clock (DIVHFCLK1)	32	RW	0000_0000h
108h	Compensation Filter for 2 FS (PREAC2FSCOEF1)	32	RW	0000_0000h
10Ch	Compensation Filter for 4 FS (PREAC4FSCOEF1)	32	RW	0000_0000h
110h	Decimator Gain Shift (GAINSHIFT1)	32	RW	0000_0000h
180h	FIFO Control (FIFO_CTRL1)	32	RW	0000_0000h
184h	FIFO Status (FIFO_STATUS1)	32	RW	0000_0000h
188h	FIFO Data (FIFO_DATA1)	32	R	0000_0000h
18Ch	Physical Control (PHY_CTRL1)	32	RW	0000_0000h
190h	DC Filter Control (DC_CTRL1)	32	RW	0000_0000h
200h	Oversample Rate (OSR2)	32	RW	0000_0000h
204h	DMIC Clock (DIVHFCLK2)	32	RW	0000_0000h
208h	Compensation Filter for 2 FS (PREAC2FSCOEF2)	32	RW	0000_0000h
20Ch	Compensation Filter for 4 FS (PREAC4FSCOEF2)	32	RW	0000_0000h
210h	Decimator Gain Shift (GAINSHIFT2)	32	RW	0000_0000h
280h	FIFO Control (FIFO_CTRL2)	32	RW	0000_0000h
284h	FIFO Status (FIFO_STATUS2)	32	RW	0000_0000h
288h	FIFO Data (FIFO_DATA2)	32	R	0000_0000h
28Ch	Physical Control (PHY_CTRL2)	32	RW	0000_0000h
290h	DC Filter Control (DC_CTRL2)	32	RW	0000_0000h
300h	Oversample Rate (OSR3)	32	RW	0000_0000h
304h	DMIC Clock (DIVHFCLK3)	32	RW	0000_0000h
308h	Compensation Filter for 2 FS (PREAC2FSCOEF3)	32	RW	0000_0000h
30Ch	Compensation Filter for 4 FS (PREAC4FSCOEF3)	32	RW	0000_0000h
310h	Decimator Gain Shift (GAINSHIFT3)	32	RW	0000_0000h
380h	FIFO Control (FIFO_CTRL3)	32	RW	0000_0000h
384h	FIFO Status (FIFO_STATUS3)	32	RW	0000_0000h
388h	FIFO Data (FIFO_DATA3)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
38Ch	Physical Control (PHY_CTRL3)	32	RW	0000_0000h
390h	DC Filter Control (DC_CTRL3)	32	RW	0000_0000h
400h	Oversample Rate (OSR4)	32	RW	0000_0000h
404h	DMIC Clock (DIVHFCLK4)	32	RW	0000_0000h
408h	Compensation Filter for 2 FS (PREAC2FSCOEF4)	32	RW	0000_0000h
40Ch	Compensation Filter for 4 FS (PREAC4FSCOEF4)	32	RW	0000_0000h
410h	Decimator Gain Shift (GAINSHIFT4)	32	RW	0000_0000h
480h	FIFO Control (FIFO_CTRL4)	32	RW	0000_0000h
484h	FIFO Status (FIFO_STATUS4)	32	RW	0000_0000h
488h	FIFO Data (FIFO_DATA4)	32	R	0000_0000h
48Ch	Physical Control (PHY_CTRL4)	32	RW	0000_0000h
490h	DC Filter Control (DC_CTRL4)	32	RW	0000_0000h
500h	Oversample Rate (OSR5)	32	RW	0000_0000h
504h	DMIC Clock (DIVHFCLK5)	32	RW	0000_0000h
508h	Compensation Filter for 2 FS (PREAC2FSCOEF5)	32	RW	0000_0000h
50Ch	Compensation Filter for 4 FS (PREAC4FSCOEF5)	32	RW	0000_0000h
510h	Decimator Gain Shift (GAINSHIFT5)	32	RW	0000_0000h
580h	FIFO Control (FIFO_CTRL5)	32	RW	0000_0000h
584h	FIFO Status (FIFO_STATUS5)	32	RW	0000_0000h
588h	FIFO Data (FIFO_DATA5)	32	R	0000_0000h
58Ch	Physical Control (PHY_CTRL5)	32	RW	0000_0000h
590h	DC Filter Control (DC_CTRL5)	32	RW	0000_0000h
600h	Oversample Rate (OSR6)	32	RW	0000_0000h
604h	DMIC Clock (DIVHFCLK6)	32	RW	0000_0000h
608h	Compensation Filter for 2 FS (PREAC2FSCOEF6)	32	RW	0000_0000h
60Ch	Compensation Filter for 4 FS (PREAC4FSCOEF6)	32	RW	0000_0000h
610h	Decimator Gain Shift (GAINSHIFT6)	32	RW	0000_0000h
680h	FIFO Control (FIFO_CTRL6)	32	RW	0000_0000h
684h	FIFO Status (FIFO_STATUS6)	32	RW	0000_0000h
688h	FIFO Data (FIFO_DATA6)	32	R	0000_0000h
68Ch	Physical Control (PHY_CTRL6)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
690h	DC Filter Control (DC_CTRL6)	32	RW	0000_0000h
700h	Oversample Rate (OSR7)	32	RW	0000_0000h
704h	DMIC Clock (DIVHFCLK7)	32	RW	0000_0000h
708h	Compensation Filter for 2 FS (PREAC2FSCOEF7)	32	RW	0000_0000h
70Ch	Compensation Filter for 4 FS (PREAC4FSCOEF7)	32	RW	0000_0000h
710h	Decimator Gain Shift (GAINSHIFT7)	32	RW	0000_0000h
780h	FIFO Control (FIFO_CTRL7)	32	RW	0000_0000h
784h	FIFO Status (FIFO_STATUS7)	32	RW	0000_0000h
788h	FIFO Data (FIFO_DATA7)	32	R	0000_0000h
78Ch	Physical Control (PHY_CTRL7)	32	RW	0000_0000h
790h	DC Filter Control (DC_CTRL7)	32	RW	0000_0000h
F00h	Channel Enable (CHANEN)	32	RW	0000_0000h
F10h	Use 2 FS register (USE2FS)	32	RW	0000_0000h
F14h	Global Channel Synchronization Enable (GLOBAL_SYNC_EN)	32	RW	0000_0000h
F18h	Global channel synchronization counter value (GLOBAL_COUNT_VAL)	32	RW	0000_0000h
F1Ch	DMIC decimator reset (DECRESET)	32	RW	0000_0000h
F80h	HWVAD Input Gain (HWVADGAIN)	32	RW	0000_0005h
F84h	HWVAD Filter Control (HWVADHPFS)	32	RW	0000_0001h
F88h	HWVAD Control (HWVADST10)	32	RW	0000_0000h
F8Ch	HWVAD Filter Reset (HWVADRSTT)	32	RW	0000_0000h
F90h	HWVAD Noise Estimator Gain (HWVADTHGN)	32	RW	0000_0000h
F94h	HWVAD Signal Estimator Gain (HWVADTHGS)	32	RW	0000_0004h
F98h	HWVAD Noise Envelope Estimator (HWVADLOWZ)	32	R	0000_0000h

49.6.1.2 Oversample Rate (OSR0 - OSR7)

Offset

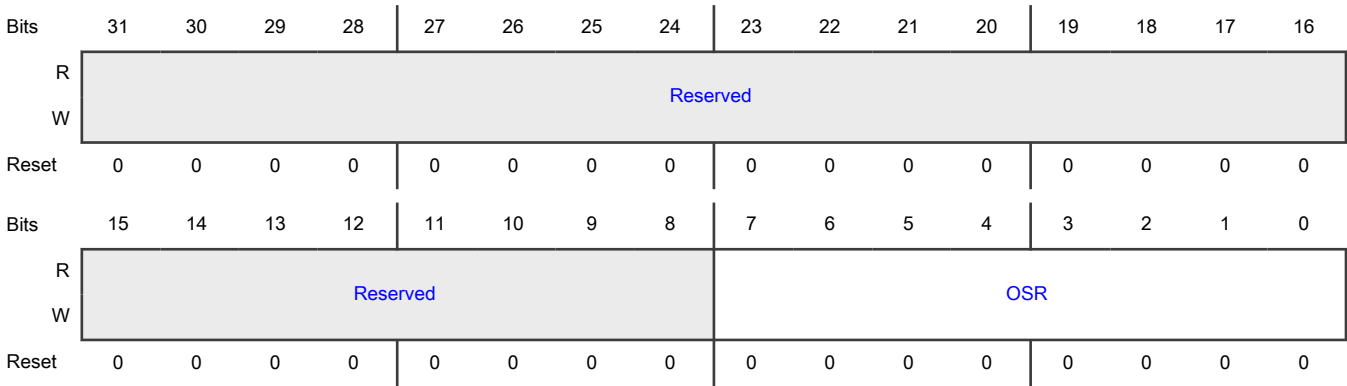
For n = 0 to 7:

Register	Offset
OSRn	0h + (n × 100h)

Function

The OSRn register selects the oversample rate (CIC decimation rate) for the related input channel.

Diagram



Fields

Field	Function
31-8	Reserved
—	Read value is undefined, only zero should be written.
7-0	Oversample Rate
OSR	Selects the oversample rate for the related input channel.

49.6.1.3 DMIC Clock (DIVHFCLK0 - DIVHFCLK7)

Offset

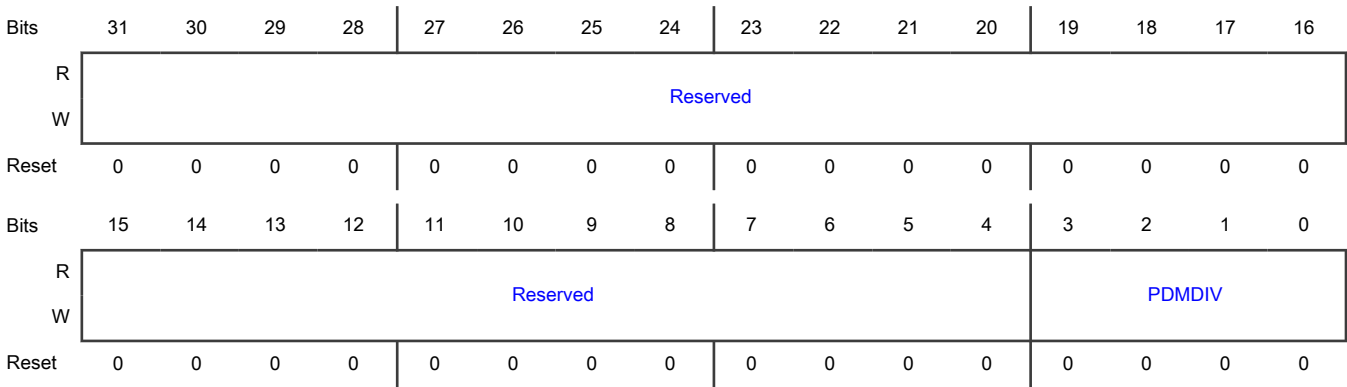
For n = 0 to 7:

Register	Offset
DIVHFCLKn	4h + (n × 100h)

Function

The DIVHFCLKn register controls the clock pre-divider for the related input channel, using the divider to generate the PDM clock from DMIC clock input.

Diagram



Fields

Field	Function
31-4	Reserved
—	Read value is undefined, only zero should be written.
3-0 PDMDIV	PDM Clock Divider Value Divide by factor to create PDM Clock (enumerated type) 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 6 0101b - Divide by 8 0110b - Divide by 12 0111b - Divide by 16 1000b - Divide by 24 1001b - Divide by 32 1010b - Divide by 48 1011b - Divide by 64 1100b - Divide by 96 1101b - Divide by 128 1110b-1111b - Reserved

49.6.1.4 Compensation Filter for 2 FS (PREAC2FSCOE0 - PREAC2FSCOE7)

Offset

For n = 0 to 7:

Register	Offset
PREAC2FSCOEFn	8h + (n × 100h)

Function

The PREAC2FSCOEFn register selects the pre-emphasis filter coefficient for the related input channel when 2 FS mode is used.

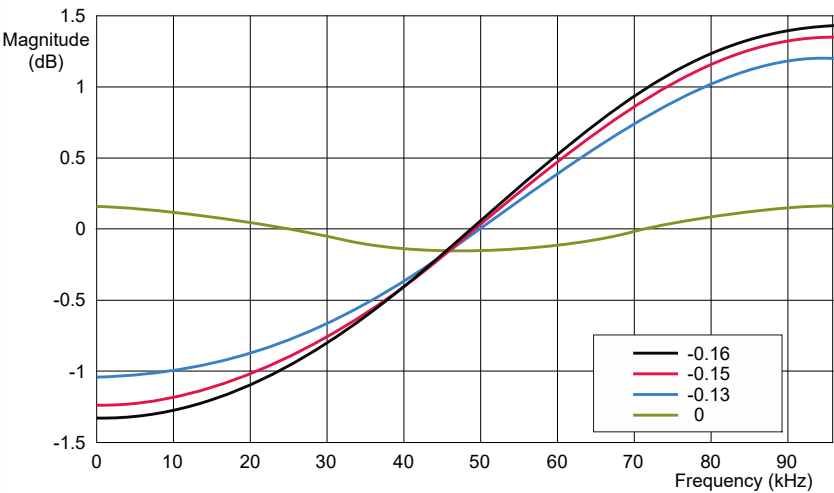
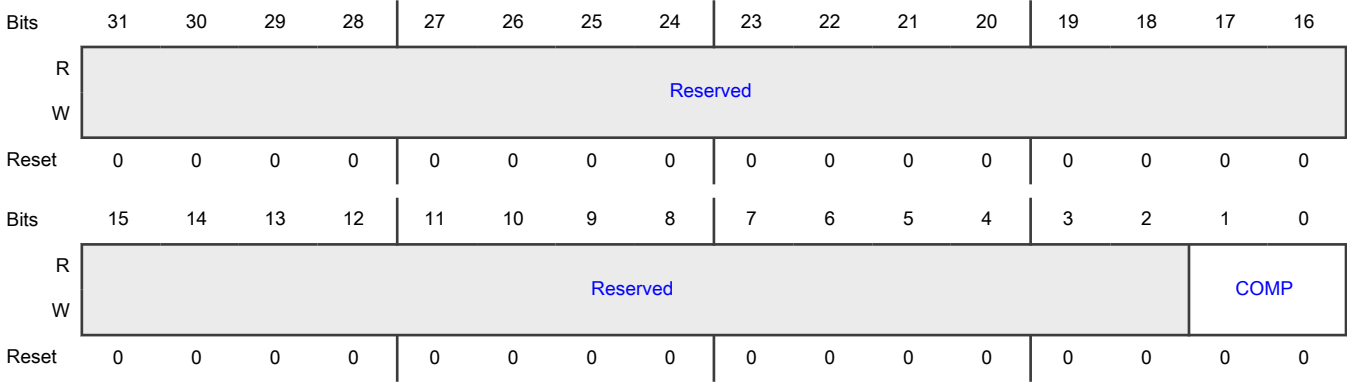


Figure 231. Pre-emphasis filter quantized response at 96 kHz

Diagram



Fields

Field	Function
31-2	Reserved
—	Read value is undefined, only zero should be written.
1-0	Compensation value
COMP	Pre-emphasis filter coefficient for 2 FS mode. Co-efficient choice for CIC droop compensation droop filter.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Compensation = 0. This is the recommended setting. 01b - Compensation = -0.16 10b - Compensation = -0.15 11b - Compensation = -0.13

49.6.1.5 Compensation Filter for 4 FS (PREAC4FSCOEf0 - PREAC4FSCOEf7)

Offset

For n = 0 to 7:

Register	Offset
PREAC4FSCOEf _n	Ch + (n × 100h)

Function

The PREAC4FSCOEf_n register selects the pre-emphasis filter coefficient for the related input channel when 4 FS mode is used. See the diagram in PREAC2FSCOEf_n.

Diagram

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

Fields

Field	Function
31-2	Reserved
—	Read value is undefined, only zero should be written.
1-0	Compensation value
COMP	Pre-emphasis filter coefficient for 4 FS mode. Co-efficient choice for CIC droop compensation droop filter.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Compensation = 0. This is the recommended setting. 01b - Compensation = -0.16 10b - Compensation = -0.15 11b - Compensation = -0.13

49.6.1.6 Decimator Gain Shift (GAINSHIFT0 - GAINSHIFT7)

Offset

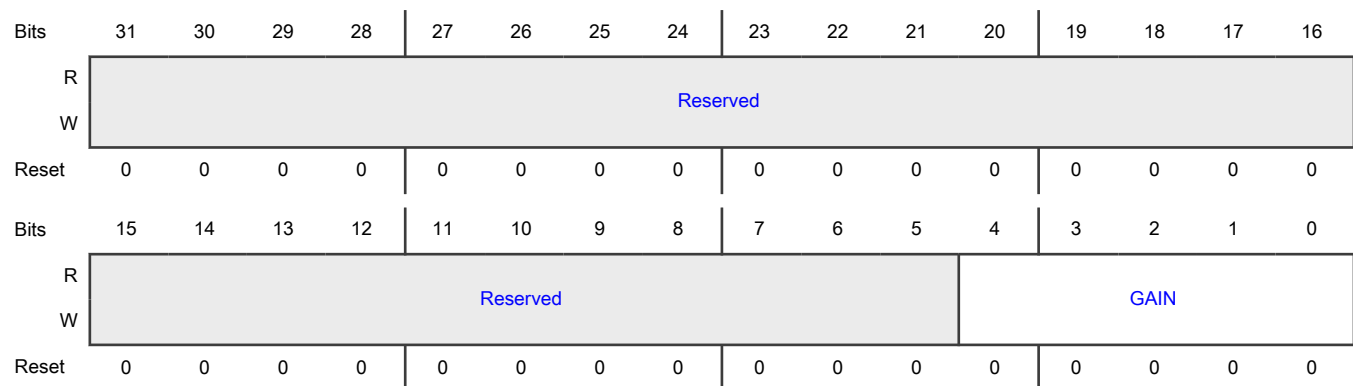
For n = 0 to 7:

Register	Offset
GAINSHIFTn	10h + (n × 100h)

Function

The GAINSHIFTn register adjusts the gain of the 4 FS PCM data from the input filter.

Diagram



Fields

Field	Function
31-5	Reserved
—	Read value is undefined, only zero should be written.
4-0	Gain
GAIN	Gain shift for decimator output can be positive or negative number. Gain control, as a positive or negative (two's complement) number of bits to shift.

49.6.1.7 FIFO Control (FIFO_CTRL0 - FIFO_CTRL7)

Offset

For n = 0 to 7:

Register	Offset
FIFO_CTRLn	80h + (n × 100h)

Function

FIFO_CTRLn configures FIFO usage.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												TRIGLVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												DMAE N	INTEN	RESE TN	ENAB LE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved Read value is undefined, only zero should be written.
20-16 TRIGLVL	FIFO Trigger Level for Interrupt Selects the data trigger level for interrupt or DMA operation. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. TRIGLVL value + 1 = the number of received entries. Representative bit settings: 0_0000b - Trigger when the FIFO has received one entry (is no longer empty). 0_0001b - Trigger when the FIFO has received two entries. 0_1110b - Trigger when the FIFO has received 15 entries. 0_1111b - Trigger when the FIFO has received 16 entries (has become full).
15-4 —	Reserved Read value is undefined, only zero should be written.
3	DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DMAEN	0b - DMA requests are not enabled. 1b - DMA requests based on FIFO level are enabled.
2 INTEN	Interrupt Enable. 0b - FIFO level interrupts are not enabled. 1b - FIFO level interrupts are enabled.
1 RESETN	FIFO Reset 0b - Reset the FIFO. This must be cleared before resuming operation. 1b - Normal operation
0 ENABLE	FIFO Enable. 0b - Disabled. FIFO is not enabled. Enabling a DMIC channel with the FIFO disabled could be useful while data is being streamed to a Flexcomm I ² S module, or in order to avoid a filter settling delay when a channel is re-enabled after a period when the data was not needed. 1b - FIFO is enabled. The FIFO must be enabled in order for the CPU or DMA to read data from the DMIC via the FIFODATA register.

49.6.1.8 FIFO Status (FIFO_STATUS0 - FIFO_STATUS7)

Offset

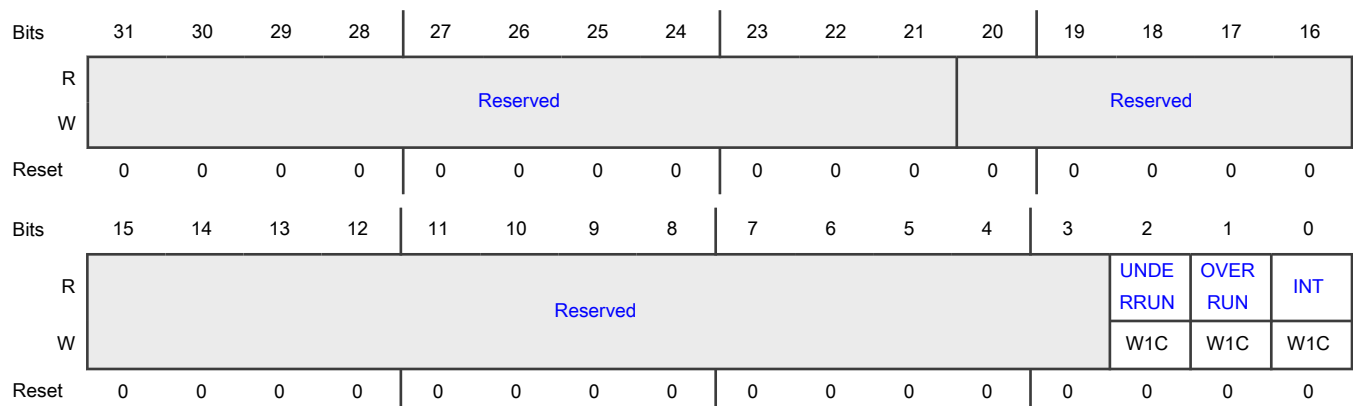
For n = 0 to 7:

Register	Offset
FIFO_STATUSn	84h + (n × 100h)

Function

The FIFO_STATUSn register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Diagram



Fields

Field	Function
31-21 —	Reserved Read value is undefined, only zero should be written.
20-16 —	Reserved
15-3 —	Reserved Read value is undefined, only zero should be written.
2 UNDERRUN	Underrun Detected (write 1 to clear)
1 OVERRUN	Overrun Detected (write 1 to clear)
0 INT	Status of Interrupt (write 1 to clear)

49.6.1.9 FIFO Data (FIFO_DATA0 - FIFO_DATA7)

Offset

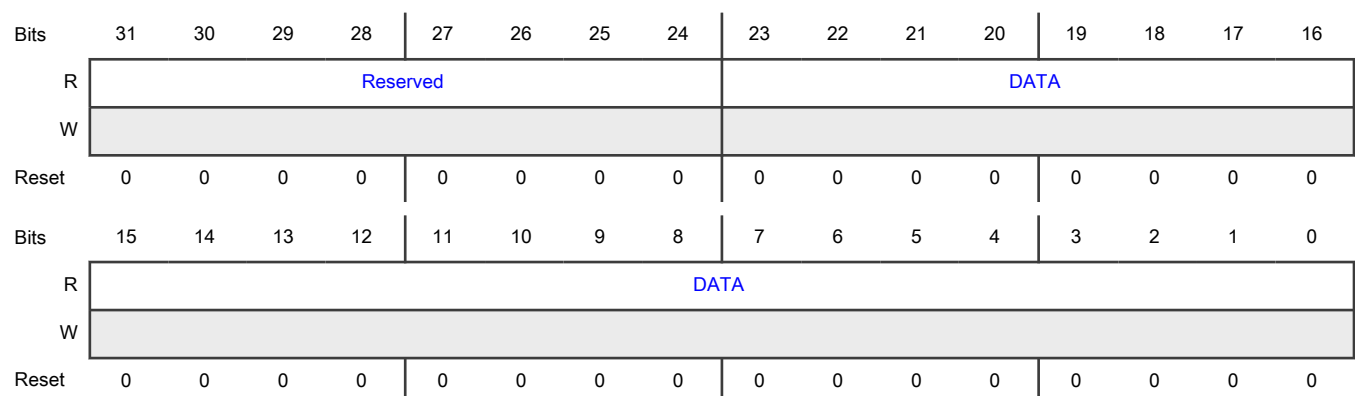
For n = 0 to 7:

Register	Offset
FIFO_DATA _n	88h + (n × 100h)

Function

The FIFODATAn register is used to read values that have been received by the via the PDM stream.

Diagram



Fields

Field	Function
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 DATA	PCM Data Data from the top of the input filter FIFO.

49.6.1.10 Physical Control (PHY_CTRL0 - PHY_CTRL7)

Offset

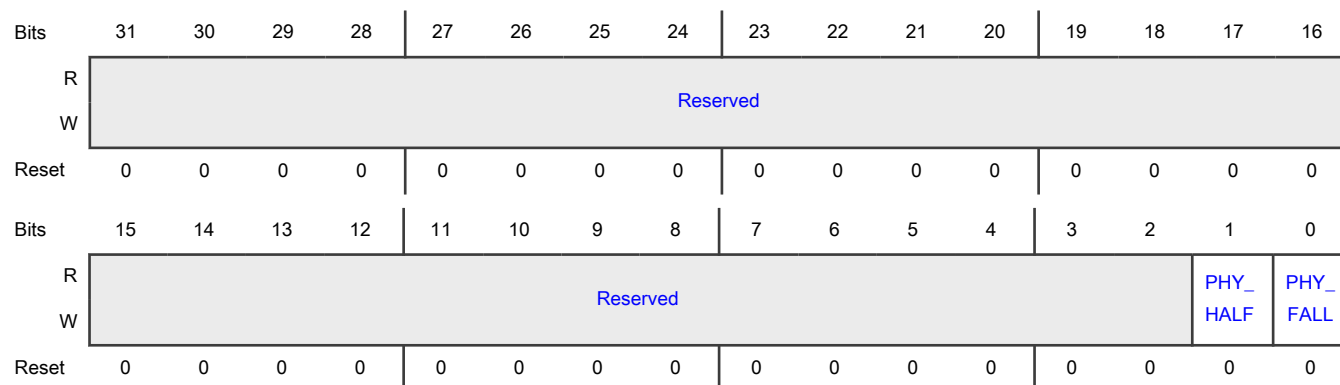
For n = 0 to 7:

Register	Offset
PHY_CTRLn	8Ch + (n × 100h)

Function

The PHY_CTRLn register configures how the PDM source signals are interpreted.

Diagram



Fields

Field	Function
31-2 —	Reserved Read value is undefined, only zero should be written.
1 PHY_HALF	Use Half rate sampling (ie Clock to dmic is sent at half the speed than the decimator is providing) 0b - Standard half rate sampling. The clock to the DMIC is sent at the same rate as the decimator is providing.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Use half rate sampling. The clock to the DMIC is sent at half the rate that the decimator is providing.
0 PHY_FALL	Capture DMIC on Falling edge (0 means on rising) 0b - Capture PDM_DATA on the rising edge of PDM_CLK. 1b - Capture PDM_DATA on the falling edge of PDM_CLK.

49.6.1.11 DC Filter Control (DC_CTRL0 - DC_CTRL7)

Offset

For n = 0 to 7:

Register	Offset
DC_CTRLn	90h + (n × 100h)

Function

The DC_CTRLn register controls the DC filter. The frequencies noted for DCPole in the table assume a PCM output frequency of 16 kHz. If the actual PCM output frequency is 8 kHz, for example, the noted frequencies would be divided by 2.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								SIGNEXTEND	SATURATE	DCGAIN				Reserved	DCPOLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-10 —	Reserved Read value is undefined, only zero should be written.
9 SIGNEXTEND	Sign Extend SIGNEXTEND allows processing of 24-bit audio data on 32-bit machines.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled. The top byte of the FIFODATA register is always 0. 1b - Enabled. The top byte of the FIFODATA register is sign extended.
8 SATURATEAT1 6BIT	Saturate at 16 Bit Selects 16-bit saturation. 0b - Do not Saturate. Results roll over if out range and do not saturate. 1b - Saturate. If the result overflows, it saturates at 0xFFFF for positive overflow and 0x8000 for negative overflow.
7-4 DCGAIN	DC Gain Fine gain adjustment in the form of a number of bits to downshift.
3-2 —	Reserved Read value is undefined, only zero should be written.
1-0 DCPOLE	DC Block Filter 00b - Flat Response, no filter 01b - 155 Hz 10b - 78 Hz 11b - 39 Hz

49.6.1.12 Channel Enable (CHANEN)

Offset

Register	Offset
CHANEN	F00h

Function

The CHANEN register allows enabling selected PDM channels.

The CHANEN register can be used to start multiple channels at the same time by setting more than one channel enable bit with a single write. In addition to synchronizing multiple channels, this allows a DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								EN_	EN_	EN_	EN_	EN_	EN_	EN_	EN_
W									CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 —	Reserved Read value is undefined, only zero should be written.
7-0 EN_CHn	Enable Channel n 0b - PDM channel n is disabled. 1b - PDM channel n is enabled.

49.6.1.13 Use 2 FS register (USE2FS)**Offset**

Register	Offset
USE2FS	F10h

Function

The USE2FS register allows selecting 2 FS output rather than 1 FS output.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															USE2F S
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-1 —	Reserved Read value is undefined, only zero should be written.
0 USE2FS	Use 2FS register 0b - Use 1 FS output for PCM data. 1b - Use 2 FS output for PCM data.

49.6.1.14 Global Channel Synchronization Enable (GLOBAL_SYNC_EN)

Offset

Register	Offset
GLOBAL_SYNC_EN	F14h

Function

This register chooses which channels to synchronize to global sync.

Diagram

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

Fields

Field	Function
31-8 —	Reserved Read value is undefined, only zero should be written.
7-0 CH_SYNC_EN	Channel synch enable Each bit controls the corresponding DMIC channel. Note that the channels should always work in pairs, if both channels in a decimator pair (for instance channels 0 and 1) are used, the same choice should be made for the both of them.

49.6.1.15 Global channel synchronization counter value (GLOBAL_COUNT_VAL)

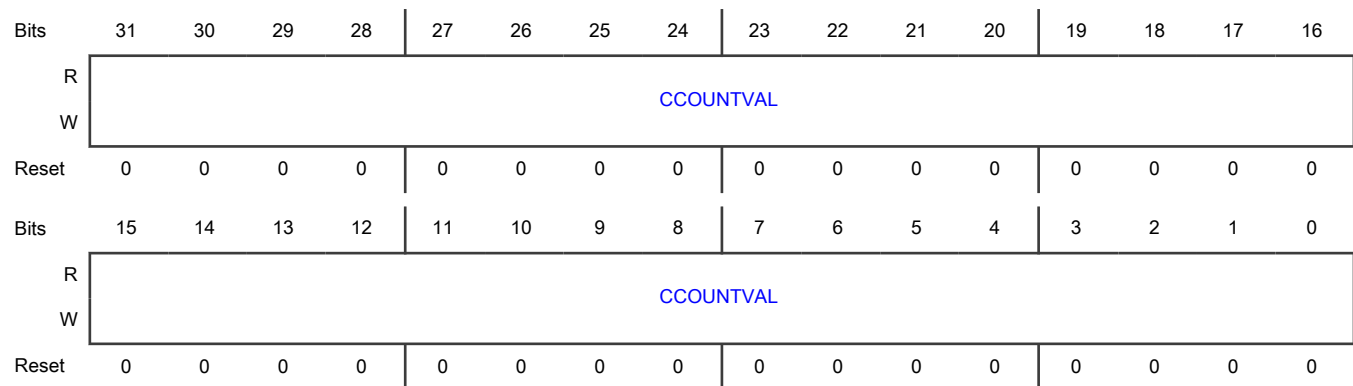
Offset

Register	Offset
GLOBAL_COUNT_VAL	F18h

Function

This register determines when global channel synchronization occurs.

Diagram



Fields

Field	Function
31-0	Channel Counter Value
CCOUNTVAL	The global sync counter will trigger a pulse whenever count reaches CCOUNTVAL. If CCOUNTVAL is set to 0, there will be a pulse on every cycle.

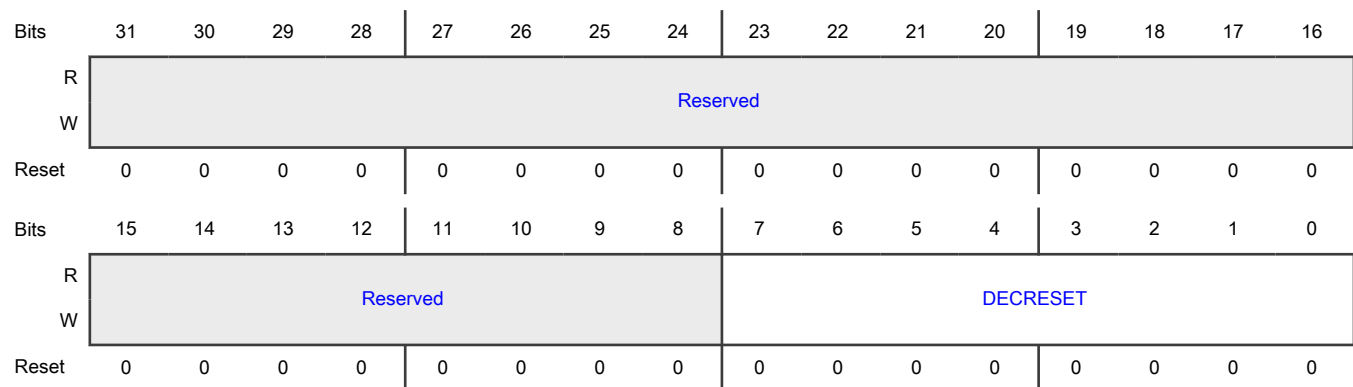
49.6.1.16 DMIC decimator reset (DECRESET)

Offset

Register	Offset
DECRESET	F1Ch

Function

This register allows resetting DMIC decimators.

Diagram**Fields**

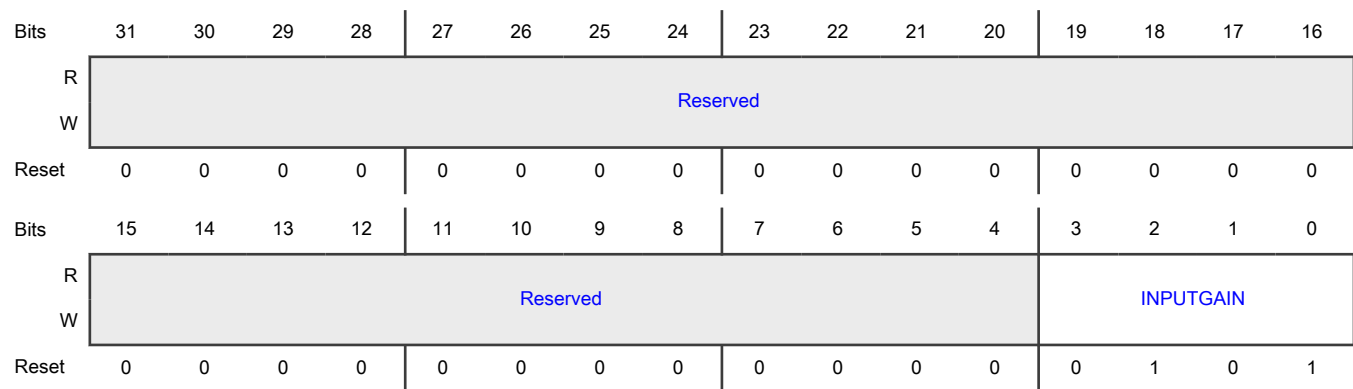
Field	Function
31-8 —	Reserved
7-0 DECRESET	Decimator reset Allows resetting DMIC decimators. Resets are applied in pairs. Bit 0 controls the decimator for channels 0 and 1. Bit 1 controls the decimator for channels 2 and 3. Bit 2 controls the decimator for channels 4 and 5. Bit 3 controls the decimator for channels 6 and 7. Bits 4 to 7 are not used. 0000_0000b - Disable. Release reset to decimator 0000_0001b - Enable. Assert reset to decimator

49.6.1.17 HWVAD Input Gain (HWVADGAIN)**Offset**

Register	Offset
HWVADGAIN	F80h

Function

The HWVADGAIN register controls the input gain of the HWVAD.

Diagram**Fields**

Field	Function
31-4	Reserved
—	Read value is undefined, only zero should be written.
3-0 INPUTGAIN	Input Gain Shift value for input bits 0000b - -10 bits 0001b - -8 bits 0010b - -6 bits 0011b - -4 bits 0100b - -2 bits 0101b - 0 bits (default) 0110b - +2 bits 0111b - +4 bits 1000b - +6 bits 1001b - +8 bits 1010b - +10 bits 1011b - +12 bits 1100b - +14 bits 1101b-1111b - Reserved

49.6.1.18 HWVAD Filter Control (HWVADHPFS)**Offset**

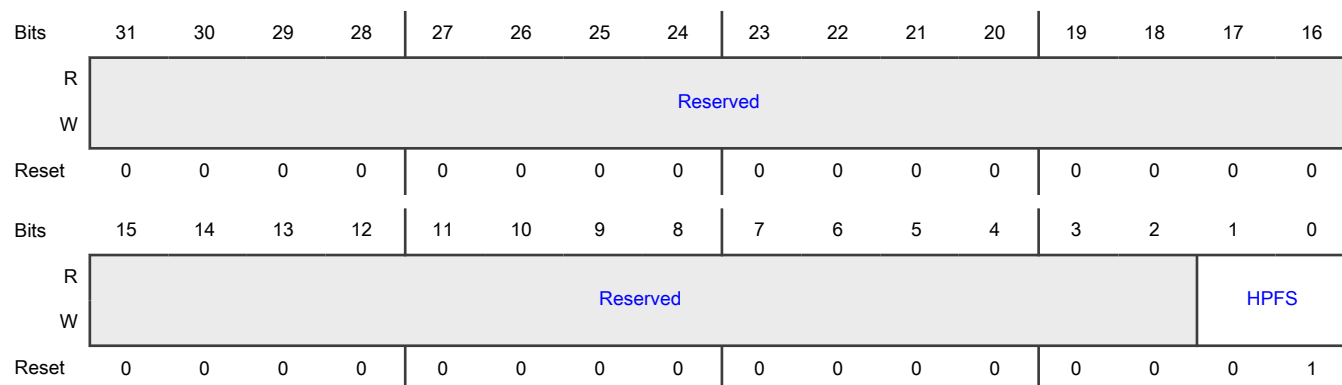
Register	Offset
HWVADHPFS	F84h

Function

This HPFS filter setting parameter can be used to optimize performance for different background noise situations. To find the best setting, software needs to perform a rough spectral analysis of the audio signal.

Rule of thumb: If the amount of low-frequency content in the background noise is small, then HPFS = 0x2, else HPRS = 0x1.

Diagram



Fields

Field	Function
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 HPFS	The HPFS field chooses the High Pass filter in first part of HWVAD. 00b - Bypass. BYPASS. First filter by-pass. 01b - High Pass 1750 Hz. HIGH_PASS_1750HZ. High pass filter with -3dB cut-off at 1750 Hz. 10b - High Pass 215 Hz. HIGH_PASS_215HZ. High pass filter with -3dB cut-off at 215 Hz. 11b - Reserved

49.6.1.19 HWVAD Control (HWVADST10)

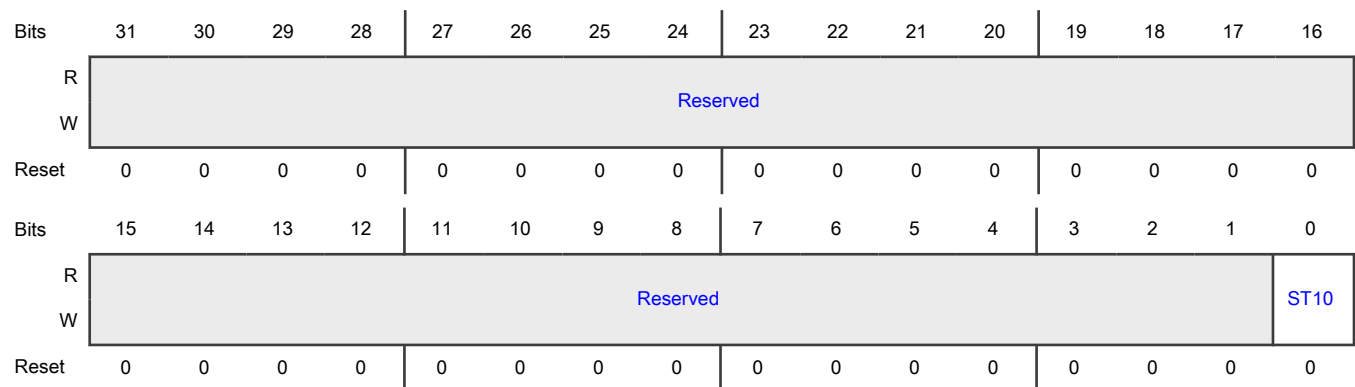
Offset

Register	Offset
HWVADST10	F88h

Function

The HWVADST10 register controls the operation of the filter block and resets the internal interrupt flag. After the HWVAD triggers an interrupt, a short '1' pulse on bit ST10 clears the interrupt.

Keeping the ST10 bit set to 1 for some time has a special function for filter convergence, see more information in [HWVAD](#)

Diagram**Fields**

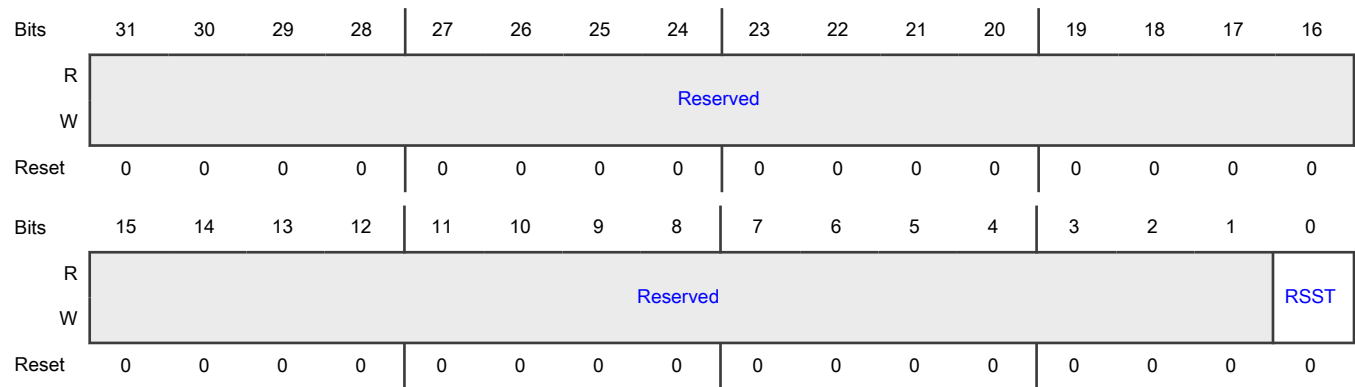
Field	Function
31-1	Reserved
—	Read value is undefined, only zero should be written.
0	STAGE 1
ST10	'1' means enter stage 1 of VAD, that is, a sound change has been detected and the HWVAD is being allowed to settle. Use 0 when changing back to detection mode. Allow several milliseconds in stage 1 for settling. 0b - Normal operation, waiting for HWVAD trigger event (stage 0). 1b - Reset internal interrupt flag by writing a '1' (stage 1) pulse.

49.6.1.20 HWVAD Filter Reset (HWVADRSTT)**Offset**

Register	Offset
HWVADRSTT	F8Ch

Function

Setting bit RSTT = 1 causes a synchronous reset of all filters inside the HWVAD. The RSTT bit must be cleared to allow HWVAD operation. See [HWVAD](#)

Diagram**Fields**

Field	Function
31-1	Reserved
—	Read value is undefined, only zero should be written.
0	Reset HWVAD
RSST	Write back to 0 to release reset.

49.6.1.21 HWVAD Noise Estimator Gain (HWVADTHGN)**Offset**

Register	Offset
HWVADTHGN	F90h

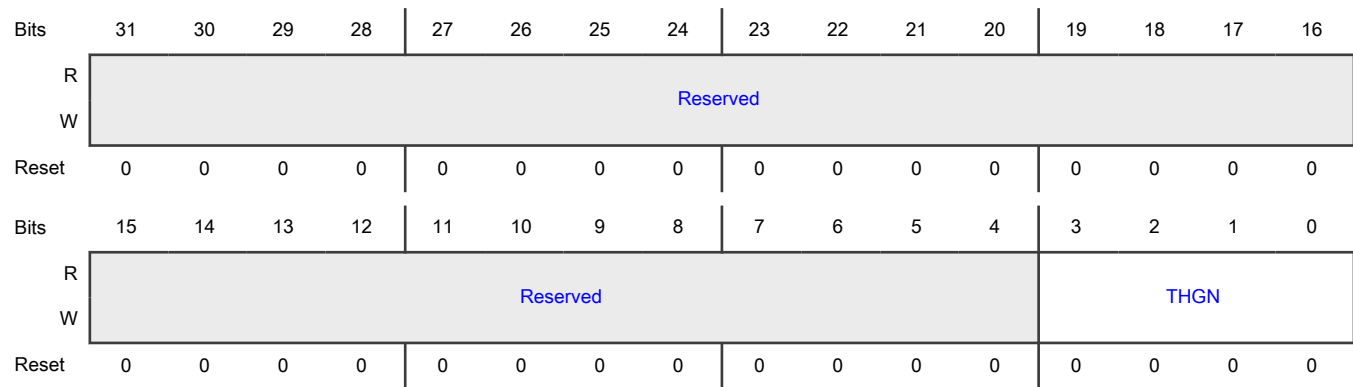
Function

The HWVADTHGN register shows the gain value for the noise estimator value; the THGN parameter is used in the following calculation (implemented in hardware):

```

if z8 * (THGS+1) > z7 * (THGN+1)
    HWVAD_RESULT = 1;
else
    HWVAD_RESULT = 0;

```

Diagram**Fields**

Field	Function
31-4	Reserved
—	Read value is undefined, only zero should be written.
3-0	Gain Factor for Noise Estimator
THGN	0 to 14: 0 corresponds to a gain of 1.

49.6.1.22 HWVAD Signal Estimator Gain (HWVADTHGS)**Offset**

Register	Offset
HWVADTHGS	F94h

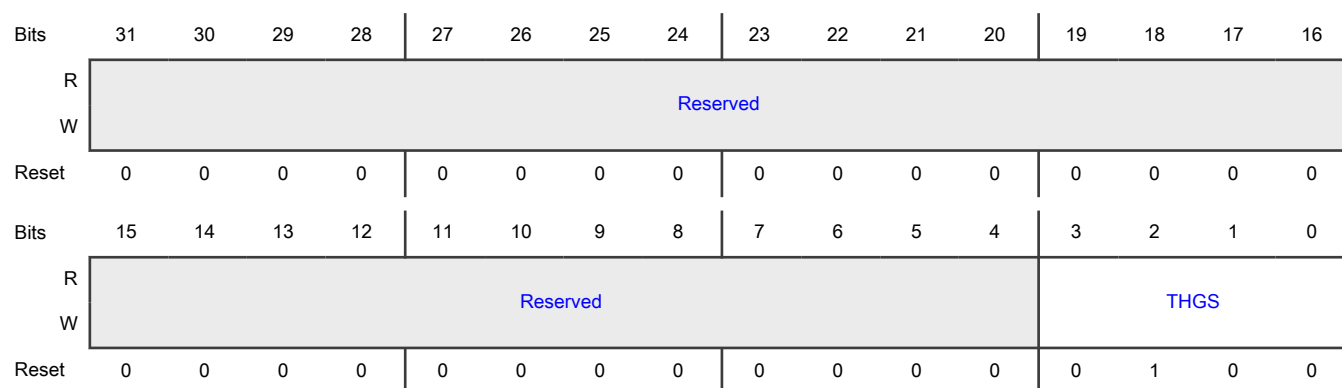
Function

The HWVADTHGS register shows the gain value for the signal estimator value; the THGS parameter used in the following calculation (implemented in hardware):

```

if z8 * (THGS+1) > z7 * (THGN+1)
    HWVAD_RESULT = 1;
else
    HWVAD_RESULT = 0;

```


Diagram**Fields**

Field	Function
31-4	Reserved
—	Read value is undefined, only zero should be written.
3-0	Signal Gain Factor
THGS	0 to 14: 0 corresponds to a gain of 1.

49.6.1.23 HWVAD Noise Envelope Estimator (HWVADLOWZ)**Offset**

Register	Offset
HWVADLOWZ	F98h

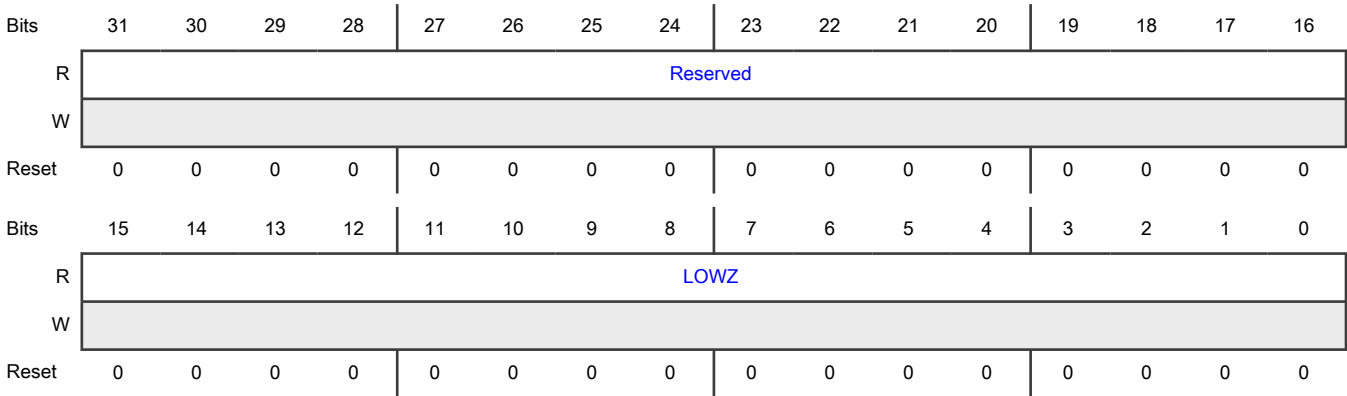
Function

The HWVADLOWZ register contains 2 bytes of the output of filter stage z7. It can be used as an indication for the noise floor and must be evaluated by software. See [HWVAD](#)

NOTE

For power saving reasons the HWVADLOWZ register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the HWVADLOWZ register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read is not be in the middle of a transition.

Diagram



Fields

Field	Function
31-16	Reserved
—	Read value is undefined, only zero should be written.
15-0 LOWZ	Average Noise-floor Value

Chapter 50

Ultra Secured Digital Host Controller (uSDHC)

50.1 Chip-specific uSDHC information

NOTE

uSDHC is also called as SDIO throughout the document.

Table 497. Reference links to related information

Topic	Related module	Reference
Full description	uSDHC	uSDHC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Initial configuration of the uSDHC interface can be accomplished as follows:

- Enable the clock to the uSDHC interface in the PSCCTL1 register. This enables the register interface and the peripheral function clock.
- Select a clock source for the uSDHC interface using the SDIO0FCLKSEL register.
- Select a clock divide for the uSDHC interface using the SDIO0FCLKDIV register.
- Clear the uSDHC interface peripheral reset in the PRSTCTL1 register by writing to the PRSTCTL1_CLR register.
- Enable the array and periphery power of the uSDHC interface RAMs using the PDRUNCFG1 register.
- The uSDHC interface provides an interrupt to the NVIC (see [Nested vector interrupts](#)). To allow interrupts to wake-up the device from deep-sleep mode, enable this in the STARTEN1 register.
- Use the IOPCTL registers to connect the uSDHC interface (inputs/outputs) to external pins.
- The uSDHC interface has an AHB master function. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the AHBMATRIXPRIOR register.

50.1.1 Module instances

This device has two instances of the uSDHC module, USDHC0 and USDHC1. Only USDHC0 supports HS400 mode.

50.2 Introduction

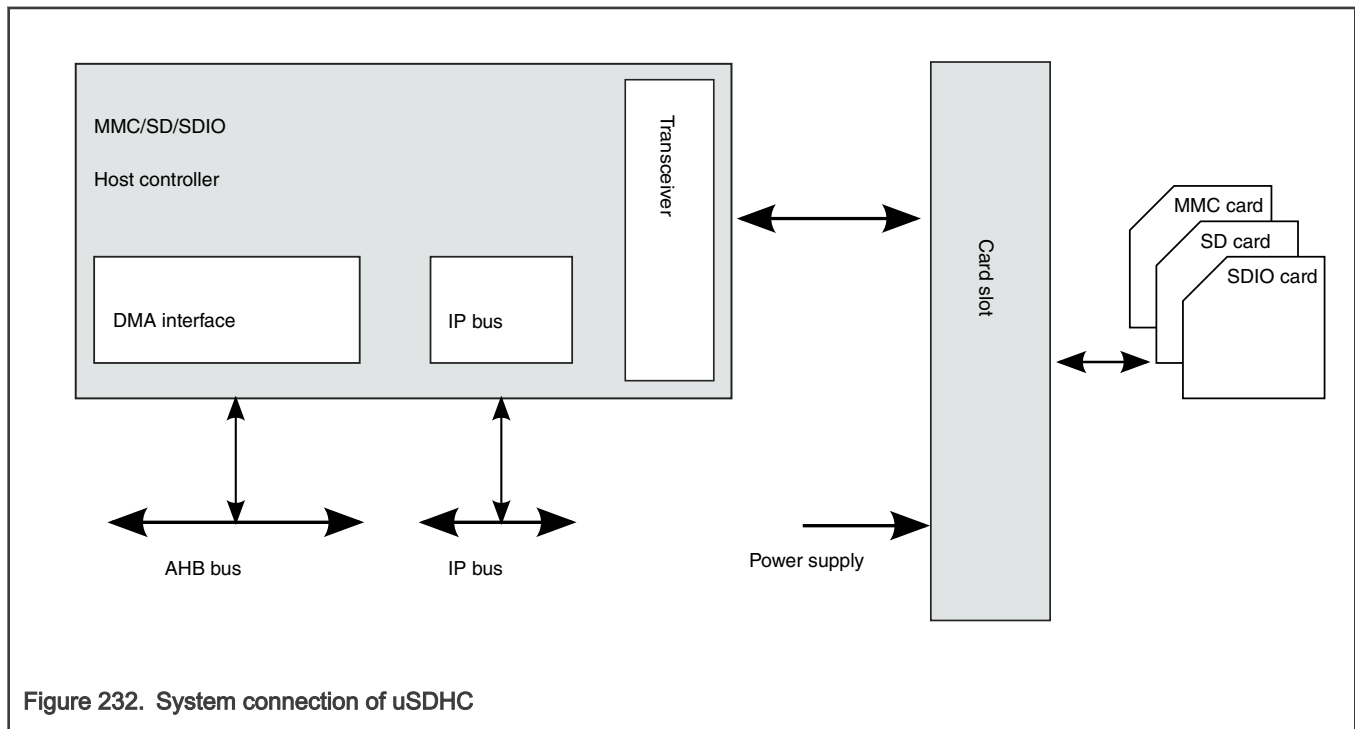
uSDHC provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in [Figure 232](#). uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by uSDHC:

The Multi Media Card (MMC) is a universal low-cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low-voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into memory card, I/O card, and combo card, which have both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low-power consumption for mobile electronic devices. For the sake of simplicity, the next figure does not show cards with reduced size or mini cards.



The following figure illustrates the block diagram of uSDHC.

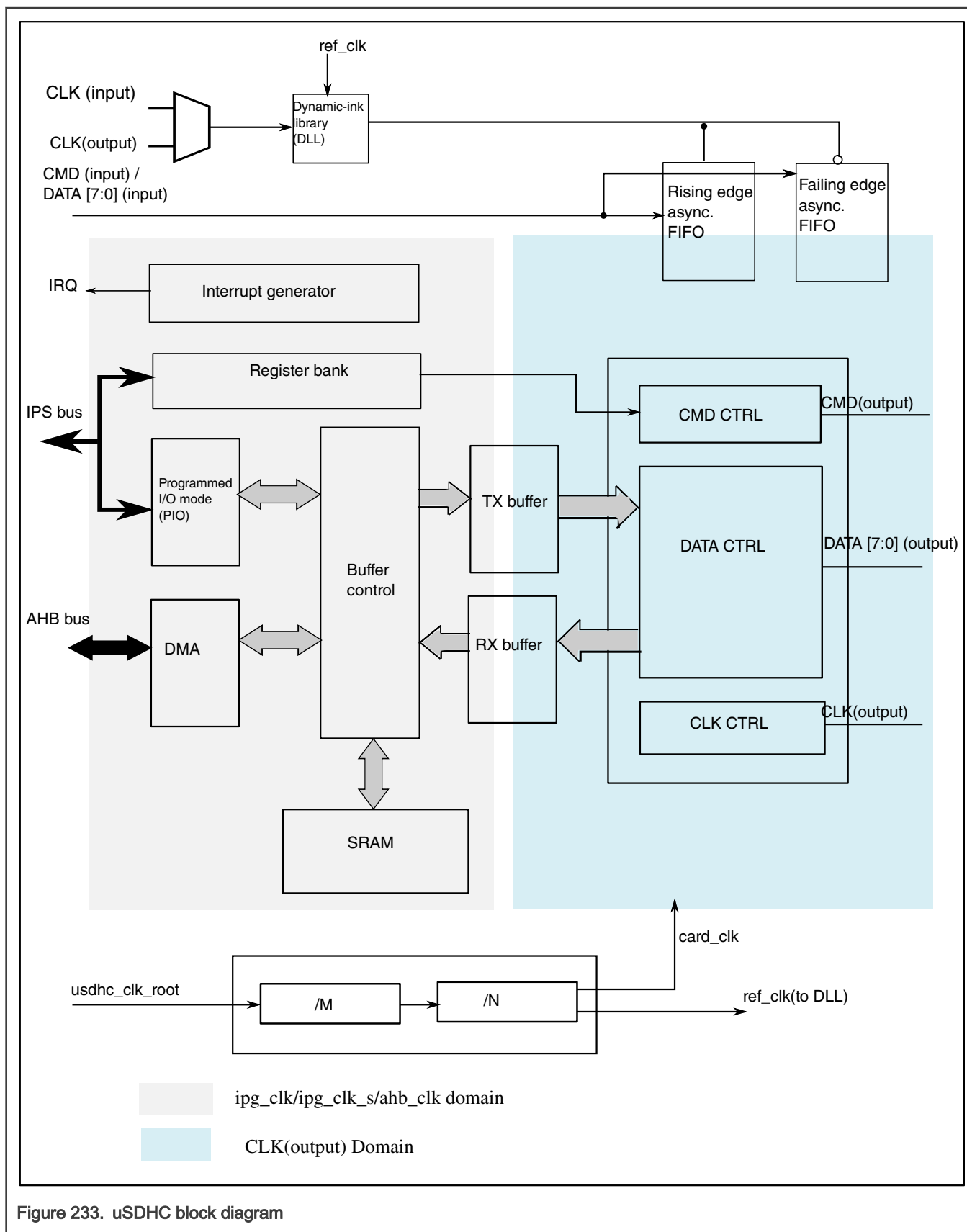


Figure 233. uSDHC block diagram

50.2.1 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0/3.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5/5.0
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 2.0/3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 208 MHz
- Supports 1-bit/4-bit SD and SDIO modes, and 1-bit/4-bit/8-bit MMC modes
 - Up to 832 Mbps of data transfer for SDIO cards using four parallel data lines in the Single Data Rate (SDR) mode
 - Up to 400 Mbps of data transfer for SDIO card using four parallel data lines in the Dual Data Rate (DDR) mode
 - Up to 832 Mbps of data transfer for SDXC cards using four parallel data lines in the Single Data Rate (SDR) mode
 - Up to 400 Mbps of data transfer for SDXC card using four parallel data lines in the Dual Data Rate (DDR) mode
 - Up to 1600 Mbps of data transfer for MMC cards using eight parallel data lines in the Single Data Rate (SDR) mode
 - Up to 3200 Mbps of data transfer for MMC cards using eight parallel data lines in the Dual Data Rate (DDR) mode
- Supports a single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes; also supports interrupt period
- Embodies a fully configurable 256x32-bit FIFO for read/write data
- Supports internal DMA capabilities
- Support voltage selection by configuring vendor-specific register bit
- Supports Advanced DMA to perform linked memory access

NOTE

This block can support all the above listed speed mode and maximum data throughput. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

50.2.2 Modes and operations

50.2.2.1 Data transfer modes

The uSDHC module can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit

- MMC 4-bit
- MMC 8-bit
- Identification mode (up to 400 kHz)
- MMC full-speed mode (up to 26 MHz)
- MMC high-speed mode (up to 52 MHz)
- MMC HS200 mode (up to 200 MHz)
- MMC HS400 mode (200 MHz both edges)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full-speed mode (up to 25 MHz)
- SD/SDIO high-speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode (up to 208 MHz in SDR mode, up to 50 MHz in the DDR mode)

NOTE

This block can support all the above listed speed mode and maximum clock frequency. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

50.3 External signals

The following table describes the external signals of uSDHC:

Table 498. uSDHC external signals

Signal	Description	Direction
CLK	Clock for MMC/SD/SDIO card	O
CMD	CMD line connect to card	I/O
DATA7	DAT7 line in the 8-bit mode — Not used in other modes	I/O
DATA6	DAT6 line in the 8-bit mode — Not used in other modes	I/O
DATA5	DAT5 line in the 8-bit mode — Not used in other modes	I/O
DATA4	DAT4 line in the 8-bit mode — Not used in other modes	I/O
DATA3	DAT3 line in the 4/8-bit mode or configured as card detection pin. The bit may be configured as card detection pin in the 1-bit mode.	I/O
DATA2	DAT2 line or Read Wait in the 4-bit mode Read Wait in 1-bit mode	I/O
DATA1	DAT1 line in the 4/8-bit mode	I/O

Table continues on the next page...

Table 498. uSDHC external signals (continued)

Signal	Description	Direction
	Also, used to detect interrupt in 1/4-bit mode	
DATA0	DAT0 line in all the modes Also, used to detect busy state	I/O
CD_B	Card detection pin If not used (for the embedded memory), tie low to indicate that there is a card attached.	I
WP	Card write protect detect If not used (for the embedded memory), tie low to indicate that it is not write protected.	I
RESET_B	Card hardware reset signal, active low	O
VSELECT	IO power voltage selection signal	O
STROBE	Input clock for eMMC HS400 mode	I

50.3.1 Signals overview

uSDHC has 15 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, and SDIO cards.
- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC module and the card.
- The CD and WP are card detection and write protection signals directly routed from the socket. These two signals are active low (0). A low on CD# means that a card is inserted, and a high on WP means that the write protect switch is active.
- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier.

CD, WP, RST, and VSELECT are all optional for system implementation. If uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high. If the uSDHC does not support the HS400 mode, STROBE can also be optional and tied to low.

50.4 Clocks

The table found here describes the clock sources for uSDHC. For more information on clocking, see the Clocking chapter.

Table 499. uSDHC clocks

Clock name	Clock root	Description
hclk	ahb_clk_root	AHB bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_perclk	usdhc_clk_root	Base clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

50.5 Functional description

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock and reset manager, and clock generator.

50.5.1 Data buffer

The uSDHC module uses one configurable data buffer to transfer data between the system bus (IP bus or advanced high-performance bus (AHB) bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock). The buffer is used as a temporary storage for transferring data between the host system and the card. The watermark levels for read and write are both configurable and can range between 1 to 128 words. The burst lengths for read and write are also configurable and can range between 1 to 31 words. The next figure provides the uSDHC buffer scheme.

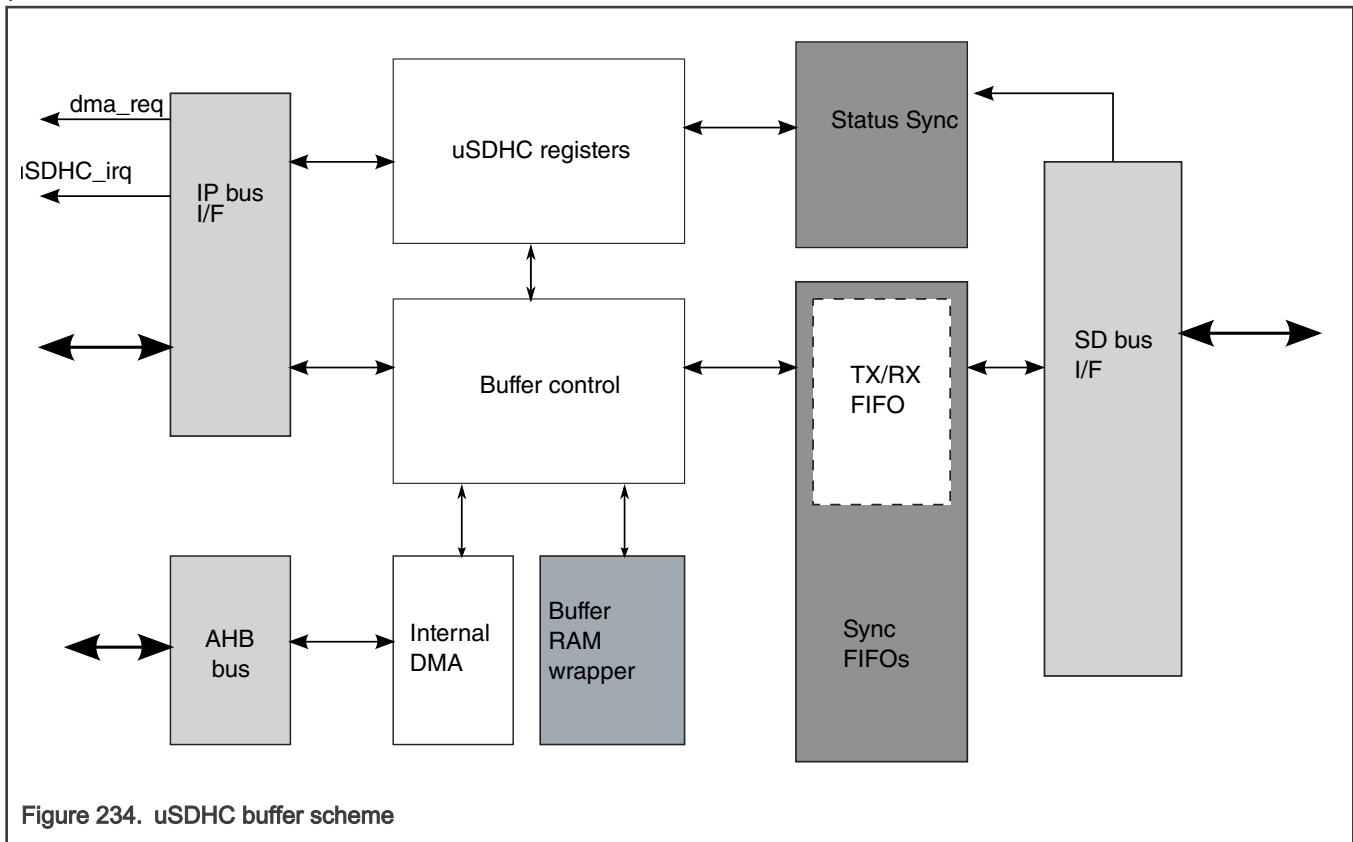


Figure 234. uSDHC buffer scheme

Here are 2 the two transfer modes to access the data buffer:

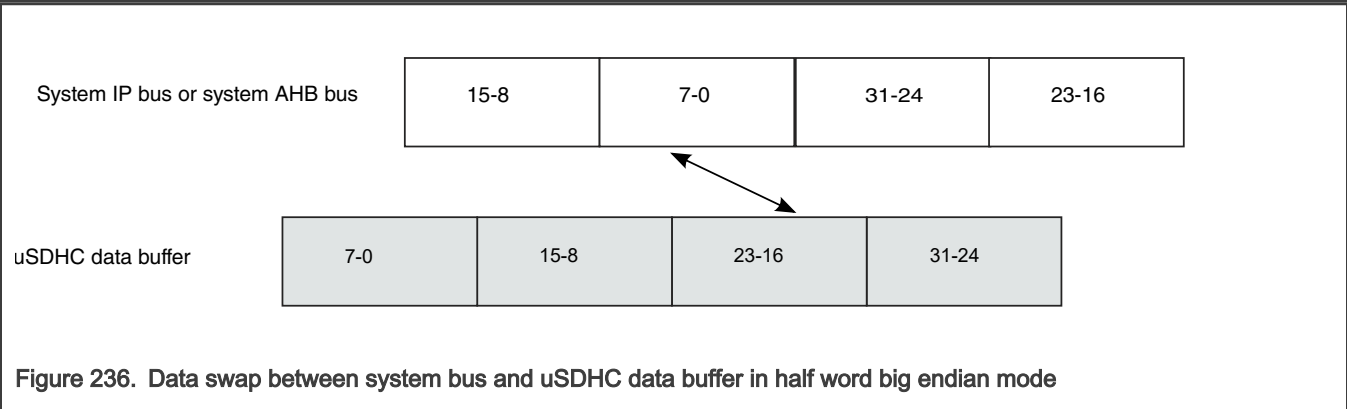
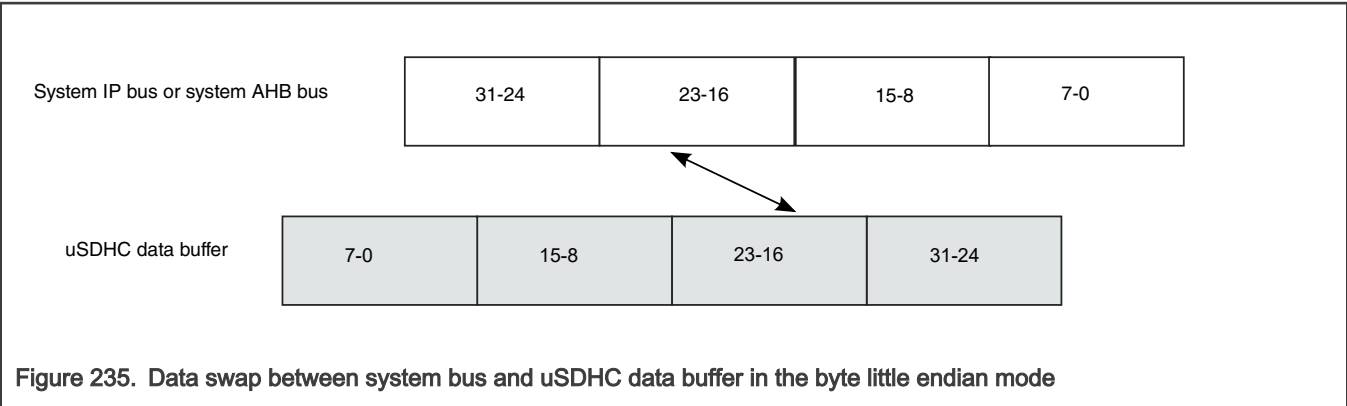
- CPU polling mode:
 - For a host-read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the host driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar. For more information on the process of writing operation, see [Write operation sequence](#).
- Internal DMA mode (includes simple and advanced DMA accesses):
 - The internal DMA access, either by simple or advanced DMA, is over the AHB bus.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always the INCR mode and the burst length depends on the shortest of the following factors:

- Burst length configured in the burst length field of the Watermark Level register

- Watermark level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 KB address boundary defined in the AHB protocol

The Write operation functions in a similar manner—sequential and contiguous access is necessary to ensure that the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, the byte order is swapped after the data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, the byte order is swapped before the data is stored in the buffer.



50.5.1.1 Write operation sequence

There are 2 ways to write data into the buffer when the user transfers data to the card:

- Processor core polling through the BWR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, the DMAEN bit in the Transfer Type register is not set when the command is sent, uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register and is ready for receiving new data. At the same time, uSDHC sets the BWR bit. The buffer write ready interrupt is generated if it is enabled by software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the write operation from that address. It is recommended that a software reset for Data be applied before the transfer is restarted.

The uSDHC module does not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the host system does not write data in time, uSDHC stops the CLK to avoid the data buffer underrun situation.

50.5.1.2 Read operation sequence

There are 2 ways to read data from the buffer when the user transfers data to the card:

- Processor core polling through the BRR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, which is available and ready for system fetching data. At the same time, uSDHC sets the BRR bit. The buffer read ready interrupt is generated if it is enabled by the software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the read operation from that address. It is recommended that a software reset for data be applied before the transfer is restarted.

For any write transfer mode, uSDHC does not start data transmission until the number of words set in the RD_WML register are in buffer. If the buffer is full and the host system does not read data in time, uSDHC stops the CLK to avoid the data buffer overrun situation.

50.5.1.3 Data buffer and block size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC module, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly. For both read and write, the watermark level can range between 1 to 128 words. For both read and write, the burst length can range between from 1 to 31 words. The host driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned), stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write twice for each block. For each block, the ending byte is abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

50.5.1.4 Dividing large data transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly into a multiple of the block size, then there are two ways to transfer the data. These two ways depend on the function and the card design. Option 1 is for the host driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the division of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size of up to 64 bytes. Although uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size of less than 64 bytes, so the data must be divided (see the example below).

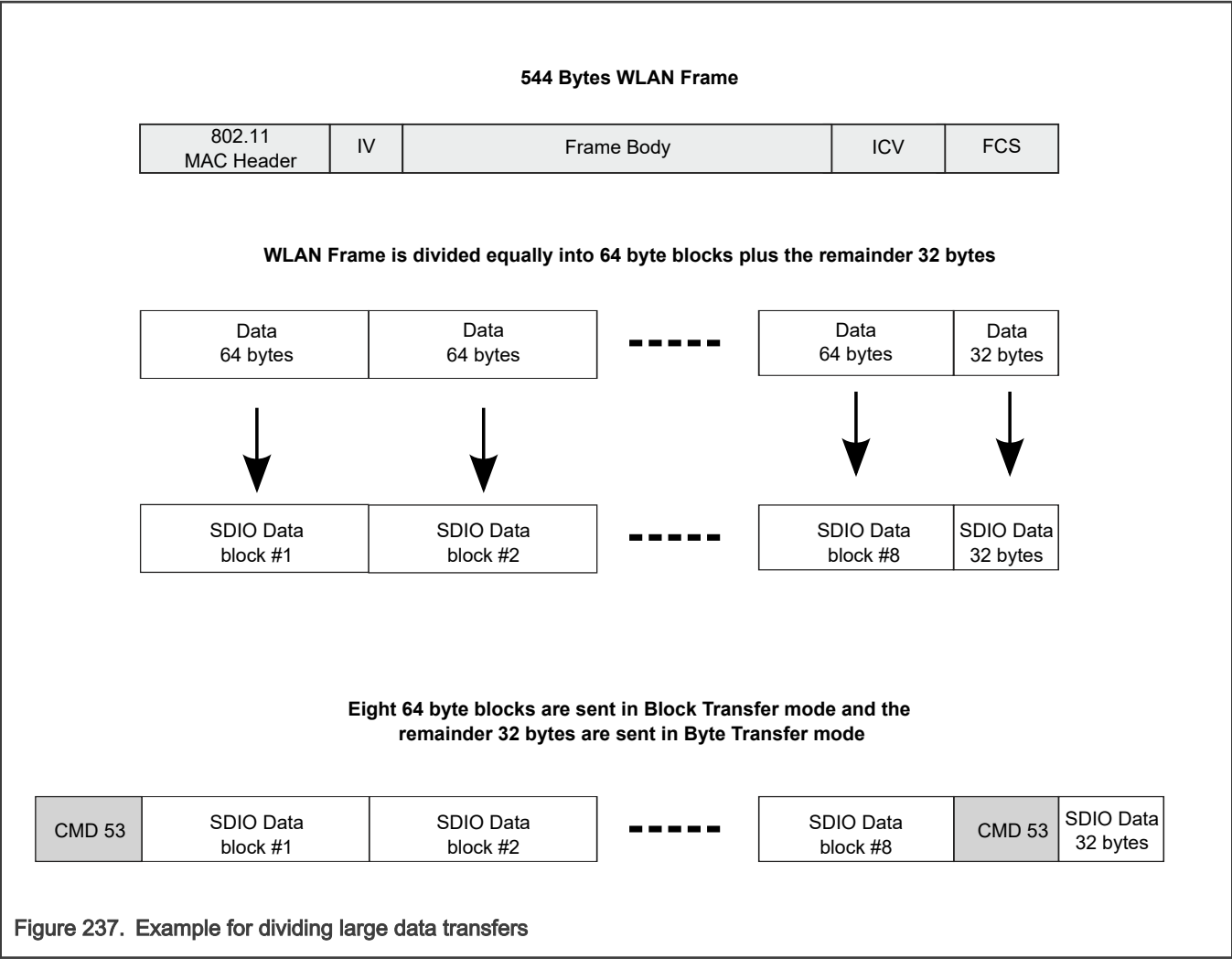
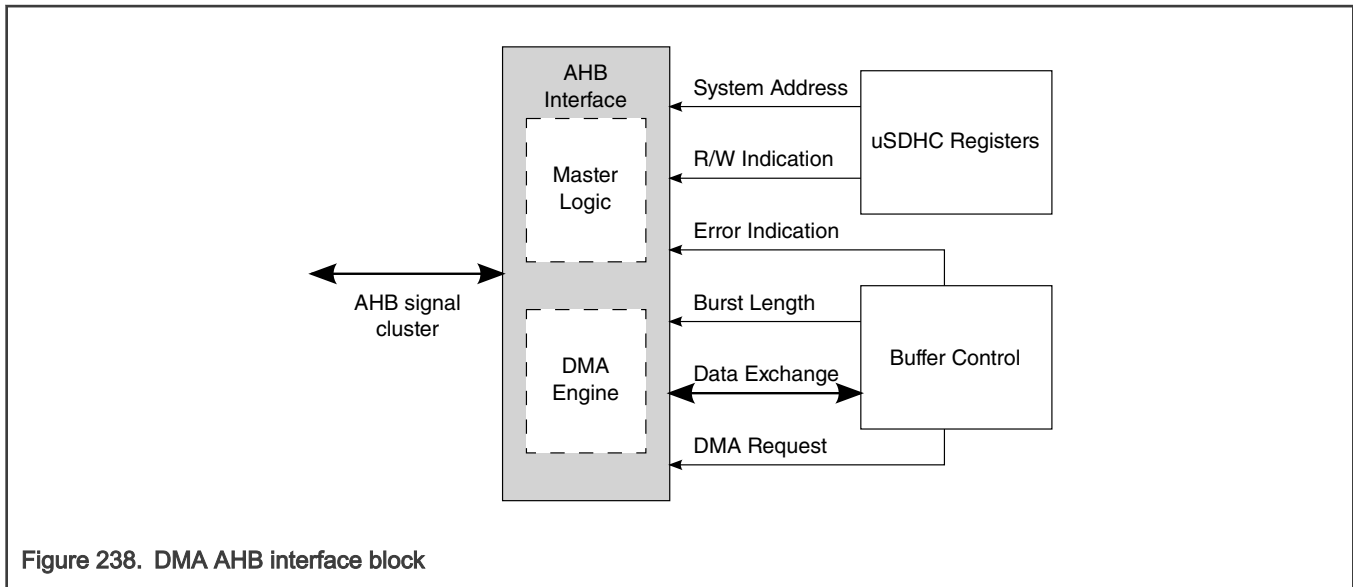


Figure 237. Example for dividing large data transfers

50.5.2 DMA AHB interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b is not asserted during the transfer, but the BWR and BRR bits are set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register. See the figure below for an illustration of the DMA AHB interface block.



50.5.2.1 Internal DMA request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the data buffer block sends a DMA request to AHB interface. The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The data buffer de-asserts the request if the data buffer space (for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the data buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of the current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to six words. After the first burst transfer, if there are more than two words in the buffer, which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is $(31 - 6 * 4) / 4 = 2$. The uSDHC module reads two words out of the buffer, with seven valid bytes and one stuffed byte.

50.5.2.2 DMA burst length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can range between 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. See the example in [Internal DMA request](#). After six words are read, the burst length is two words, then the next burst length is six words. This is because the next block starts, which is 31 bytes, more than six words. The host driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length is the same.

50.5.2.3 AHB master interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state. At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register is generated to host CPU to report a bus error condition.

After the DMAE interrupt is received, the software sends a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and restart the transfer from this address to recover the corrupted block. DMA operation resumes when the interrupt is serviced by the software.

50.5.2.4 ADMA engine

In the SD host controller standard, a new DMA transfer algorithm called the Advanced DMA (ADMA) is defined. For simple DMA, after the page boundary is reached, a DMA interrupt is generated and the new system address is programmed by the host driver. The ADMA defines the programmable descriptor table in the system memory. The host driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized because the host MCU intervention is not needed during long DMA-based data transfers.

There are two types of ADMA in host controller: ADMA1 and ADMA2. ADMA1 can support data transfer of 4KB aligned data in system memory, and ADMA2 improves the restriction so that the data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors defined in SD host controller Standard, and if the "End" flag is detected in the descriptor, ADMA stops after this descriptor is processed.

50.5.2.4.1 ADMA concept and descriptor format

ADMA1 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Set data length descriptor
- Set data address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA2 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Rsv descriptor
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA starts read/write operation after it reaches the tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last set type descriptor before the tran type descriptor. Every tran type triggers a transfer, and the transfer data length is extracted from the most recent set type descriptor. If there is no set type descriptor after the previous Trans descriptor, the data length is the value for previous transfer, or 0 if no set descriptor is ever met.

For ADMA2, the tran type descriptor contains both data length and transfer data address, so only a tran type descriptor can start a data transfer.

See the figure below for the format of the descriptor table for ADMA1.

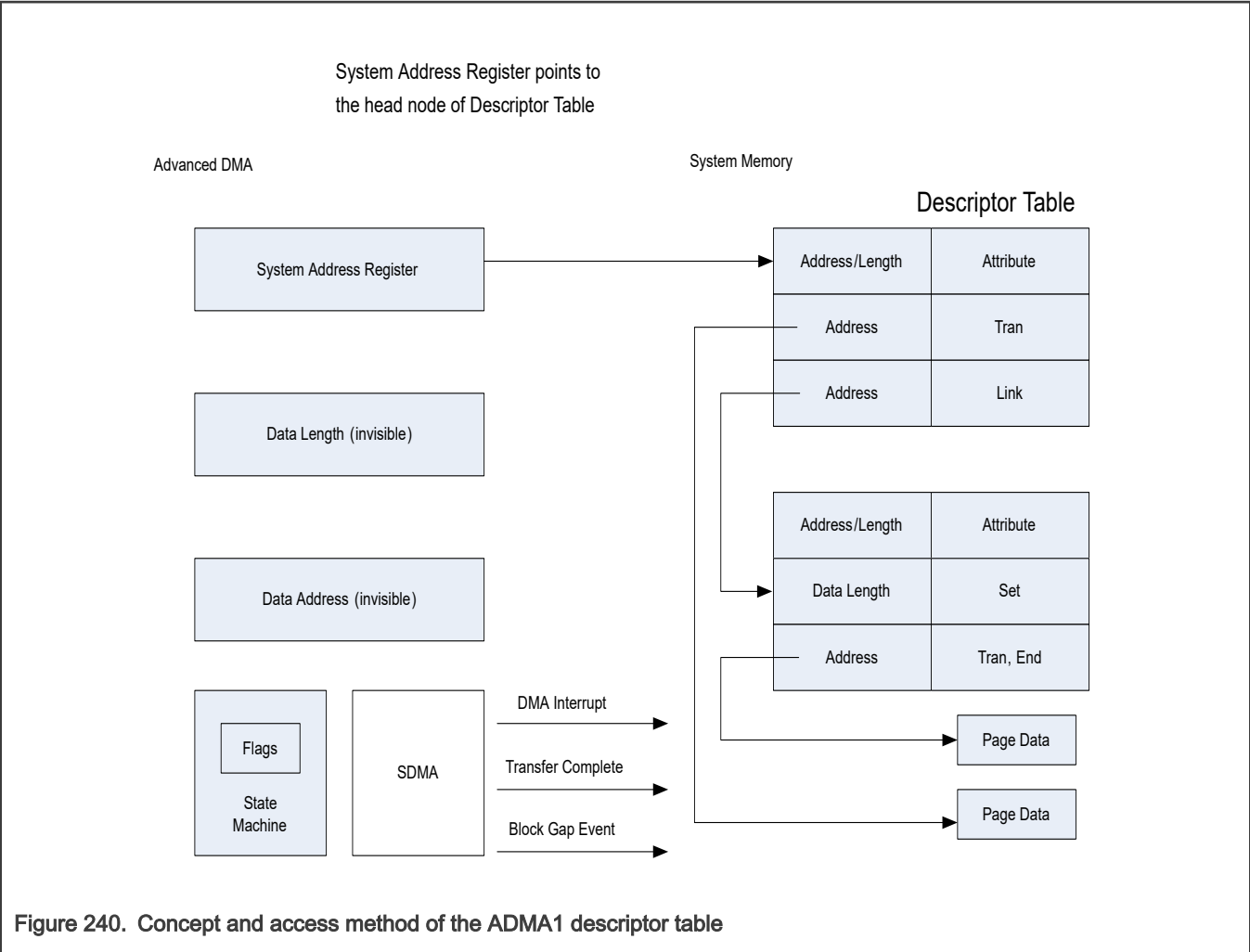
Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 239. Format of the ADMA1 descriptor table

See the figure below for the concept and access method of the descriptor table for ADMA1.



The figure below explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it ignores the higher 32-bit. The Address field should be set to word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

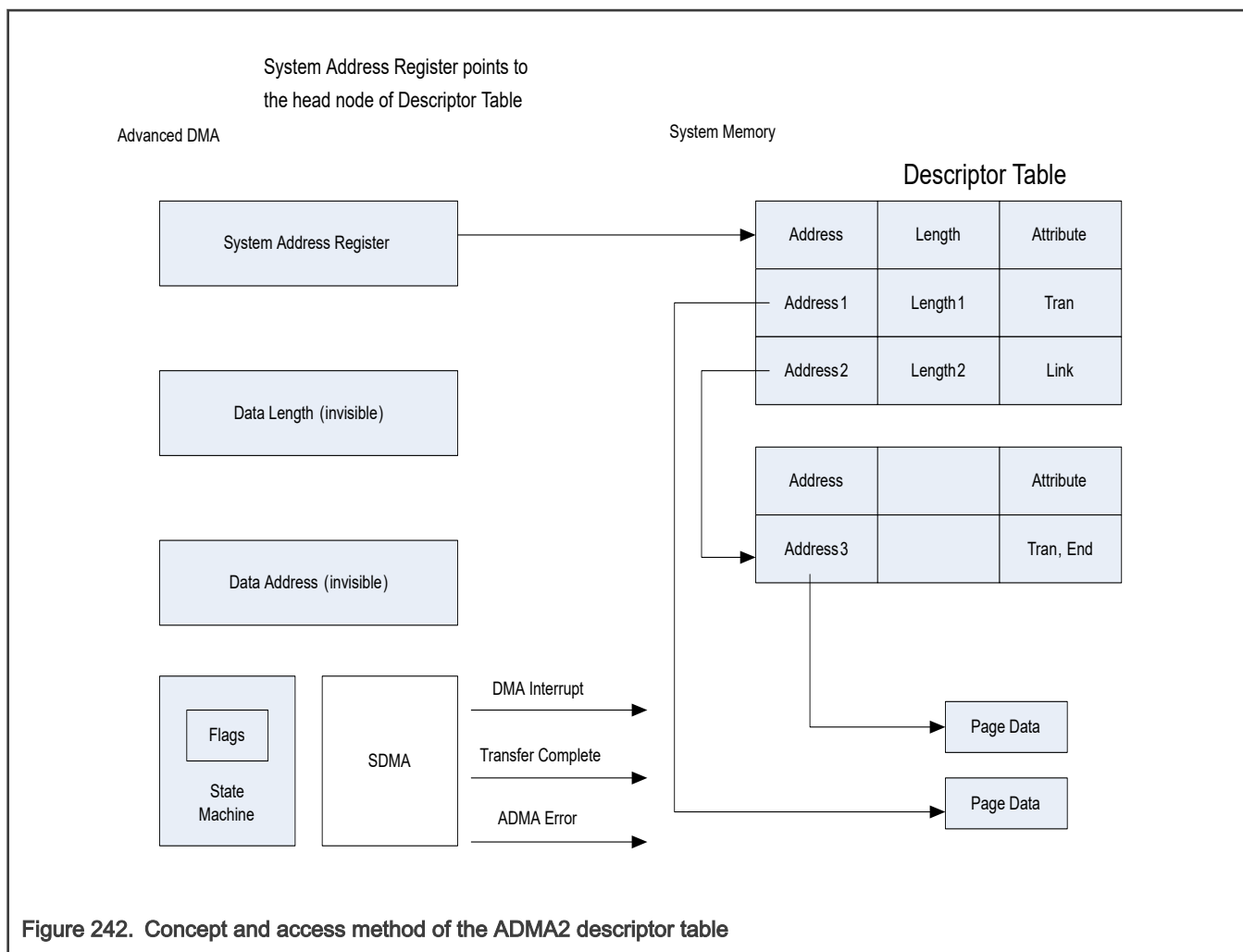
Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 241. Format of the ADMA2 descriptor table

See the figure below for the concept and access method of the descriptor table for ADMA2.



50.5.2.4.2 ADMA interrupt

If the interrupt flag descriptor is set, ADMA generates an interrupt according to various types of descriptors.

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

50.5.2.4.3 ADMA error

The ADMA stops whenever an error is encountered. These errors include:

- Fetching descriptor error

- AHB response error
- Data length mismatch error

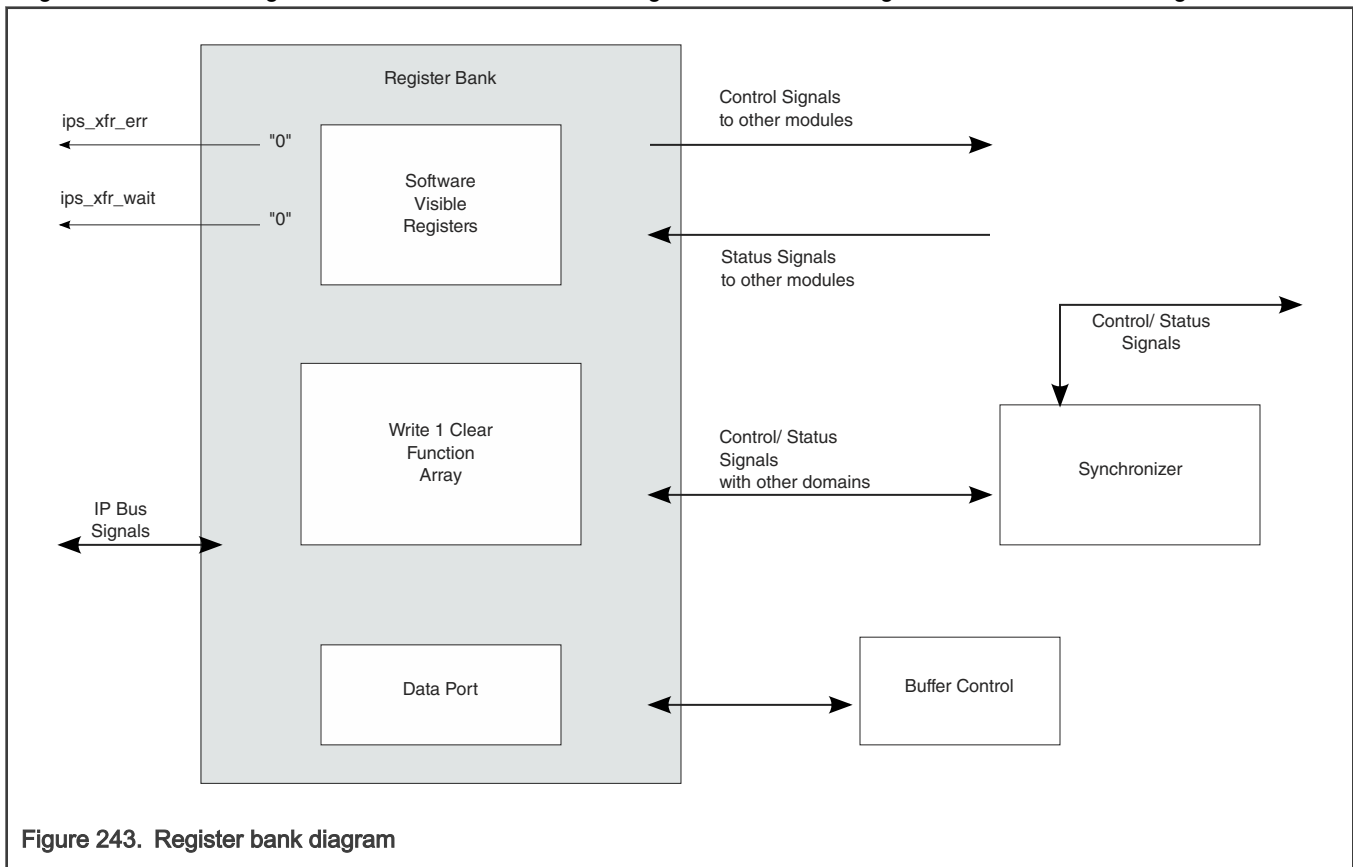
An ADMA descriptor error is generated when it fails to detect a "valid" flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the "Interrupt" flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error is generated.

When BLKCNTEN bit is not set, then the whole data length set in descriptor is a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors occur.

50.5.3 Register bank with IP bus interface

Register accesses through the IP bus interface are on the register bank. See the figure below for the block diagram.



Only a 32-bit access is allowed, and no partial read/write is supported; therefore, all accesses are word aligned.

50.5.3.1 SD protocol unit

The SD protocol unit deals with all SD protocol affairs.

The SD protocol unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card

- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD protocol unit consists of four submodules:

- SD control misc
- Command control
- Data control
- Clock control

50.5.3.2 SD control misc

In the SD control misc unit, the card detection (including the CD_B and DATA3 used for card detection), write protection, and card interrupt are implemented.

This module monitors the signal level on all the eight data lines and the command lines. It directly routes the level values into the register bank.

The module also detects the Write Protect (WP) line. If WP is active, writes to the register bank are ignored.

50.5.3.3 SD clock control

If the internal data buffer is near full (for read) or near empty (for write), the SD clock must be gated off to avoid buffer over/under-run. This module asserts the gate of the output SD clock to shut the clock off. After the buffer has space (for read) or has data (for write), the clock gate of this module opens, and the SD clock is active again.

50.5.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC shift register.

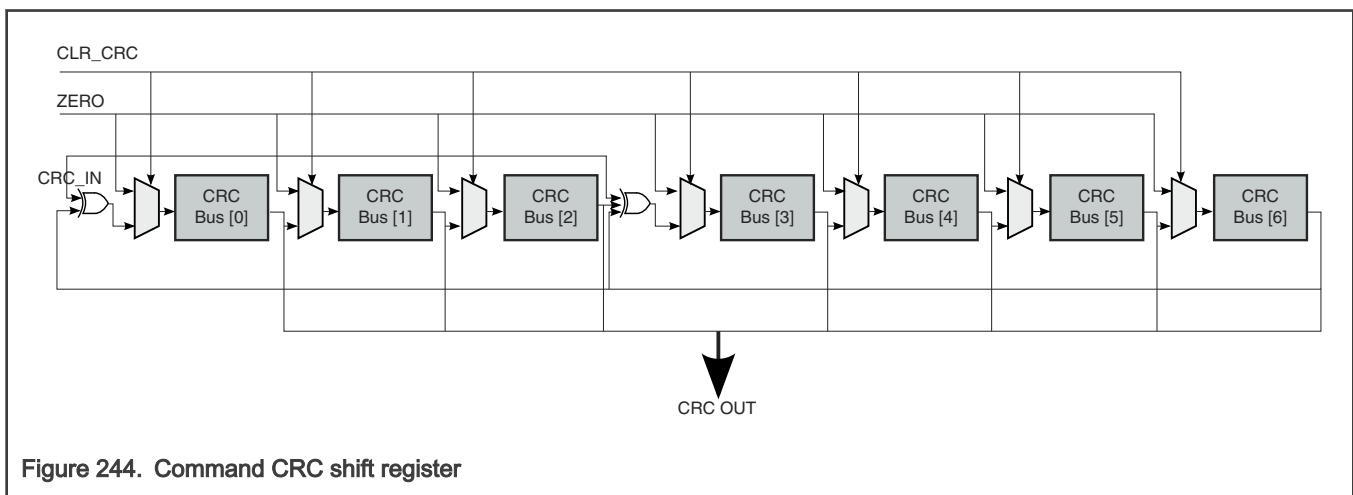


Figure 244. Command CRC shift register

The CRC polynomials for the CMD are as follows:

```
Generator polynomial:  $G(x) = x^7 + x^3 + 1$ 
 $M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \dots + (\text{last bit}) * x^0$ 
CRC[6:0] = Remainder  $[(M(x) * x^7) / G(x)]$ 
```

50.5.3.5 Data control

The data agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the read wait state by the request from the transceiver.

The CRC polynomials for the data are as follows:

```
Generator polynomial:  $G(x) = x^{16} + x^{12} + x^5 + 1$ 
 $M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \dots + (\text{last bit}) * x^0$ 
CRC[15:0] = Remainder [ $(M(x) * x^{16}) / G(x)$ ]
```

50.5.4 Clock and reset manager

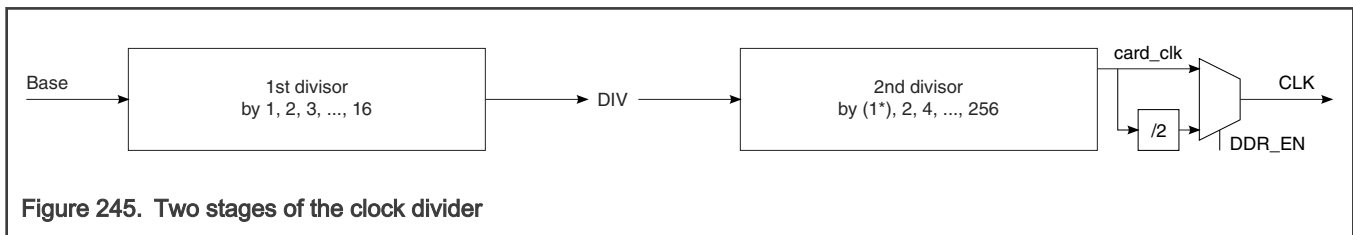
This module controls all four kinds reset signals within uSDHC:

- Hardware reset
- Software reset for all logic
- Software reset for the data logic
- Software reset for the command logic

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

50.5.5 Clock generator

The clock generator generates the card CLK by peripheral source clock in two stages. The clock divisor can be configured through register `SYS_CTRL[SDCLKFS]` is for prescaler configuration while the `[SYS_CTRL[DVS]]` is for divisor configuration. Details can be found in the register function description. See the following figure for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.



The first stage outputs an intermediate clock (DIV) that can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler and outputs the actual internal working clock (card_clk). This clock is the driving clock for all the sub modules of the SD protocol unit, and helps in syncing FIFOs (see [Figure 234](#)) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage can be DIV, DIV/2, DIV/4, ..., or DIV/256. Therefore, the highest frequency of the card_clk is base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of the base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

NOTE

CLK is different for the SDR and DDR modes.

- In the SDR mode, CLK is equal to the internal working clock (card_clk).
- In the DDR mode, CLK is equal to card_clk/2.

50.5.6 SDIO card interrupt

Information on interrupts in 1-bit mode, interrupts in 4-bit mode, and card interrupt handling are detailed in the sections below.

50.5.6.1 Interrupts in 1-bit mode

In this case, the DATA1 pin provides the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

50.5.6.2 Interrupt in 4-bit mode

As the interrupt and data line 1 share pin 8 in a 4-bit mode, an interrupt is only sent by the card and recognized by the host during a specific time. This is known as the interrupt period. The uSDHC module only provides sample the level on pin 8 during the interrupt period. At all other times, the host ignores the level on pin 8 and treats it as the data signal. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in a 4-bit mode, there is only a limited period of time that the interrupt period can be active because of the limited period of data line availability between the multiple blocks of data. This requires stricter definition of the interrupt period. For this case, the interrupt period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line holds low for one clock cycle with the last clock cycle pulling DATA1 high. On completion of the interrupt period, the card releases the DATA1 line into the high Z state. The uSDHC module provides sample of the DATA1 during the interrupt period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10 for further information about the SDIO card interrupt.

50.5.6.3 Card interrupt handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, uSDHC clears the interrupt request to the host system. The host driver should clear this bit before servicing the SDIO interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. To clear this bit, it is required to reset the interrupt source from the external card followed by writing 1 to this bit. In a 1-bit mode, uSDHC detects the SDIO interrupt with or without the SD clock (to support wakeup). In a 4-bit mode, the interrupt signal is sampled during the interrupt period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status is set and the host driver needs to service this interrupt, the SDIO bit in the Interrupt Control Register of SDIO card is cleared. This is required to clear the SDIO interrupt status latched in uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The host driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1 and uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

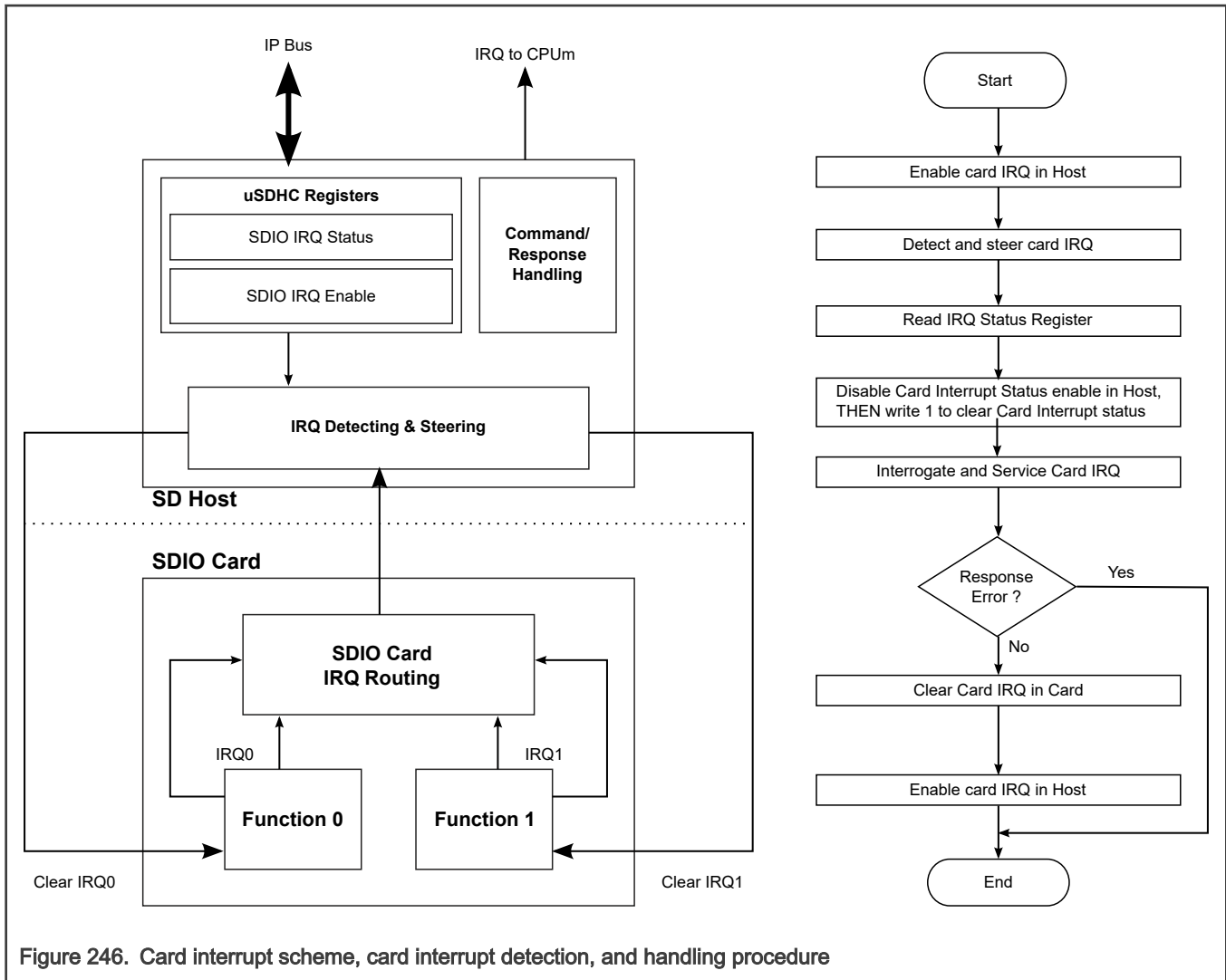


Figure 246. Card interrupt scheme, card interrupt detection, and handling procedure

50.5.7 Card insertion and removal detection

The uSDHC module uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 is pulled to a low voltage level by default. When any card is inserted to or removed from the socket, uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, uSDHC sends an interrupt (if enabled) to inform the Host system that a card is inserted.

50.5.8 Power management and wakeup events

When there is no operation between uSDHC and the card through the SD bus, the user can completely disable the peripheral clock and base clock in the chip-level clock control module to save power. When the user needs to use uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to uSDHC are disabled, for instance, when the system is in low-power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC module can generate these interrupts even when there are no clocks enabled. The three interrupts that can be used as wakeup events are these:

- Card removal interrupt
- Card insertion interrupt
- Interrupt from SDIO card

The uSDHC module offers a power management feature. By clearing the clock enabled bits in the System Control register, the clocks are gated in the low position to uSDHC. For maximum power saving, the user can disable all the clocks to uSDHC when there is no operation in progress.

These three wakeup events (or wakeup interrupts) can also be used to wakeup the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to uSDHC are disabled or when the entire system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to [Protocol Control \(PROT_CTRL\)](#) for more information on the uSDHC Protocol Control register.

50.5.8.1 Setting wakeup events

For uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters the sleep mode. Before the software disables the host clock, it should ensure that all the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

50.5.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature that requires hardware support. There are two types of fast boot modes in the eMMC4.3 specification: boot operation and alternative boot operation in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

In the boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFF (optional for slave), before issuing CMD1.

NOTE

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

50.5.9.1 Boot operation

If the CMD line is held low for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after the CMD line goes low, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line low to read all of the boot data.

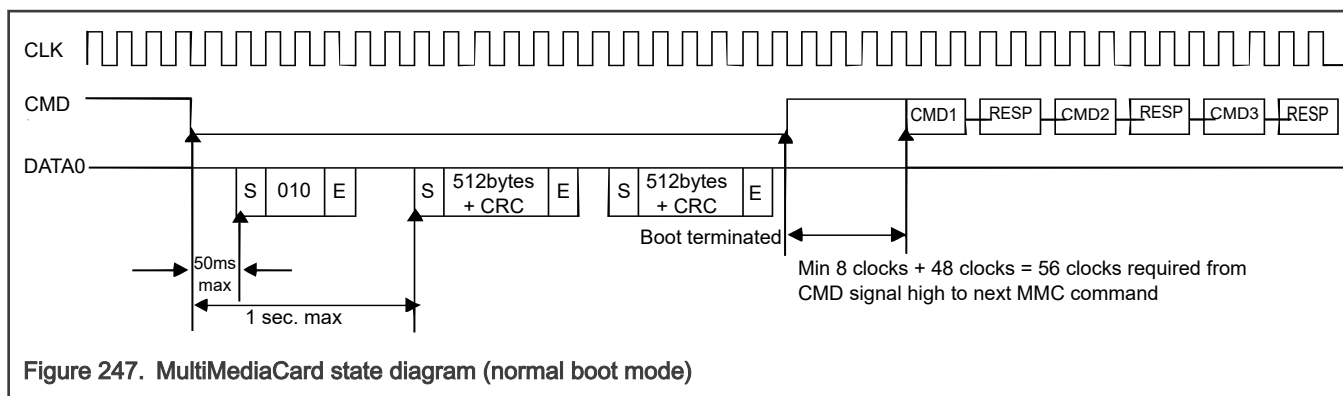
NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes low. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode with the CMD line high.

The boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



50.5.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

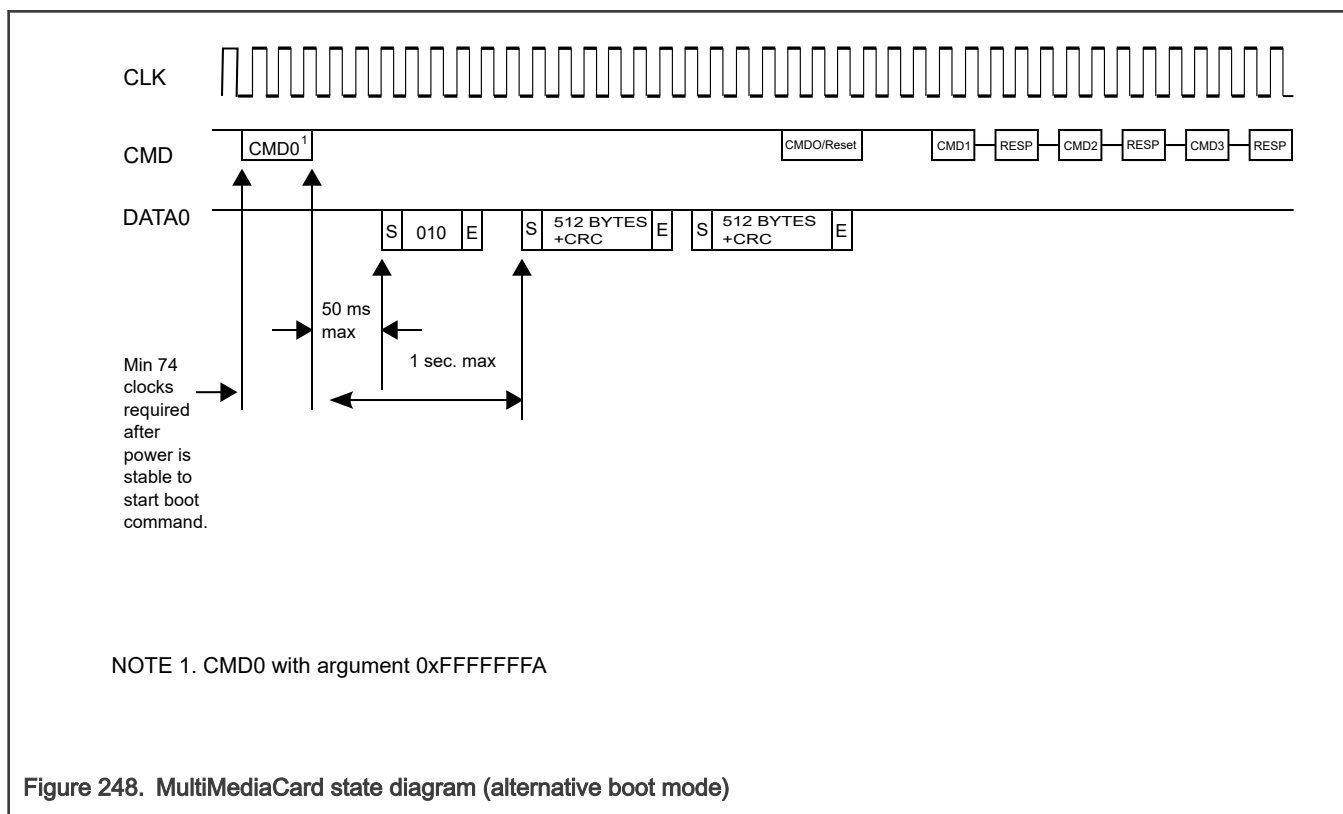
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFF after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after CMD0 with the argument of 0xFFFFFFFF is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFF is received. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode by issuing CMD0 (Reset).

Boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



50.6 Initialization/application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, and ALL_SEND_CID. In the Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to [Commands for MMC/SD/SDIO](#) for the commands of bc and bcr categories.

After the broadcast command CMD3 is issued, the cards enter the standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads turns to the push-pull mode to have the driving capability for maximum frequency operation. Refer to [Commands for MMC/SD/SDIO](#) for the commands of ac and adtc categories.

50.6.1 Command send & response receive basic operation

Assuming that the data type WORD is an unsigned 32-bit integer, the flow indicated below presents a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
    WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
    recommended to implement in a bit-field manner
    wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
    set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
    if (internal DMA is used) wCmd |= 0x1;
    if (multi-block transfer) {
        set MSBSEL bit;
        if (finite block number) {
            set BCEN bit;
            if (auto12 command is to use) set AC12EN bit;
        }
    }
    write_reg(CMDARG, <cmd_arg>); // configure the command argument
    write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
    while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
    read IRQ Status register and check if any error bits about Command are set
    if (any error bits are set) report error;
    write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure that the corresponding interrupt status bits are enabled.

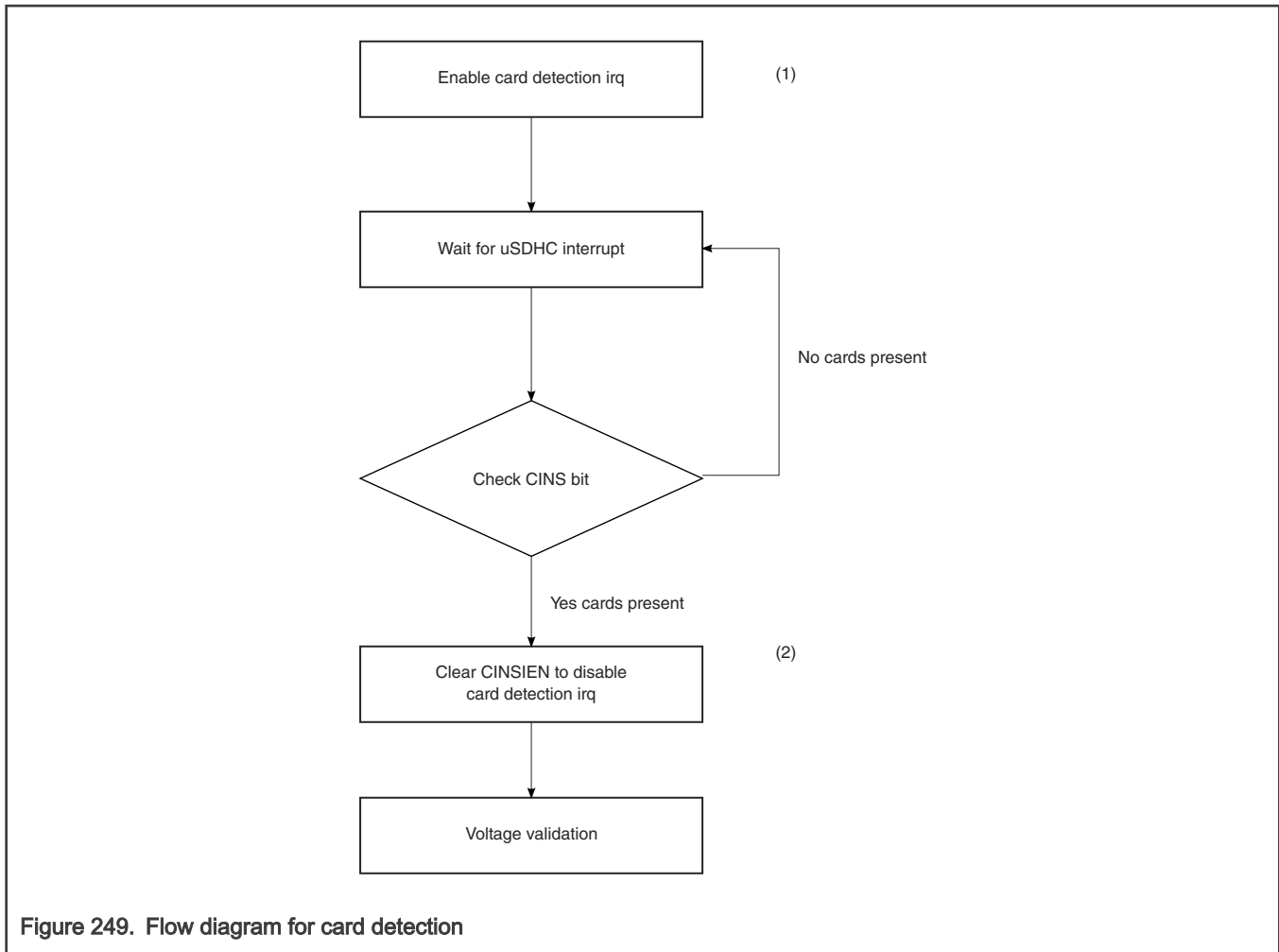
For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and move to the Standby state no response to the Host when CMD2 is sent. The host driver deals with "fake" errors like this with caution.

50.6.2 Card identification mode

When a card is inserted to the socket or the card is reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for MMC cards.

50.6.2.1 Card detect

See the figure below for a flow diagram showing the detection of MMC, SD, and SDIO cards using uSDHC.



Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt.
- When an interrupt from uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion.
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards.

50.6.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) that is driven by Power On Reset (POR).
- Software reset (Host only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the host controller, respectively.
- Card reset (Card only): The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards and SD memory cards. This command sets each card into the idle state regardless of the current card state. For an SD I/O card, CMD52 is used to write an I/O reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both uSDHC and the card.

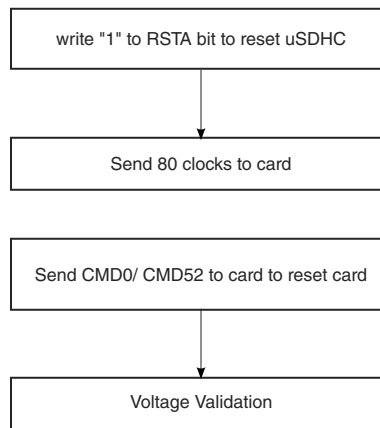


Figure 250. Flow chart for resetting uSDHC and SD I/O card

```

software_reset()
{
  set_bit(SYSCTRL, RSTA); // software reset the Host
  set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
  configure IO pad to set the power voltage of external card to around 3.0V
  poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
  set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
  send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
  or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
  
```

50.6.2.3 Voltage validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for VDD are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer VDD conditions. This means that if the host and card have non-common VDD ranges, the card is neither able to complete the identification cycle nor able to send CSD data.

Therefore, special commands Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory), and IO_Send_Op_Cont (CMD5 for SD I/O) are used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards that do not match the VDD range(s) desired by the host. This is accomplished when the host sends the desired VDD voltage window as the operand of this command. Cards that cannot perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive state. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive state. This query should be used if the host is able to select a common voltage range or if a notification is sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```

voltage_validation(voltage_range_argument)
{
  label the card as UNKNOWN;
  send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
  operation voltage, command argument is zero
  if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
    if (0 < number of IO functions) {
      label the card as SDIO;
    }
  }
}
  
```

```

        IORDY = 0;
        while (!(IORDY in IO OCR response)) { // set voltage range for each IO function
            send_command(IO_SEND_OP_COND, <voltage range>, <other parameter>);
            wait_for_response(IO_SEND_OP_COND);
        } // end of while ...
    } // end of if (0 < ...
    if (memory part is present inside SDIO card) Label the card as SDCombo; // this is an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
    send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage range
for memory part or SD card
    wait_for_response(SD_APP_OP_COND); // voltage range is set
    if (card type is UNKNOWN) label the card as SD;
    return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
    deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card if (card is already labelled as SDCombo) { //
change label
    re-label the card as SDIO;
    ignore the error or report it;
    return; // card is identified as SDIO card
} // of if (card is ...
send_command(SEND_OP_COND, <voltage range>, <...>);
if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted, either
    label the card as UNKNOWN;
    return;
} // of if (RESP_TIMEOUT ...
} // of else
}

```

50.6.2.4 Card registry

Card registry for the MMC and SD/SDIO/SD combo cards are different. For the SD card, the identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the card spec). Currently, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host requests the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command is sent to all the new cards in the system. Incompatible cards are put into the Inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready state), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification state.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA is used to address the card for future data transfer operations. After the RCA is received, the card changes its state to the Standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in the system.

For MMC operation, the host starts the card identification process in the open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line to allow parallel card

operation during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive state. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in the Ready state) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. As the CID is unique for each card, only one card can be successfully sent its full CID to the host. This card then goes into the Identification state. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign a relative card address (RCA) to the card. After the RCA is received, the state of the card changes to standby, and the card does not react in further identification cycles. Also, its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

The following steps show how to perform an operation using MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
    if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
        send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to publish its
RCA
        retrieve RCA from response;
    } // end if (card is labelled as SDCombo ...
    else if (card is labelled as SD) { // for SD card
        send_command(ALL_SEND_CID, <...>);
        if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
        send_command(SET_RELATIVE_ADDR, <...>);
        retrieve RCA from response;
    } // else if (card is labelled as SD ...
    else if (card is labelled as MMC) {
        send_command(ALL_SEND_CID, <...>);
        rca = 0x1; // arbitrarily set RCA, 1 here for example
        send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper 16
bits
    } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

50.6.3 Card access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

50.6.3.1 Block write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

50.6.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card indicates that the failure on the DATA line. The transferred data is discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write-protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card responds with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby state and release the DATA line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling data to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
 - For SD/MMC cards, use SET_BLOCKLEN (CMD16).
 - For SDIO cards or the I/O portion of SDCCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

50.6.3.1.2 DDR write

uSDHC supports the dual data rate mode.

The software flow to write to a card in the DDR mode is described as below:

1. Check the card status and wait until the card is ready for data.

NOTE

For eMMC4.4 card, block length only can be set to 512byte.

2. Set the uSDHC number block register (NOB), where nob is 5, for instance.
3. Set the eMMC4.4 card to high-speed mode and use SWITCH(CMD6).
4. Set the eMMC4.4 card bus with 4-bit/8-bit DDR mode and use SWITCH(CMD6).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred or another error that occurred during the auto12 command sending and response receiving.

50.6.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set the Stop At Block Gap Request (SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Similar to the flow described in [Normal Write](#), the write with pause is shown with the same kind of write operation:

1. Check the card status and wait until the card is ready for data.
2. Set the card block length/size:
 - For SD/MMC, use SET_BLOCKLEN (CMD16).
 - For SDIO cards or the I/O portion of SDCCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred or some another error that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The host driver reads the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible that the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is the end of the transfer. These types of requests are ignored, the driver should treat these as a non-pause transfer, and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the host system is not stopped, and the transfer is active until the data buffer is full. Because of this, it is recommended to avoid using the Suspend command for the SDIO card. This is because when such a command is sent, uSDHC interprets the system that switches to another function on the SDIO card and flush the data buffer. uSDHC takes the Resume command as a normal command with data transfer, and it is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC module automatically sends a CMD12 to mark the end of the multi-block transfer.

50.6.3.2 Block read

Information about Normal read, DDR read, Read with Pause, and Delay Line (DLL) in Read Path are detailed in the sections below.

50.6.3.2.1 Normal read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18,

CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer state. For multi blocks read, data blocks are continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status and wait until card is ready for data.
2. Set the card block length/size:
 - For SD/MMC, use SET_BLOCKLEN (CMD16).
 - For SDIO cards or the I/O portion of SDCCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer read ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

50.6.3.2.2 DDR read

The uSDHC module supports dual data rate mode.

The software flow to write to a card in the DDR mode is described below:

1. Check the card status and wait until the card is ready for data.
2. For eMMC4.4 card, block length can be set to only 512 bytes.
3. Set the uSDHC number block register (NOB) where nob is 5, for instance.
4. Set the eMMC4.4 card to high-speed mode and use SWITCH (CMD6).
5. Set the eMMC4.4 card bus with 4-bit /8-bit DDR mode and use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

50.6.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the host driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, ensure that the RWCTL bit in the Protocol Control register is set, otherwise uSDHC does not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit after the Read Wait capability of the SDIO card is recognized.

Similar to the flow described in [Normal read](#), the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm that the card supports the Read Wait mode.
2. Set the RWCTL bit.

3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
 - For SD/MMC, use SET_BLOCKLEN (CMD16).
 - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
5. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
6. Set the uSDHC number block register (NOB), where nob is 5, for instance.
7. Disable the buffer read ready interrupt, configure the DMA setting, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

Similar to the Write operation, it is possible to meet the ending block of the transfer when paused. In this case, uSDHC ignores the Stop At Block Gap Request and treats it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause is transferred to the host system. No matter if the Suspend command is sent or not, the internal data buffer is not flushed.

If the Suspend command is sent and the transfer is later resumed by means of a Resume command, uSDHC takes the command as a normal one accompanied with data transfer. It is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC automatically sends CMD12 to mark the end of multi-block transfer.

50.6.3.2.4 Delay Line (DLL) in Read Path

The DLL is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage, and temperature (PVT).

The reasons why DLL is needed for uSDHC are these:

- The path of read data traveling from card to host varies.
- In the SD/MMC DDR mode, the minimum input setup and hold time are both at 2.5 ns.

The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in the override mode. The override value set is the number of delay cells. In the override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL automatically adjusts the number of delay cells according to the target value set by the user and PVT changes. Be aware that the target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns; setting target value of 16 means 5ns = (16/32)*10ns. The software can disable automatic update by the setting dll_gate_update bit.

As the user may change the frequency of card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. The following is the correct flow, which should be ignored if DLL_CTRL_ENABLE is not set.

Step 1: Set the DLL_CTRL_RESET bit

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: Clear the DLL_CTRL_RESET bit

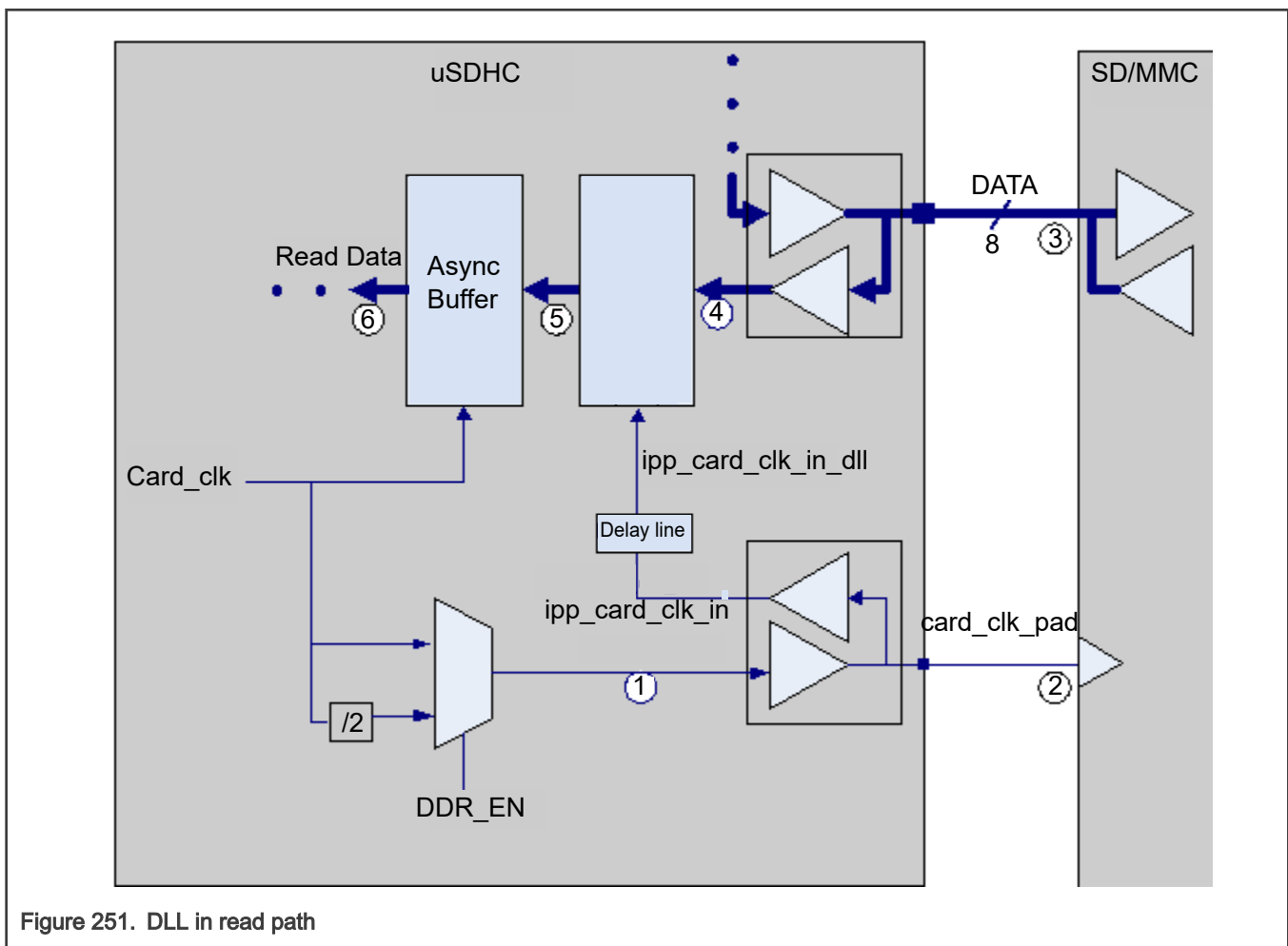
Step 5: Wait until both the DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: Set the DLL_CTRL_SLV_FORCE_UPD

Step 7: Clear the DLL_CTRL_SLV_FORCE_UPD

NOTE

The software should make sure that the `DLL_CTRL_SLV_FORCE_UPD` lasts for at least one `card_clk`. So, the software may need to add some delay between step 6 and step 7.



50.6.3.3 Suspend Resume

The uSDHC module supports the Suspend Resume operations of SDIO cards, although slightly different than the suggested implementation of Suspend in the SDIO card specification.

50.6.3.3.1 Suspend

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC module does not monitor the content of the response, so it does not know if the Suspend command succeeded or not.

Accordingly, it does not de-assert Read Wait for read pause. To solve this problem, the host driver does not mark the Suspend command as "Suspend", (that is, setting the CMDTYP bits to 01). Instead, the driver sends this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform uSDHC that the current transfer is suspended. Here is the sequence for the Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

50.6.3.3.2 Resume

To resume the data transfer, a Resume command is issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation.
2. Send the Resume command: In the Transfer Type register, all the bit fields are set to the value as if this were another ordinary data transfer instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer is resumed.

50.6.3.4 ADMA usage

To use the ADMA in a data transfer, the host driver must prepare the correct descriptor chain prior to sending the read/write command. The steps to prepare the correct descriptor chain are these:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4KB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

50.6.3.5 Transfer error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 errors are detailed in the sections below.

50.6.3.5.1 CRC error

It is possible at the end of a block transfer that a write CRC status error or read CRC error occurs. For this type of error, the latest block received is discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the host driver issues a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 is sent by uSDHC. In this case, the host driver re-sends or re-obtains the last block with a single block transfer.

50.6.3.5.2 Internal DMA error

During the data transfer with internal Simple DMA, if the DMA engine encounters an error on the AHB bus, the DMA operation is aborted, and the DMA error interrupt is sent to the host system. When acknowledged by such an interrupt, the host driver calculates the start address of data block in which the error occurs. The start address can be calculated by either:

- Reading the DMA System Address register: The error occurs during the previous burst. Considering the block size, the previous burst length and the start address of the next burst transfer, it is straight forward to obtain the start address of the corrupted block.
- Reading the BLKCNT field of the Block Attribute register: Considering the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register do not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover from the error.

50.6.3.5.3 Transfer ADMA error

There are three kinds of possible ADMA errors: The AHB transfer, invalid descriptor, and data-length mismatch. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the host driver should recover the error as shown below and re-transfer from the place of interruption.

- AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or to the next block if no block is corrupted.
- Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and recreate the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
- Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

50.6.3.5.4 Auto CMD12 error

After the last block of the multi-block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, uSDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, it is recommended to the host driver to deal with the situations in the following manner:

- Auto CMD12 response time-out: It is not certain whether the command is accepted by the card or not. The host driver clears the auto CMD12 error status bits and re-send CMD12 until it is accepted by the card.
- Auto CMD12 response CRC error: As the card responds to CMD12, it aborts the transfer. The host driver may ignore the error and clear the error status bit.
- Auto CMD12 conflict error or not sent: The command is not sent; therefore, the host driver sends a CMD12 manually.

50.6.3.6 Card interrupt

The external cards can inform the host controller by means of some special signals. For the SDIO card, it can be the low-level on the DATA1 line during some special period. The uSDHC module only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by uSDHC, and the host system is informed by uSDHC asserting the uSDHC interrupt line, the interrupt service from the host driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written. Refer to [Card interrupt handling](#) for the card interrupt handling flow.

50.6.4 Switch function

A switch command is issued by the host driver to enable new features added to the SD/MMC spec. The SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed modes are also defined. To enable these features, a switch command is issued by the host driver.

For SDIO cards, the high-speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high-speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards, the high-speed mode/HS200/HS400 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit, DDR4-bit, and DDR8-bit width of MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards, it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

50.6.4.1 Query, enable, and disable SDIO high-speed mode

```
enable_sdio_high_speed_mode(void)
{
    send CMD52 to query bit SHS at address 0x13;
    if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
    send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
    change clock divisor value or configure the system clock feeding into uSDHC to generate the
    card_clk of around 50MHz;
    (data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
    send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is cleared;
    change clock divisor value or configure the system clock feeding into uSDHC to generate the
    card_clk of the desired value below 25MHz;
    (data transactions like normal peers)
}
```

50.6.4.2 Query, enable, and disable SD high-speed mode/DDR50/SDR50/SDR104

```
enable_sd_speed_mode(void)
{
    set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
    send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response; (high speed
    mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
    wait data transfer done bit is set;
    check if the bit x of received 512 bits is set;
```

```

if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFx and read 64 bytes of data accompanying the R1 response; (high
speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the card_clk
of around 50MHz for high speed mode, 100MHz for SDR50, 200MHz for SDR104, 50MHz for DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}

```

50.6.4.3 Query, enable, and disable MMC high-speed mode

```

enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}

```

50.6.4.4 Set MMC bus width

```
change_mmc_bus_width(void)
{
    send CMD9 to get CSD value of MMC;
    check if the value of SPEC_VER field is 4 or above;
    if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and return;
    send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5; 8-bit,
    x=2; 4-bit, x=1; 1-bit, x=0)
    send CMD13 to wait card ready (busy line released);
    (data transactions like normal peers)
}
```

50.6.5 ADMA operation

Here are the codes for the ADMA1 and ADMA2 operations.

50.6.5.1 ADMA1 operation

```
Set_adma1_descriptor
{
    if (to start data transfer) {
        // Make sure the address is 4KB align.
        Set 'Set' type descriptor;
        {
            Set Act bits to 01;
            Set [31:12] bits data length (byte unit);
        }
        Set 'Tran' type descriptor;
        {
            Set Act bits to 10;
            Set [31:12] bits address (4KB align);
        }
    }
    else if (to fetch descriptor at non-continuous address) {
        Set Act bits to 11;
        Set [31:12] bits the next descriptor address (4KB aligned);
    }
    else { // other types of descriptor
        Set Act bits accordingly
    }
    if (this descriptor is the last one) {
        Set End bit to 1;
    }
    if (to generate interrupt for this descriptor) {
        Set Int bit to 1;
    }
    Set Valid bit to 1;
}
```

50.6.5.2 ADMA2 operation

```
Set_adma2_descriptor
{
    if (to start data transfer) {
        // Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
        Set higher 32-bit of descriptor for this data transfer initial address;
        Set [31:16] bits data length (byte unit);
    }
}
```



```

Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}

```

50.6.6 Fast boot operation

50.6.6.1 Normal fast boot flow

Here are the steps of normal fast boot flow:

1. Software must configure the init_active bit (system control register bit 27) to make sure that 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode. If the data is sent through the DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set Data Transfer Width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN retain the default value, where DPSEL bit is set to 1, DTDSEL is set to 1 and MSBSEL is set to 1.
7. DMAEN should be configured as 0 in the polling mode and if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When step 6 is configured, the boot process begins. The software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt is triggered, and the software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This render CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin the normal initialization process.
10. Reset the host and then can begin the normal process.

50.6.6.2 Alternative fast boot flow

Here are the steps of alternative fast boot flow:

1. Software needs to configure `init_active` bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through the DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in the DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the data transfer width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with the 0xFFFFFFFFFA argument. In alternative boot, `CMDINX`, `CMDTYP`, `RSPTYP`, `CICEN`, `CCEN`, `AC12EN`, `BCEN`, and `DMAEN` retain the default value. `DPSEL` bit is set to 1, `DTDSEL` is set to 1, and `MSBSEL` is set to 1. Note `DMAEN` should be configured as 0 in the polling mode, and if `BCEN` is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in the DDR fast boot mode, `DDR_EN` needs to be set to 1.
7. When step 6 is configured, the boot process begins. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host sends out the interrupt and the software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process.
8. If there is no time out, the software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

50.6.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use Advanced DMA2 (ADMA2) with two destinations.

The detailed flow is described below:

1. The software needs to configure `INIT_ACTIVE` bit (system control register bit 27) to make sure that 74 card clocks are finished.
2. The software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (`BLK_CNT` - `VALUE1`). Here `VALUE1` is the value of the block count that needs to transfer the first time, so that the host stops at the block gap when the uSDHC controller gets `VALUE1` blocks from the device. Also, configure bits[3-0] to select the ack timeout value according to the SD clock frequency.
3. The software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (`BLK_CNT`) to the max value (16'hfff).
4. The software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enable ADMA2 by configuring Protocol Control Register bits [9-8].
6. The software needs to set at least three pairs of ADMA2 descriptor in boot memory (that is, in IRAM, at least six words). The first pair descriptor defines the start address (that is, IRAM) and data length (that is, 512byte*`VALUE1`) of the first part boot code. The software also needs to set the second pair descriptor, the second start address (any value that

- is writable), and data length is suggested to set 1~2word (record as VALUE2). Note that the second couple desc also transfers useful data even at least 1 word, because our ADMA2 cannot support 0 data_length data transfer descriptor.
7. The software needs to configure Command Argument Register to set argument to 0xFFFFFFFF in alternative fast boot and do not need to be set in normal fast boot.
 8. The software needs to configure Transfer Type Register to start the boot process. CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN retain the default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. DMAEN is configured as 1 in the DMA mode. And, if BCEN is configured as 1, then configure blk no in Block Attributes Register to the max value. And, if in the DDR fast boot mode, DDR_EN needs to be set to 1.
 9. When step 8 is configured, boot process begins, the first VALUE1 block number data gets transferred. The software needs to poll the TC bit (bit1 in Interrupt Status Register) to determine first transfer is ended. Also, the software needs to poll the BGE bit (bit2 in Interrupt Status Register) to determine if the first transfer stops at the block gap.
 10. When TC and BGE bits are set to 1, the software can analyze the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of the boot code (VALUE3, the remain boot code block). Remember to set the last descriptor with END.
 11. The software needs to configure the MMC Boot Register (offset 0xc4) again. Set bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot), to 1 (alternative boot); and bit 4 to select the ack mode or not. In the DMA mode, configure bit 7 to 1 for enabling the automatically stop at block gap feature. Also, configure bit31-bit16 to set the (BLK_CNT - (VALUE1+1+VALUE3)), that host stops at block gap when the uSDHC controller gets (VALUE1+1+VALUE3) blocks from device totally include the blocks received in step 9. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequency. Note that the software does not need to configure the BLK_CNT again, because it is counted down automatically by the uSDHC controller.
 12. The software needs to clear the TC and BGE bits and the software needs to clear SABGREQ (bit 16 in the Protocol control register) and set CREQ (bit17 in the Protocol control register) to 1 to resume the data transfer. Host transfers the VALUE2 and VALUE3 data to the destination that is set by descriptor.
 13. The software needs to do poll BGE bit to determine if the fast boot is over.

Note:

1. When ADMA boot flow starts, for uSDHC, it is like a normal ADMA read operation. So, set ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2-word data in second descriptor setting, it is a useful data, so the software needs to deal the data because of the application case.

50.7 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards is provided here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. Broadcast commands (bc), no response
2. Broadcast commands with response (bcr), response from all cards simultaneously
3. Addressed (point-to-point) commands (ac), no data transfer on the DATA
4. Addressed (point-to-point) data transfer commands (adtc)

Response: A response is a token that is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

Table 500. Commands for MMC/SD/SDIO cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3 ¹	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send them operation conditions register contents in the response on the CMD line.
CMD6 ²	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6 ³	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between

Table continues on the next page...

Table 500. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
		[15:0] stuff bits			the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselected all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	adtc	[31:0] reserved bits(all 0)	R1	SEND_TUNING_BLOCK	64 bytes tuning pattern is sent for SDR50 and SDR104.

Table continues on the next page...

Table 500. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.
CMD21	adtc	[31:0] stuff bits	R1	SEND_TUNING_BLOCK	128 clocks of tuning pattern (64 byte in 4 bit mode or 128 byte in 8 bit mode) is sent for HS200 optimal sampling point detection.
CMD22-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command is issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				

Table continues on the next page...

Table 500. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				

Table continues on the next page...

Table 500. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 is used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6 ⁴	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13 ⁴	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.

Table continues on the next page...

Table 500. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
ACMD22 ⁴	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23 ⁴	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).
ACMD41 ⁴	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42 ⁴	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/ DATA3 of the card.
ACMD51 ⁴	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high-speed SD cards. Command SWITCH_FUNC is for high-speed SD cards.
3. Command SWITCH is for high speed MMC cards. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in [Table 501](#).
4. ACMDs is preceded with the APP_CMD command. Commands listed are used for SD only, other SD commands not listed are not supported on this module.

The access bits for the EXT_CSD access modes are shown below.

Table 501. EXT_CSD access modes

Bits	Access name	Operation
00	Command set	The command set is changed according to the Cmd Set field of the argument.
01	Set bits	The bits in the pointed byte are set, according to the bits set to 1 in the Value field.
10	Clear bits	The bits in the pointed byte are cleared, according to the bits set to 1 in the Value field.
11	Write byte	The Value field is written into the pointed byte.

50.8 Software restrictions

50.8.1 Initialization active

The driver cannot set INITA bit in System Control register when any of the command line or data lines are active, so the driver must ensure both CDIHB and CIHB bits are cleared.

50.8.2 Software polling procedure

For polling read or write, after the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in the Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block. For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

50.8.3 Suspend operation

To suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such a 'suspend' command, uSDHC treats the current transfer is aborted and change the BLKCNT register to its original value, instead of retaining the remaining number of blocks.

50.8.4 Data length setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

50.8.5 (A)DMA address setting

To configure the ADMA1/ADMA2/DMA address register, when TC bit is set, the register always updates itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring the ADMA1/ADMA2/DMA address register.

50.8.6 Data port access

Data port does not support parallel access. For example, during an internal DMA access, it is not allowed to write any data to the data port by CPU; or during a CPU read operation, it is also prohibited to write any data to the data port, by either CPU or internal DMA. Otherwise the data is corrupted inside the uSDHC buffer.

50.8.7 Change clock frequency

The uSDHC module does not automatically gate off the card clock when the host driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing the clock divisor value (SDCLKFS or DVS in System Control Register) or setting the RSTA bit.

Also, before changing the clock divisor value, the host driver should make sure that the SDSTB bit is high.

50.8.8 Multi-block read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode. In this case, the card may not respond to

this extra abort command and uSDHC gets response timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

50.9 uSDHC memory map/register definition

50.9.1 uSDHC register descriptions

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map of uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

50.9.1.1 uSDHC memory map

USDHC0 base address: 4013_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA System Address (DS_ADDR)	32	RW	0000_0000h
4h	Block Attributes (BLK_ATT)	32	RW	0001_0000h
8h	Command Argument (CMD_ARG)	32	RW	0000_0000h
Ch	Command Transfer Type (CMD_XFR_TYP)	32	RW	0000_0000h
10h	Command Response0 (CMD_RSP0)	32	R	0000_0000h
14h	Command Response1 (CMD_RSP1)	32	R	0000_0000h
18h	Command Response2 (CMD_RSP2)	32	R	0000_0000h
1Ch	Command Response3 (CMD_RSP3)	32	R	0000_0000h
20h	Data Buffer Access Port (DATA_BUFF_ACC_PORT)	32	RW	0000_0000h
24h	Present State (PRES_STATE)	32	R	See section
28h	Protocol Control (PROT_CTRL)	32	RW	0880_0020h
2Ch	System Control (SYS_CTRL)	32	RW	0080_800Fh
30h	Interrupt Status (INT_STATUS)	32	RW	0000_0000h
34h	Interrupt Status Enable (INT_STATUS_EN)	32	RW	0000_0000h
38h	Interrupt Signal Enable (INT_SIGNAL_EN)	32	RW	0000_0000h
3Ch	Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)	32	RW	0000_0000h
40h	Host Controller Capabilities (HOST_CTRL_CAP)	32	RW	07F3_B407h
44h	Watermark Level (WTMK_LVL)	32	RW	0810_0810h
48h	Mixer Control (MIX_CTRL)	32	RW	8000_0000h
50h	Force Event (FORCE_EVENT)	32	RW	0000_0000h
54h	ADMA Error Status (ADMA_ERR_STATUS)	32	R	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
58h	ADMA System Address (ADMA_SYS_ADDR)	32	RW	0000_0000h
60h	DLL (Delay Line) Control (DLL_CTRL)	32	RW	0000_0000h
64h	DLL Status (DLL_STATUS)	32	R	0000_0200h
68h	CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)	32	RW	0000_0000h
70h	Strobe DLL control (STROBE_DLL_CTRL)	32	RW	0000_0000h
74h	Strobe DLL status (STROBE_DLL_STATUS)	32	R	0000_0200h
C0h	Vendor Specific Register (VEND_SPEC)	32	RW	2000_7809h
C4h	MMC Boot (MMC_BOOT)	32	RW	0000_0000h
C8h	Vendor Specific 2 Register (VEND_SPEC2)	32	RW	0000_1006h
CCh	Tuning Control (TUNING_CTRL)	32	RW	0021_2800h

50.9.1.2 DMA System Address (DS_ADDR)

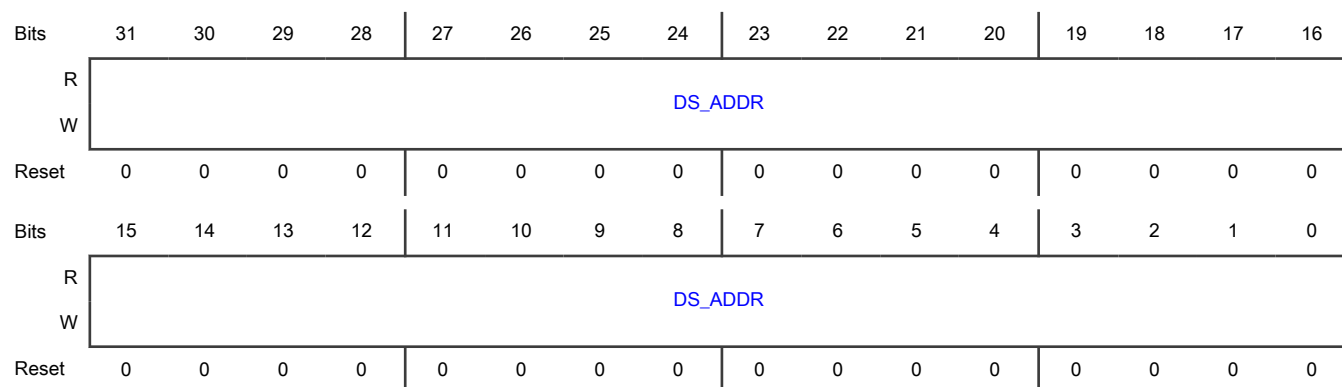
Offset

Register	Offset
DS_ADDR	0h

Function

This register contains the physical system memory address used for DMA transfers.

Diagram



Fields

Field	Function
31-0 DS_ADDR	System address DMA system address / argument 2 When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23. 1. SDMA system address Because the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The host driver waits until the DLA bit in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC automatically changes SEQ burst type to NSEQ. Because this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC bit is set. Such restriction is also listed in Software restrictions. 2. Argument 2 This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

50.9.1.3 Block Attributes (BLK_ATT)

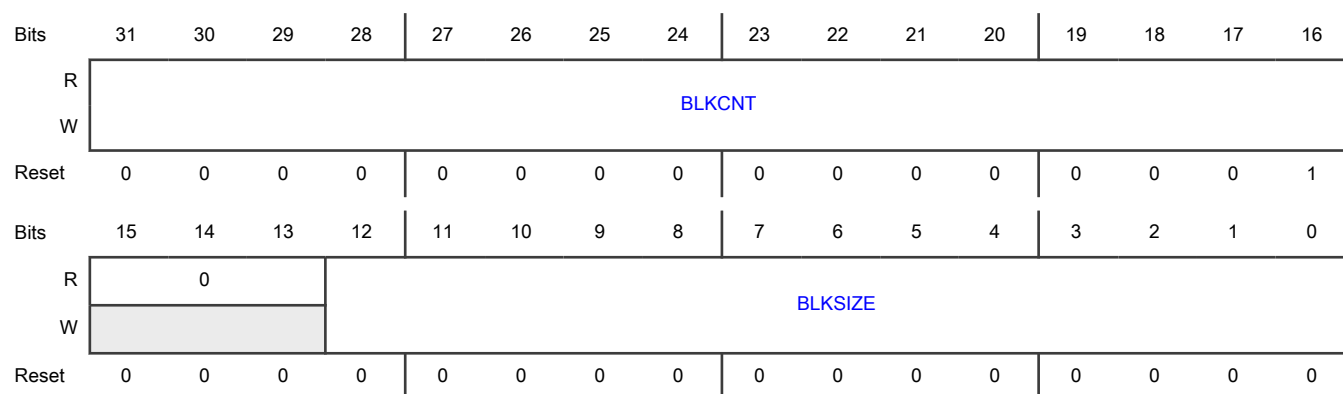
Offset

Register	Offset
BLK_ATT	4h

Function

This register is used to configure the number of data blocks and the number of bytes in each block.

Diagram



Fields

Field	Function
31-16 BLKCNT	Blocks count for current transfer This register is enabled when the Block Count Enable field in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register always reads as 1. The host driver sets this register to a value between 1 and the maximum block count. The uSDHC module decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to zero results in no data blocks being transferred. This register should be accessed only when no transaction is executing (that is, after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When saving transfer content because of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when the Suspend command is sent out, uSDHC treats the current transfer as aborted and change the BLKCNT register back to its original value instead of keeping the dynamical indicator of the remaining block count. When restoring transfer content prior to issuing a Resume command, the host driver restores the previously saved block count. NOTE Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL field is indicating a single block transfer, the read value of BLKCNT is always 1. 0000_0000_0000_0000b - Stop count 0000_0000_0000_0001b - 1 block 0000_0000_0000_0010b - 2 blocks 1111_1111_1111_1111b - 65535 blocks
15-13 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
12-0 BLKSIZE	Transfer block size This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations are ignored. 0_0000_0000_0000b - No data transfer 0_0000_0000_0001b - 1 byte 0_0000_0000_0010b - 2 bytes 0_0000_0000_0011b - 3 bytes 0_0000_0000_0100b - 4 bytes 0_0001_1111_1111b - 511 bytes 0_0010_0000_0000b - 512 bytes 0_1000_0000_0000b - 2048 bytes 1_0000_0000_0000b - 4096 bytes

50.9.1.4 Command Argument (CMD_ARG)

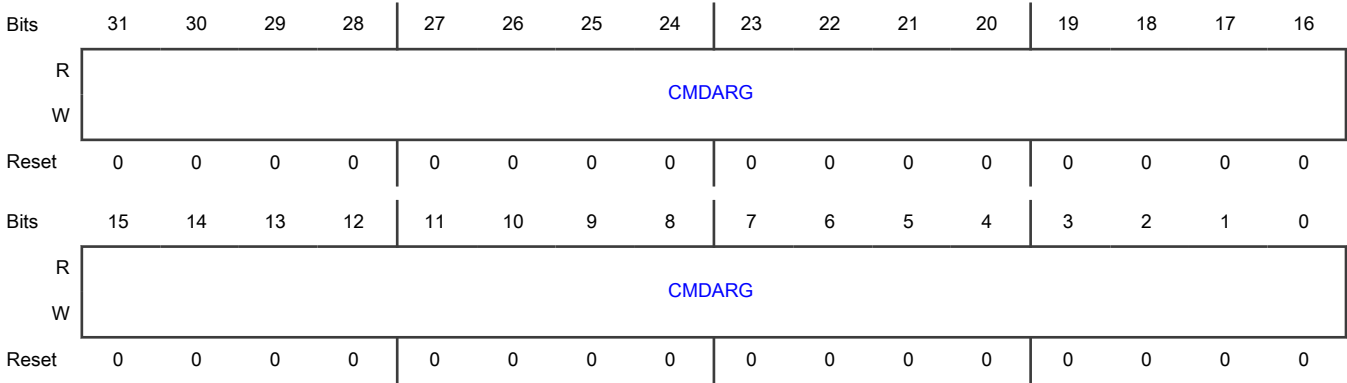
Offset

Register	Offset
CMD_ARG	8h

Function

This register contains the SD/MMC command argument.

Diagram



Fields

Field	Function
31-0 CMDARG	Command argument The SD/MMC command argument is specified as bits 39-8 of the command format in the SD or MMC specification. This register is write protected when the Command Inhibit (CMD) field in the Present State register is set.

50.9.1.5 Command Transfer Type (CMD_XFR_TYP)

Offset

Register	Offset
CMD_XFR_TYP	Ch

Function

This register is used to control the operation of data transfers. The host driver sets this register before issuing a command followed by a data transfer or before issuing a Resume command. To prevent data loss, uSDHC prevents writing to the bits, which are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver checks the Command Inhibit DAT field ([PRES_STATE\[CDIHB\]](#)) and the Command Inhibit CMD field ([PRES_STATE\[CIHB\]](#)) in the Present State register before writing to this register. When the CDIHB field in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB field is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as a single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC ignores the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active ([PRES_STATE\[WPSPL\]](#) field of Present State register is '1'); otherwise, uSDHC also ignores the command.

If the commands with data transfer do not receive the response in 64 clock cycles, that is, if response time-out happens, uSDHC treats the external device, does not accept the command, and aborts the data transfer. In this scenario, the driver should issue the command again to retry the transfer. It is also possible that for some reason the card responds to the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 502. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

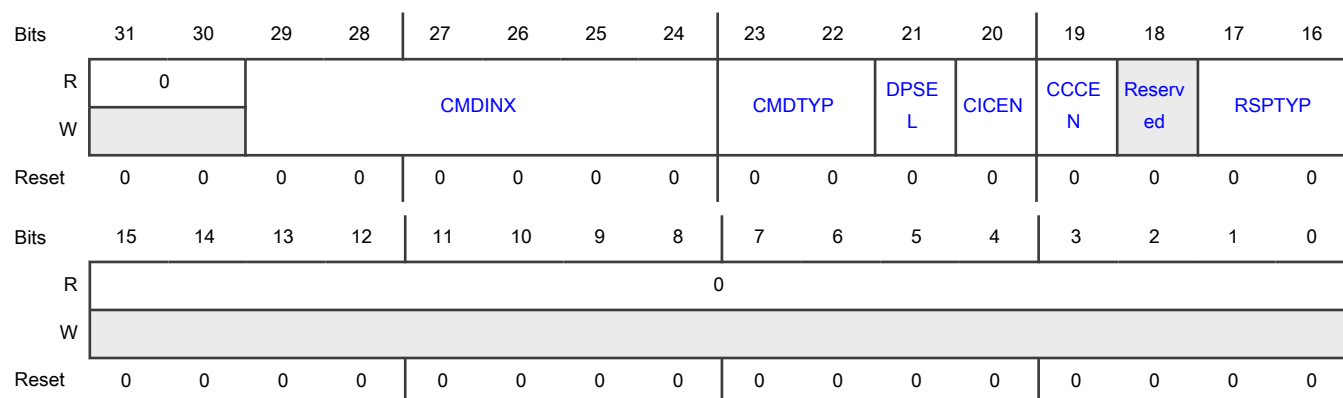
The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, regarding the Response Type bits as well as the name of the response type.

Table 503. Relationship between parameters and the name of the response type

Response type	Index check enable	CRC check enable	Name of response type
00	0	0	No response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification, but R5b is defined in this specification to specify that uSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command is used with R5b.
- The CRC fields for R3 and R4 are expected to be all 1 bits. The CRC check is disabled for these response types.

Diagram



Fields

Field	Function
31-30 —	Reserved
29-24 CMDINX	Command index These bits are set to the command number that is specified in bits 45-40 of the command-format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23-22 CMDTYP	Command type There are three types of special commands: Suspend, Resume, and Abort. These bits are set to 00b for all other commands.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- Suspend command: If the Suspend command succeeds, uSDHC assumes that the card bus has been released and that it is possible to issue the next command that uses the DATA line. Because uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the host driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform uSDHC that a Suspend command was successfully issued. See Suspend Resume for more details. After the end bit of command is sent, uSDHC deasserts Read Wait for read transactions and stops checking busy for write transactions. In a 4-bit mode, the interrupt cycle starts. If the Suspend command fails, uSDHC maintains its current state, and the host driver restarts the transfer by setting the Continue Request field in the Protocol Control register. - Resume command: The host driver re-starts the data transfer by restoring the registers saved before sending the Suspend command and then sends the Resume command. The uSDHC module checks for a pending busy state before starting write transfers. - Abort command: If this command is set when executing a read transfer, uSDHC stops reads to the buffer. If this command is set when executing a write transfer, uSDHC stops driving the DATA line. After issuing the Abort command, the host driver should issue a software reset (Abort Transaction). 00b - Normal other commands 01b - Suspend CMD52 for writing bus suspend in CCCR 10b - Resume CMD52 for writing function select in CCCR 11b - Abort CMD12, CMD52 for writing I/O Abort in CCCR
21 DPSEL	Data present select This field is set to 1 to indicate that data is present and is transferred using the DATA line. It is set to 0 for the following: - Commands using only the CMD line (for example, CMD52) - Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b (for example, CMD38)) NOTE In resume command, this field is set, and other bits in this register is set the same as when the transfer was initially launched. When the write protect switch is on, (that is, the WPSPL field is active as '0'), any command with a write operation ignored. When this field is set, while the DTDSEL field is 0, writes to the register Transfer Type are ignored. 0b - No data present 1b - Data present
20 CICEN	Command index check enable If this field is set to 1, uSDHC checks the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this field is set to 0, the Index field is not checked. 0b - Disable command index check 1b - Enables command index check

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 CCEN	Command CRC check enable If this field is set to 1, uSDHC checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this field is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. See RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP). 0b - Disables command CRC check 1b - Enables command CRC check
18 —	Reserved
17-16 RSPTYP	Response type select 00b - No response 01b - Response length 136 10b - Response length 48 11b - Response length 48, check busy after response
15-0 —	Reserved

50.9.1.6 Command Response0 (CMD_RSP0)

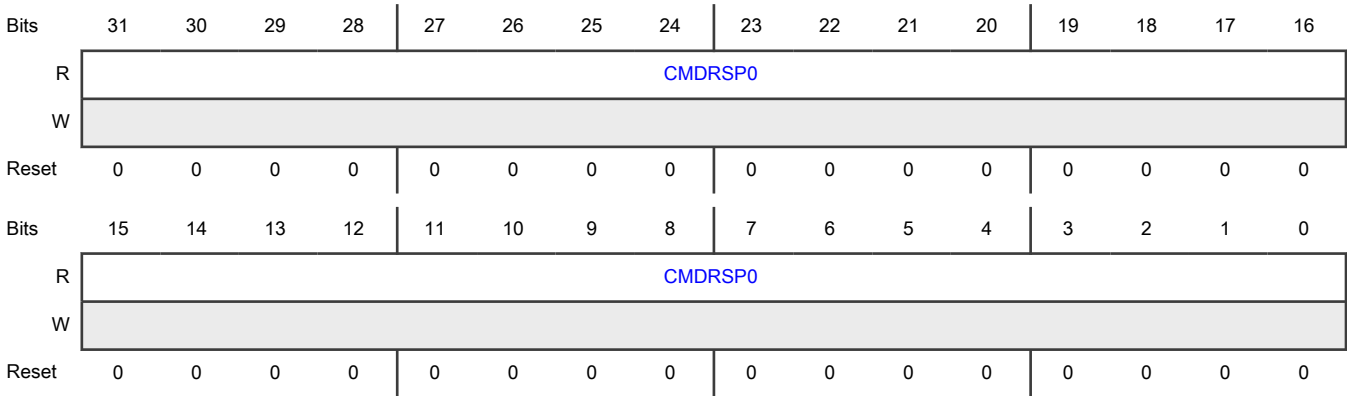
Offset

Register	Offset
CMD_RSP0	10h

Function

This register is used to store part 0 of the response bits from the card.

Diagram



Fields

Field	Function
31-0	Command response 0
CMDRSP0	See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.1.7 Command Response1 (CMD_RSP1)

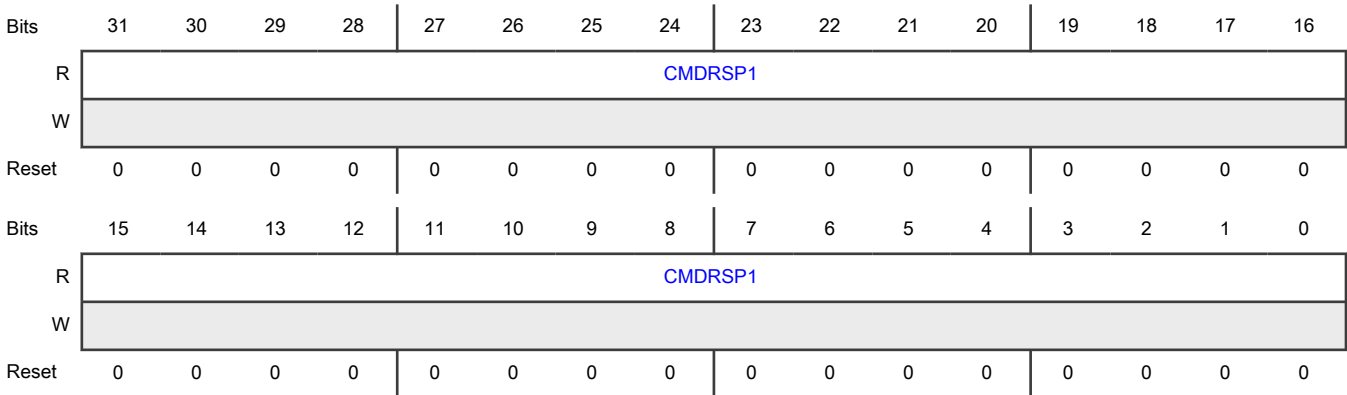
Offset

Register	Offset
CMD_RSP1	14h

Function

This register is used to store part 1 of the response bits from the card.

Diagram



Fields

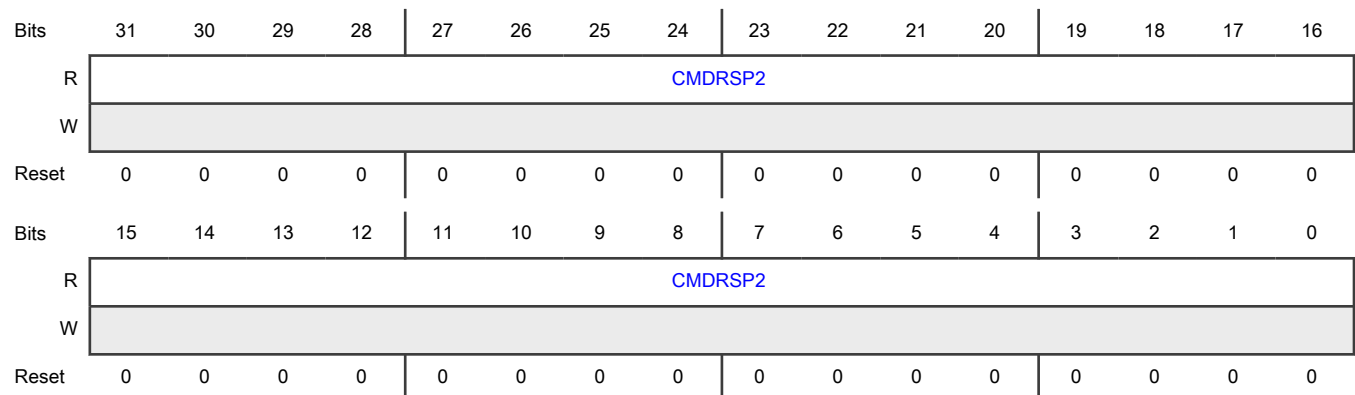
Field	Function
31-0 CMDRSP1	Command response 1 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.1.8 Command Response2 (CMD_RSP2)**Offset**

Register	Offset
CMD_RSP2	18h

Function

This register is used to store part 2 of the response bits from the card.

Diagram**Fields**

Field	Function
31-0 CMDRSP2	Command response 2 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.1.9 Command Response3 (CMD_RSP3)**Offset**

Register	Offset
CMD_RSP3	1Ch

Function

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD bus to Command Response registers for each response type. In this table, R[] refers to a bit range within the response data as transmitted on the SD bus.

Table 504. Response bit definition for each response type

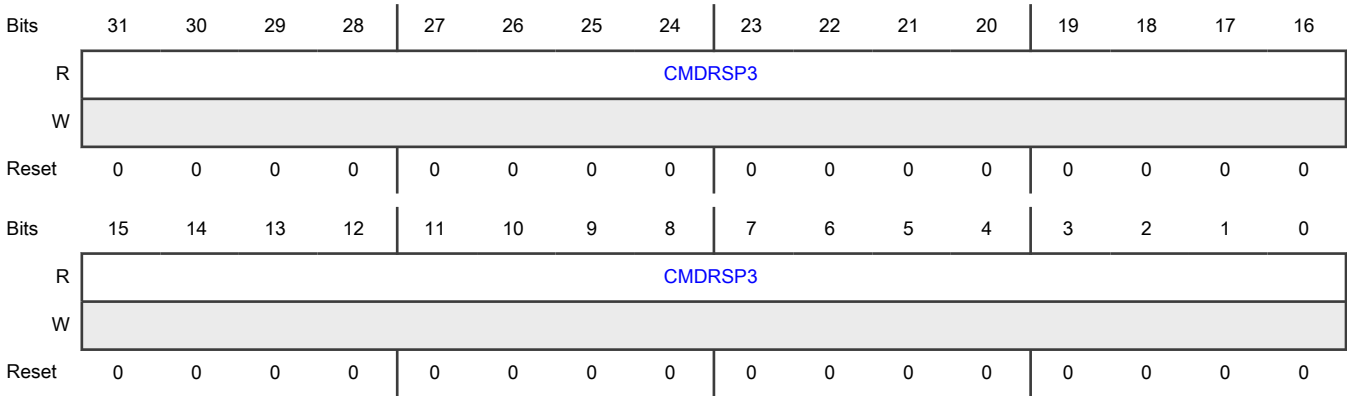
Response type	Meaning of response	Response field	Response register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0
R1b (auto CMD12 response)	Card status for auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (publish RCA)	New published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, uSDHC only stores part of the response data in the Command Response registers. This enables the host driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, uSDHC checks R[47:1], and if the response length is 136 the uSDHC checks R[119:1].

Because uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Diagram



Fields

Field	Function
31-0	Command response 3
CMDRSP3	See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.1.10 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

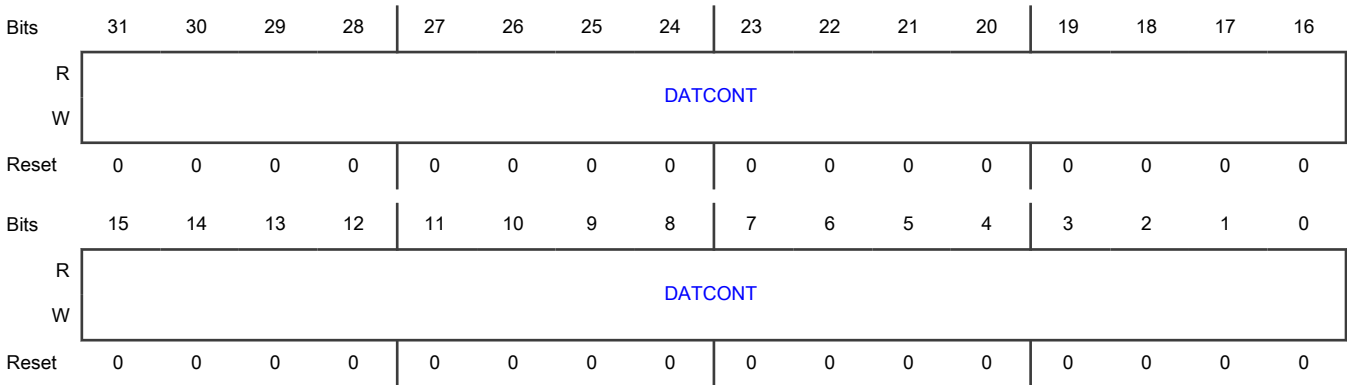
Offset

Register	Offset
DATA_BUFF_ACC_PORT	20h

Function

This is a 32-bit data port register used to access the internal buffer.

Diagram



Fields

Field	Function
31-0 DATCONT	Data content The Buffer Data Port register is for 32-bit data access by the ARM platform. When the internal DMA is enabled, any write to this register is ignored, and any read from this register always yields 0s.

50.9.1.11 Present State (PRES_STATE)

Offset

Register	Offset
PRES_STATE	24h

Function

The host driver can get status of uSDHC from this 32-bit read only register.

The host driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands are issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

NOTE

The reset value of Present State register depends on board connectivity.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DLSL								CLSL	0				WPSP L	CDPL	0	CINST
W																	
Reset	u	u	u	u	u	u	u	u	u	0	0	0	u	u	0	u	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	TSCD	0		RTR	BREN	BWEN	RTA	WTA	SDOF F	PERO FF	HCKO FF	IPGOF F	SDST B	DLA	CDIHB	CIHB	
W																	
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

Fields

Field	Function
31-24 DLSL	DATA[7:0] line signal level

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This status is used to check the DATA line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. 0000_0000b - Data 0 line signal level 0000_0001b - Data 1 line signal level 0000_0010b - Data 2 line signal level 0000_0011b - Data 3 line signal level 0000_0100b - Data 4 line signal level 0000_0101b - Data 5 line signal level 0000_0110b - Data 6 line signal level 0000_0111b - Data 7 line signal level
23 CLSL	CMD line signal level This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this field after reset is 1'b1, when the command line is pulled up.
22-20 —	Reserved
19 WPSPL	Write protect switch pin level The Write Protect switch is supported for memory and combo cards. This field reflects the inverted value of the WP pin of the card socket. A software reset does not affect this field. The reset value is affected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this field is high and write is enabled. 0b - Write protected (WP = 1) 1b - Write enabled (WP = 0)
18 CDPL	Card detect pin level This field reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this field. This field may be valid, but is not guaranteed, because of propagation delay. Use of this field is limited to testing because it must be debounced by software. A software reset does not affect this field. A write to the Force Event Register does not affect this field. The reset value is affected by the external card detection pin. This field shows the value on the CD_B pin (that is, when a card is inserted in the socket, it is 0 on the CD_B input, and consequently, the CDPL reads 1. 0b - No card present (CD_B = 1) 1b - Card present (CD_B = 0)
17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 CINST	Card inserted This field indicates whether a card has been inserted. The uSDHC module debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not affect this field. The Software Reset For All in the System Control register does not affect this field. A software reset does not affect this field. 0b - Power on reset or no card 1b - Card inserted
15 TSCD	Tape select change done This field indicates the delay setting is effective after write CLK_TUNE_CTRL_STATUS register. 0b - Delay cell select change is not finished. 1b - Delay cell select change is finished.
14-13 —	Reserved
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode and EMMC HS200 mode) Host controller may request host driver to execute re-tuning sequence by setting this field when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This field is cleared when a command is issued with setting Execute Tuning field in MIXER_CTRL register. Changing of this field from 0 to 1 generates Re-Tuning Event. See Interrupt status registers for more detail. This field isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock). 0b - Fixed or well tuned sampling clock 1b - Sampling clock needs re-tuning
11 BREN	Buffer read enable This status field is used for non-DMA read transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this field is high, valid data greater than the watermark level exist in the buffer. A change of this field from 1 to 0 occurs when some reads from the buffer (read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this field from 0 to 1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 0b - Read disable 1b - Read enable
10 BWEN	Buffer write enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This status field is used for non-DMA write transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this field is 1, valid data greater than the watermark level can be written to the buffer. A change of this field from 1 to 0 occurs when some writes to the buffer (write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this field from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 0b - Write disable 1b - Write enable
9 RTA	Read transfer active This status field is used for detecting completion of a read transfer. This field is set for either of the following conditions: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer A transfer complete interrupt is generated when this field changes to 0. This field is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent because of the Stop At Block Gap Request being set to 1. 0b - No valid data 1b - Transferring data
8 WTA	Write transfer active This status field indicates a write transfer is active. If this field is 0, it means no valid write data exists in uSDHC. This field is set in either of the following cases: • After the end field of the write command • When writing 1 to the Continue Request field in the Protocol Control register to restart a write transfer This field is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request During a write transaction, a Block Gap Event interrupt is generated when this field is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the host driver in determining when to issue commands during Write Busy state.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No valid data 1b - Transferring data
7 SDOFF	SD clock gated off internally This status field indicates that the SD clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON field is 0 to stop the SD clock in idle status. This field is for the host driver to debug data transaction on the SD bus. 0b - SD clock is active. 1b - SD clock is gated off.
6 PEROFF	IPG_PERCLK gated off internally This status field indicates that the IPG_PERCLK is internally gated off. This field is for the host driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK is gated off, otherwise IPG_PERCLK is always active. 0b - IPG_PERCLK is active. 1b - IPG_PERCLK is gated off.
5 HCKOFF	HCLK gated off internally This status field indicates that the HCLK is internally gated off. This field is for the host driver to debug during a data transfer. 0b - HCLK is active. 1b - HCLK is gated off.
4 IPGOFF	Peripheral clock gated off internally This status field indicates that the peripheral clock is internally gated off. This field is for the host driver to debug. 0b - Peripheral clock is active. 1b - Peripheral clock is gated off.
3 SDSTB	SD clock stable This status field indicates that the internal card clock is stable. This field is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON field in System Control register to remove glitches on the card clock when the frequency is changing. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. 0b - Clock is changing frequency and not stable. 1b - Clock is stable.
2 DLA	Data line active This status field indicates whether one of the DATA lines on the SD bus is in use.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	In the case of read transactions: This status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer This field is cleared in either of the following cases: • When the end field of the last data block is sent from the SD bus to uSDHC. • When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC module waits at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait to use the suspend / resume function. This field remains 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the write command • When writing to 1 to the Continue Request field in the Protocol Control register to continue a write transfer This field is cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, uSDHC also detects if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, uSDHC assumes the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and because of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This field is cleared when the DATA0 line is released. 0b - DATA line inactive 1b - DATA line active
1 CDIHB	Command inhibit (DATA) This status field is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this field is 0, it indicates that uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE The SD host driver can save registers for a suspend transaction after this field has changed from 1 to 0. 0b - Can issue command that uses the DATA line 1b - Cannot issue command that uses the DATA line
0 CIHB	Command inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and uSDHC can issue a SD / MMC command using the CMD line. This field is set also immediately after the Transfer Type register is written. This field is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, commands using only the CMD line can be issued if this field is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If uSDHC cannot issue the command because of a command conflict error (see Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this field remains 1 and the Command Complete is not set. The Status of issuing an auto CMD12 does not show on this field. 0b - Can issue command using only CMD line 1b - Cannot issue command

50.9.1.12 Protocol Control (PROT_CTRL)

Offset

Register	Offset
PROT_CTRL	28h

Function

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether uSDHC issues a Suspend command or the SD card accepts the Suspend command.

- If the host driver does not issue a Suspend command, the Continue request is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card accepts it, a Resume command is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card does not accept it, the Continue request is used to restart the transfer.

Any time stop at block gap request stops the data transfer, the host driver waits for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the host driver clears the Stop At Block Gap Request before or simultaneously.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved	NON_EXA...	BURST_LEN_EN			WECR_M	WECI_NS	WECI_NT	Reserved				RD_D_ONE...	IABG	RWCT_L	CREQ	SABG_REQ
W																	
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0							DMASEL	CDSS	CDTL	EMODE		D3CD	DTW		Reserved	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	

Fields

Field	Function
31 —	Reserved Always write as 0
30 NON_EXACT_BLOCK_READ	Non-exact block read Current block read is non-exact block read. It is only used for SDIO. 0b - The block read is exact block read. Host driver does not need to issue abort command to terminate this multi-block read. 1b - The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read.
29-27 BURST_LEN_ENABLE	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLES on the AXI side. 1xxb - Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP. x1xb - Burst length is enabled for INCR4 / INCR8 / INCR16. xx1b - Burst length is enabled for INCR.
26 WECRM	Wakeup event enable on SD card removal This field enables a wakeup event, via a card removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this field. When this field is set, the Card Removal Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Removal Status and uSDHC interrupt. 0b - Disables wakeup event enable on SD card removal 1b - Enables wakeup event enable on SD card removal
25	Wakeup event enable on SD card insertion

Table continues on the next page...

Table continued from the previous page...

Field	Function
WECINS	This field enables a wakeup event, via a card insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this field. When this field is set, the Card Insertion Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Insertion Status and uSDHC interrupt. 0b - Disable wakeup event enable on SD card insertion 1b - Enable wakeup event enable on SD card insertion
24 WECINT	Wakeup event enable on card interrupt This field enables a wakeup event, via a card interrupt, in the Interrupt Status register. This field can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this field is set, the Card Interrupt Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Interrupt Status and uSDHC interrupt. 0b - Disables wakeup event enable on card interrupt 1b - Enables wakeup event enable on card interrupt
23-21 —	Reserved Always write as 3'b100
20 RD_DONE_NO_8CLK	Read performed number 8 clock According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end field of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks are active after the end field of the last read data transaction. However, these 8 clocks should not be active if user wants to use stop at block gap (include the auto stop at block gap in boot mode) feature for read and the RWCTL field (bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid these 8 clocks. Otherwise, the device might send extra data to uSDHC while uSDHC ignores these data. In a summary, this field should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.
19 IABG	Interrupt at block gap This field is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this field should be set to 0 to avoid an inadvertent interrupt. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. 0b - Disables interrupt at block gap 1b - Enables interrupt at block gap
18 RWCTL	Read wait control The read wait function is optional for SDIO cards. If the card supports read wait, set this field to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise, uSDHC has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. If the card does not support read wait, this field should

Table continues on the next page...

Table continued from the previous page...

Field	Function
	never be set to 1; otherwise, DATA line conflicts might occur. If this field is set to 0, stop at block gap during read operation is also supported, but uSDHC stops the SD clock to pause reading operation. 0b - Disables read wait control and stop SD clock at block gap when SABGREQ field is set 1b - Enables read wait control and assert read wait without stopping SD clock at block gap when SABGREQ field is set
17 CREQ	Continue request This field is used to restart a transaction which was stopped using the stop at block gap request. When a suspend operation is not accepted by the card, it is also by setting this field to restart the paused transfer. To cancel stop at the block gap, set stop at block gap request to 0 and set this field to 1 to restart the transfer. The uSDHC module automatically clears this field, therefore it is not necessary for the host driver to set this field to 0. If both stop at block gap request and this field are 1, the continue request is ignored. 0b - No effect 1b - Restart
16 SABGREQ	Stop at block gap request This field is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion, the host driver leaves this field set to 1. Clearing both the stop at block gap request and continue request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC module supports the stop at block gap request for write transfers, but for read transfers it requires that the SDIO card support read wait. Therefore, the host driver does not set this field during read transfers unless the SDIO card supports Read Wait and has set the read wait control to 1; otherwise, uSDHC stops the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the host driver writes data to the Data Port register, the host driver sets this field after all block data is written. If this field is set to 1, the host driver does not write data to the Data Port register after a block is sent. Once this field is set, the host driver does not clear this field before the Transfer Complete field in Interrupt Status register is set, otherwise uSDHC's behavior is undefined. This field effects read transfer active, write transfer active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 0b - Transfer 1b - Stop
15-10 —	Reserved
9-8 DMASEL	DMA select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00b - No DMA or simple DMA is selected. 01b - ADMA1 is selected. 10b - ADMA2 is selected.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Reserved
7 CDSS	Card detect signal selection This field selects the source for the card detection. 0b - Card detection level is selected (for normal purpose). 1b - Card detection test level is selected (for test purpose).
6 CDTL	Card detect test level This bit is enabled while the card detection signal selection is set to 1 and it indicates card insertion. 0b - Card detect test level is 0, no card inserted 1b - Card detect test level is 1, card inserted
5-4 EMODE	Endian mode The uSDHC module supports all three endian modes in data transfer. See Data buffer for more details. 00b - Big endian mode 01b - Half word big endian mode 10b - Little endian mode 11b - Reserved
3 D3CD	DATA3 as card detection pin If this field is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 might set the card into the SPI mode, which uSDHC does not support. 0b - DATA3 does not monitor card insertion 1b - DATA3 as card detection pin
2-1 DTW	Data transfer width This field selects the data width of the SD bus for a data transfer. The host driver sets it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits. 00b - 1-bit mode 01b - 4-bit mode 10b - 8-bit mode 11b - Reserved
0 —	Reserved

50.9.1.13 System Control (SYS_CTRL)

Offset

Register	Offset
SYS_CTRL	2Ch

Function

This register provides control of the system. See detail in the field description.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			RSTT	INITA	RSTD	RSTC	RSTA	IPP_R ST...	Reserv ed	Reserv ed	0	DTCV			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDCLKFS								DVS				Reserved			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Fields

Field	Function
31-29 —	Reserved
28 RSTT	Reset tuning When set this field to 1, it resets tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning field in AUTOCMD12_ERR_STATUS also sets this field to 1 to reset tuning circuit
27 INITA	Initialization active When this field is set, 80 SD-clocks are sent to the card. After the 80 clocks are sent, this field is self cleared. This field is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this field is already 1 has no effect. Writing 0 to this field at any time has no effect. When either of the CIHB and CDIHB fields in the Present State register are set, writing 1 to this field is ignored (that is, when command line or data lines are active, write to this field is not allowed). On the other-hand, when this field is set, that is, during initialization active period, it is allowed to issue command, and the command bit stream appears on the CMD pad after all 80 clock cycles are done. So, when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs to send 80 cycles to the card and does not want to wait till this field is self cleared.
26	Software reset for data line

Table continues on the next page...

Table continued from the previous page...

Field	Function
RSTD	Only part of the data circuit is reset. DMA circuit is also reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Data Port register • Buffer is cleared and initialized • Present State register • Buffer read enable • Buffer write enable • Read transfer active • Write transfer active • DATA line active • Command Inhibit (DATA) Protocol Control register • Continue request • Stop At Block Gap Request Interrupt Status register • Buffer read ready • Buffer write ready • DMA interrupt • Block gap event • Transfer complete NOTE When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
25 RSTC	Software reset for CMD line Only part of the command circuit is reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Present State Register Command Inhibit (CMD) • Interrupt Status register Command Complete 0b - No reset 1b - Reset
24 RSTA	Software reset for all

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This reset effects the entire host controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the host driver is set this field to 1 to reset uSDHC. The uSDHC module resets this field to 0 when the capabilities registers are valid and the host driver can read them. Additional use of Software Reset For All does not affect the value of the capabilities registers. After this field is set, it is recommended that the host driver reset the external card and re-initialize it. After this field is set, the software should wait for self-clear. In tuning process, after every CMD19 is finished, this field is set to retest uSDHC. NOTE When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
23 IPP_RST_N	Hardware reset This register's value is output to card through pad directly to hardware reset pin of the card if the card supports this feature.
22 —	Reserved
21 —	Reserved
20 —	Reserved
19-16 DTCV	Data timeout counter value This value determines the interval by which DAT line timeouts are detected. See the Data Timeout Error field in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency is generated by dividing the base clock SDCLK value by this value. The host driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 0000b - SDCLK x 2¹⁴ 0001b - SDCLK x 2¹⁵ 0010b - SDCLK x 2¹⁶ 0011b - SDCLK x 2¹⁷ 0100b - SDCLK x 2¹⁸ 0101b - SDCLK x 2¹⁹ 0110b - SDCLK x 2²⁰

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0111b - SDCLK x 2 ²¹ 1000b - SDCLK x 2 ²² 1001b - SDCLK x 2 ²³ 1010b - SDCLK x 2 ²⁴ 1011b - SDCLK x 2 ²⁵ 1100b - SDCLK x 2 ²⁶ 1101b - SDCLK x 2 ²⁷ 1110b - SDCLK x 2 ²⁸ 1111b - SDCLK x 2 ²⁹ + SDCLK x 2 ²⁸ + SDCLK x 2 ²⁷ + SDCLK x 2 ²⁶
15-8 SDCLKFS	SDCLK frequency select This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. In Single Data Rate mode (DDR_EN field of MIXERCTRL is '0') Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 While in Dual Data Rate mode (DDR_EN field of MIXERCTRL is '1') Only the following settings are allowed: 80h) Base clock divided by 512 40h) Base clock divided by 256 20h) Base clock divided by 128 10h) Base clock divided by 64 08h) Base clock divided by 32 04h) Base clock divided by 16 02h) Base clock divided by 8 01h) Base clock divided by 4

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00h) Base clock divided by 2 When the software changes the DDR_EN field, SDCLKFS might need to be changed also. In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: $\text{Clock Frequency} = (\text{Base Clock}) / (\text{prescaler} \times \text{divisor})$ For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h yields 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD clock frequency is 50 MHz and is never exceed this limit. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. If setting SDCLKFS and DVS can generate the same clock frequency,(for example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.
7-4 DVS	Divisor This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. Before changing clock divisor value (SDCLKFS or DVS), Host driver should make sure the SDSTB field is high. The settings are as follows: 0000b - Divide-by-1 0001b - Divide-by-2 1110b - Divide-by-15 1111b - Divide-by-16
3-0 —	Reserved Always write as 1.

50.9.1.14 Interrupt Status (INT_STATUS)

Offset

Register	Offset
INT_STATUS	30h

Function

An interrupt is generated when the normal interrupt signal enable is enabled and at least one of the status fields is set to 1. For all fields, writing 1 to a bit clears it; writing 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For card interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the card driver services the interrupt condition; otherwise, the CINT field is asserted again.

The table below shows the relationship between the command timeout error and the command complete.

Table 505. uSDHC status for command timeout error/command complete bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the transfer complete and the data timeout error.

Table 506. uSDHC status for data timeout error/transfer complete bit combinations

Transfer complete	Data timeout error	Meaning of the status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data transfer complete

The table below shows the relationship between the command CRC error and command timeout error.

Table 507. uSDHC status for command CRC error/command timeout error bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				W1C		W1C		W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TP	0	RTE	0			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		W1C		W1C				W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 DMAE	DMA error Occurs when an Internal DMA transfer has failed. This field is set to 1, when some error occurs in the data transfer. This error can be caused by either simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Because any error corrupts the whole data block, the host driver restarts the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 0b - No error 1b - Error
27 —	Reserved
26 TNE	Tuning error: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This field is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, host driver needs to abort a command executing and perform tuning.
25 —	Reserved
24 AC12E	Auto CMD12 error Occurs when detecting that one of the fields in the Auto CMD12 Error Status register has changed from 0 to 1. This field is set to 1, not only when the errors in Auto CMD12 occur, but also, when the Auto CMD12 is not executed due to the previous command error. 0b - No error 1b - Error

Table continues on the next page...

Table continued from the previous page...

Field	Function
23 —	Reserved
22 DEBE	Data end bit error Occurs either when detecting 0 at the end field position of read data that uses the DATA line, or at the end field position of the CRC. This field is not asserted in tuning process. 0b - No error 1b - Error
21 DCE	Data CRC error Occurs when detecting a CRC error when transferring read data that uses the DATA line, or when detecting the Write CRC status having a value other than 010. This field is not asserted in tuning process. 0b - No error 1b - Error
20 DTOE	Data timeout error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This field is not asserted in tuning process. 0b - No error 1b - Time out
19 CIE	Command index error Occurs if a command index error occurs in the command response. This field is not asserted in tuning process. 0b - No error 1b - Error
18 CEBE	Command end bit error Occurs when detecting that the end field of a command response is 0. This field is not asserted in tuning process. 0b - No error 1b - End bit error generated

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 CCE	Command CRC error Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this field is set when detecting a CRC error in the command response. • The uSDHC module detects a CMD line conflict by monitoring the CMD line when a command is issued. If uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then uSDHC aborts the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error should also be set to 1 to distinguish CMD line conflict. This field is not asserted in tuning process. 0b - No error 1b - CRC error generated
16 CTOE	Command timeout error Occurs only if no response is returned within 64 SDCLK cycles from the end field of the command. If uSDHC detects a CMD line conflict, in which case a Command CRC Error is also be set (as shown in Interrupt Status (INT_STATUS)), this field is set without waiting for 64 SDCLK cycles. This is because the command is aborted by uSDHC. This field is not asserted in tuning process. 0b - No error 1b - Time out
15 —	Reserved
14 TP	Tuning pass:(only for SD3.0 SDR104 mode and EMMC HS200 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 —	Reserved
12 RTE	Re-tuning event: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. host controller requests host driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 0b - Re-tuning is not required. 1b - Re-tuning should be performed.
11-9 —	Reserved
8	Card interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
CINT	This status field is set when an interrupt signal is detected from the external card. In 1-bit mode, uSDHC detects the Card Interrupt without the SD clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the host system. Writing this field to 1 can clear this field, but as the interrupt source from the SDIO card does not clear, this field is set again. to clear this field, it is required to reset the interrupt source from the external card followed by a writing 1 to this field. When this status has been set, and the host driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the host system. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal might not be asserted), write 1 to clear this field, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 0b - No card interrupt 1b - Generate card interrupt
7 CRM	Card removal This status field is set if the Card Inserted field in the Present State register changes from 1 to 0. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if no card is inserted. to leave it cleared, clear the Card Removal Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or inserted 1b - Card removed
6 CINS	Card insertion This status field is set if the Card Inserted field in the Present State register changes from 0 to 1. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if a card is inserted. to leave it cleared, clear the Card Inserted Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or removed 1b - Card inserted
5 BRR	Buffer read ready This status field is set if the Buffer Read Enable field, in the Present State register, changes from 0 to 1. See the Buffer Read Enable field in the Present State register for additional information. This field indicates that cmd19 is finished in tuning process. 0b - Not ready to read buffer 1b - Ready to read buffer
4 BWR	Buffer write ready This status field is set if the Buffer Write Enable field, in the Present State register, changes from 0 to 1. See the Buffer Write Enable field in the Present State register for additional information.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not ready to write buffer 1b - Ready to write buffer
3 DINT	DMA interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this field does not be set. Instead, the DMAE field is set. Either Simple DMA or ADMA finishes data transferring, this field is set. 0b - No DMA interrupt 1b - DMA interrupt is generated.
2 BGE	Block gap event If the Stop At Block Gap Request field in the Protocol Control register is set, this field is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this field is not set to 1. In the case of a Read Transaction: This field is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD bus timing). The Read Wait must be supported to use this function. In the case of Write Transaction: This field is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 0b - No block gap event 1b - Transaction stopped at block gap
1 TC	Transfer complete This field is set when a read or write transfer is completed. In the case of a Read Transaction: This field is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the host system). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request field in the Protocol Control register (after valid data has been read to the host system). In the case of a Write Transaction: This field is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request field in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this field is set when busy is deasserted. This field is not asserted in tuning process. 0b - Transfer does not complete 1b - Transfer complete
0 CC	Command complete This field is set when you receive the end field of the command response (except auto CMD12). See the Command Inhibit (CMD) in the Present State register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field is not asserted in tuning process. 0b - Command not complete 1b - Command complete

50.9.1.15 Interrupt Status Enable (INT_STATUS_EN)

Offset

Register	Offset
INT_STATUS_EN	34h

Function

Setting the fields in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any field is cleared, the corresponding Interrupt Status field is also cleared (that is, when the field in this register is cleared, the corresponding field in Interrupt Status register is always 0).

- Depending on IABG field setting, uSDHC might be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There are some delays on the Card Interrupt, asserted from the card, to the time the host system is informed.
- To detect a CMD line conflict, the host driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0	TNES	0	AC12E	0	DEBE	DCES	DTOE	CIESE	CEBE	CCES	CTOE
W						EN		SEN		SEN	EN	SEN	N	SEN	EN	SEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPSE	0	RTES	0				CRMS	CINSS	BRRS	BWRS	DINTS	BGES	TCSE	CCSE
W		N		EN					EN	EN	EN	EN	EN	EN	N	N
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 DMAESEN	DMA error status enable 0b - Masked 1b - Enabled
27 —	Reserved
26 TNESEN	Tuning error status enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12ESEN	Auto CMD12 error status enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBESEN	Data end bit error status enable 0b - Masked 1b - Enabled
21 DCESEN	Data CRC error status enable 0b - Masked 1b - Enabled
20 DTESEN	Data timeout error status enable 0b - Masked 1b - Enabled
19 CIESEN	Command index error status enable 0b - Masked 1b - Enabled
18 CEBESEN	Command end bit error status enable 0b - Masked 1b - Enabled
17	Command CRC error status enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
CCESSEN	0b - Masked 1b - Enabled
16 CTOESSEN	Command timeout error status enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPSEN	Tuning pass status enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTESSEN	Re-tuning event status enable 0b - Masked 1b - Enabled
11-9 —	Reserved
8 CINTSEN	Card interrupt status enable If this field is set to 0, uSDHC clears the interrupt request to the system. The Card Interrupt detection is stopped when this field is cleared and restarted when this field is set to 1. The host driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this field again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 0b - Masked 1b - Enabled
7 CRMSSEN	Card removal status enable 0b - Masked 1b - Enabled
6 CINSSEN	Card insertion status enable 0b - Masked 1b - Enabled
5 BRRSEN	Buffer read ready status enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Masked 1b - Enabled
4 BWRSEN	Buffer write ready status enable 0b - Masked 1b - Enabled
3 DINTSEN	DMA interrupt status enable 0b - Masked 1b - Enabled
2 BGESEN	Block gap event status enable 0b - Masked 1b - Enabled
1 TCSEN	Transfer complete status enable 0b - Masked 1b - Enabled
0 CCSEN	Command complete status enable 0b - Masked 1b - Enabled

50.9.1.16 Interrupt Signal Enable (INT_SIGNAL_EN)

Offset

Register	Offset
INT_SIGNAL_EN	38h

Function

This register is used to select which interrupt status is indicated to the host system as the interrupt. These status fields all share the same interrupt line. Setting any of these fields to 1 enables interrupt generation. The corresponding Status register field generates an interrupt when the corresponding interrupt signal enable field is set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAEI	0	TNEIE	0	AC12E	0	DEBEI	DCEIE	DTOEI	CIEIE	CEBEI	CCEIE	CTOEI
W				EN		N		IEN		EN	N	EN	N	EN	N	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPIEN	0	RTEIE	0			CINTI	CRMI	CINSI	BRRIE	BWRI	DINTI	BGEIE	TCIEN	CCIEN
W				N				EN	EN	EN	N	EN	EN	N		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 DMAEIEN	DMA error interrupt enable 0b - Masked 1b - Enable
27 —	Reserved
26 TNEIEN	Tuning error interrupt enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12EIEN	Auto CMD12 error interrupt enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBEIEN	Data end bit error interrupt enable 0b - Masked 1b - Enabled
21	Data CRC error interrupt enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DCEIEN	0b - Masked 1b - Enabled
20 DIOEIEN	Data timeout error interrupt enable 0b - Masked 1b - Enabled
19 CIEIEN	Command index error interrupt enable 0b - Masked 1b - Enabled
18 CEBEIEN	Command end bit error interrupt enable 0b - Masked 1b - Enabled
17 CCEIEN	Command CRC error interrupt enable 0b - Masked 1b - Enabled
16 CTOEIEN	Command timeout error interrupt enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPIEN	Tuning Pass interrupt enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTEIEN	Re-tuning event interrupt enable 0b - Masked 1b - Enabled
11-9 —	Reserved
8 CINTIEN	Card interrupt enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Masked 1b - Enabled
7 CRMIEN	Card removal interrupt enable 0b - Masked 1b - Enabled
6 CINSIEN	Card insertion interrupt enable 0b - Masked 1b - Enabled
5 BRRIEN	Buffer read ready interrupt enable 0b - Masked 1b - Enabled
4 BWRIEN	Buffer write ready interrupt enable 0b - Masked 1b - Enabled
3 DINTIEN	DMA interrupt enable 0b - Masked 1b - Enabled
2 BGEIEN	Block gap event interrupt enable 0b - Masked 1b - Enabled
1 TCIEN	Transfer complete interrupt enable 0b - Masked 1b - Enabled
0 CCIEN	Command complete interrupt enable 0b - Masked 1b - Enabled

50.9.1.17 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

Offset

Register	Offset
AUTOCMD12_ERR_STATUS	3Ch

Function

When the Auto CMD12 Error Status field in the Status register is set, the host driver checks this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in field 04-01. This register is valid only when the Auto CMD12 Error status field is set.

The table below shows the relationship between the Auto CMD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 508. Relationship between command CRC error and command timeout error for auto CMD12

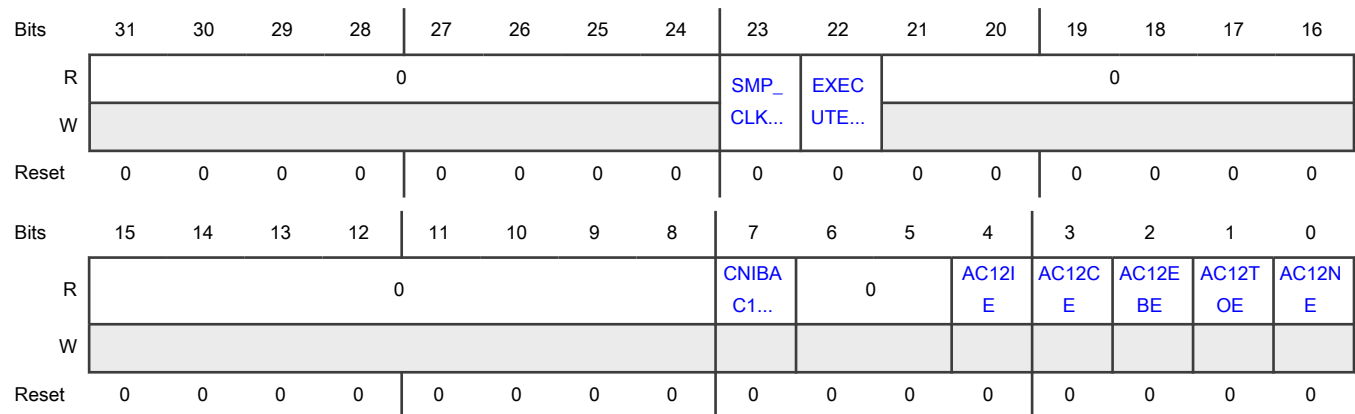
Auto CMD12 CRC error	Auto CMD12 timeout error	Type of error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

- When uSDHC is going to issue an Auto CMD12
 - Set field 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
 - Set field 0 to 0 if the Auto CMD12 is issued
- At the end field of an Auto CMD12 response
 - Check errors correspond to fields 1-4
 - Set fields 1-4 corresponding to detected errors
 - Clear fields 1-4 corresponding to detected errors
- Before reading the Auto CMD12 Error Status field 7
 - Set field 7 to 1 if there is a command that can't be issued
 - Clear field 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, field 7 is sampled when the driver is not writing to the Command register. So, it is suggested to read this register only when the AC12E field in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error fields (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 SMP_CLK_SEL	Sample clock select When std_tuning_en field is set, this field is used to select sampling clock to receive CMD and DATA. Otherwise, this field is reserved. This field is set by ty tuning procedure and valid after the completion of tuning(When Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Writing 1 to this field is meaningless and ignored. A tuning circuit is reset by writing to 0. This field can be cleared with setting Execute Tuning. Once the tuning circuit is reset, it takes time to complete tuning sequence. Therefore, host driver should keep this field to 1 to perform re-tuning sequence to complete re-tuning sequence in a short time. Change of this field is not allowed while the Host controller us receiving response or a read data block. 0b - Fixed clock is used to sample data 1b - Tuned clock is used to sample data
22 EXECUTE_TUNING	Execute tuning When std_tuning_en field is set, this field is used to start tuning procedure. Otherwise, this field is reserved. This field is set to start tuning procedure and automatically cleared when runing procedure is completed. The result of tuning is indicated to sam_clk_sel field. Tuning procedure is aborted by writing 0.
21-8 —	Reserved
7 CNIBAC12E	Command not issued by Auto CMD12 error Setting this field to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04-D01) in this register. 0b - No error 1b - Not issued

Table continues on the next page...

Table continued from the previous page...

Field	Function
6-5 —	Reserved
4 AC12IE	Auto CMD12 / 23 index error Occurs if the command index error occurs in response to a command. 0b - No error 1b - Error, the CMD index in response is not CMD12/23
3 AC12CE	Auto CMD12 / 23 CRC error Occurs when detecting a CRC error in the command response. 0b - No CRC error 1b - CRC error met in Auto CMD12/23 response
2 AC12EBE	Auto CMD12 / 23 end bit error Occurs when detecting that the end field of command response is 0 which should be 1. 0b - No error 1b - End bit error generated
1 AC12TOE	Auto CMD12 / 23 timeout error Occurs if no response is returned within 64 SDCLK cycles from the end field of the command. If this field is set to 1, the other error status fields (2-4) have no meaning. 0b - No error 1b - Time out
0 AC12NE	Auto CMD12 not executed If memory multiple block data transfer is not started, due to a command error, this field is not set because it is not necessary to issue an Auto CMD12. Setting this field to 1 means uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this field is set to 1, other error status fields (1-4) have no meaning. 0b - Executed 1b - Not executed

50.9.1.18 Host Controller Capabilities (HOST_CTRL_CAP)

Offset

Register	Offset
HOST_CTRL_CAP	40h

Function

This register provides the host driver with information specific to uSDHC implementation. The value in this register is the power-on-reset value and does not change with a software reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMA S	0	MBL		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RETUNING_M ODE	USE_T UN...	Reserv ed	TIME_COUNT_RETUNING				Reserved					DDR5 0_S...		SDR10 4_...	SDR50 _S...
W																
Reset	1	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-27 —	Reserved
26 VS18	Voltage support 1.8 V This field depends on the host system ability. 0b - 1.8 V not supported 1b - 1.8 V supported
25 VS30	Voltage support 3.0 V This field depends on the host system ability. 0b - 3.0 V not supported 1b - 3.0 V supported
24 VS33	Voltage support 3.3 V This field depends on the host system ability. 0b - 3.3 V not supported 1b - 3.3 V supported
23 SRS	Suspend / resume support This field indicates whether uSDHC supports Suspend / Resume functionality. If this field is 0, the Suspend and Resume mechanism, as well as the read wait, are not supported, and the host driver does not issue either Suspend or Resume commands.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not supported 1b - Supported
22 DMAS	DMA support This field indicates whether uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 0b - DMA not supported 1b - DMA supported
21 HSS	High speed support This field indicates whether uSDHC supports High Speed mode and the host system can supply a SD clock frequency from 25 MHz to 50 MHz. 0b - High speed not supported 1b - High speed supported
20 ADMAS	ADMA support This field indicates whether uSDHC supports the ADMA feature. 0b - Advanced DMA not supported 1b - Advanced DMA supported
19 —	Reserved
18-16 MBL	Max block length This value indicates the maximum block size that the host driver can read and write to the buffer in uSDHC. The buffer transfers block size without wait cycles. 000b - 512 bytes 001b - 1024 bytes 010b - 2048 bytes 011b - 4096 bytes
15-14 RETUNING_MODE	Retuning Mode This field selects retuning method. 00b - Mode 1 01b - Mode 2 10b - Mode 3 11b - Reserved
13	Use Tuning for SDR50

Table continues on the next page...

Table continued from the previous page...

Field	Function
USE_TUNING_SDR50	This field is set to 1. Host controller requires tuning to operate SDR50. 0b - SDR does not require tuning. 1b - SDR50 requires tuning.
12 —	Reserved
11-8 TIME_COUNT_RETUNING	Time counter for retuning This field indicates an initial value of the Retuning Timer for Re-Tuning Mode1 and 3. Setting to 0 disables Retuning Timer.
7-3 —	Reserved
2 DDR50_SUPPORT	DDR50 support This field indicates support of DDR50 mode.
1 SDR104_SUPPORT	SDR104 support This field indicates support of SDR104 mode.
0 SDR50_SUPPORT	SDR50 support This field indicates support of SDR50 mode.

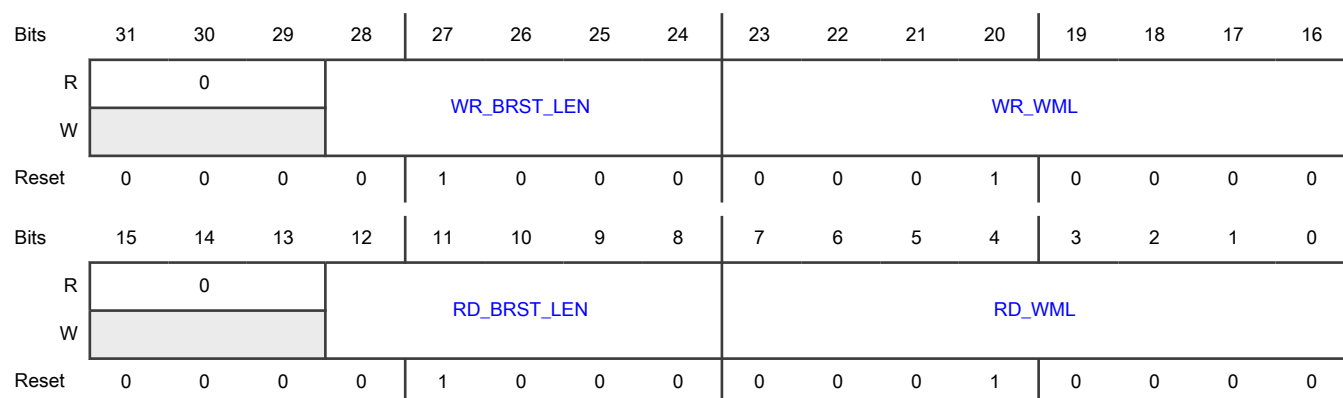
50.9.1.19 Watermark Level (WTMK_LVL)

Offset

Register	Offset
WTMK_LVL	44h

Function

Both write and read watermark levels (FIFO threshold) are configurable. Their value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. Their value can range from 1 to 31 words.

Diagram**Fields**

Field	Function
31-29 —	Reserved
28-24 WR_BRST_LEN	Write burst length due to system restriction, the actual burst length might not exceed 16 The number of words uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
23-16 WR_WML	Write watermark level The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also, the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15-13 —	Reserved
12-8 RD_BRST_LEN	Read burst length due to system restriction, the actual burst length might not exceed 16 The number of words uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
7-0 RD_WML	Read watermark level The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also, the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

50.9.1.20 Mixer Control (MIX_CTRL)

Offset

Register	Offset
MIX_CTRL	48h

Function

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 509. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserv	Reserv	0	0	HS400	FBCLK	AUTO	SMP_	EXE_	0					
W	ed	ed	ed			_M...	_S...	_TU...	CLK...	TUNE						
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AC23E	NIBBL	MSBS	DTDS	DDR_	AC12E	BCEN	DMAE
W									N	E_...	EL	EL	EN	N		N
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Reserved
—	Always write as 1.
30	Reserved
—	Always write as 0.
29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Always write as 0.
28 —	Reserved
27 —	Reserved
26 HS400_MODE	Enable HS400 mode HS400 Enable
25 FBCLK_SEL	Feedback clock source selection (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Feedback clock comes from the loopback CLK 1b - Feedback clock comes from the ipp_card_clk_out
24 AUTO_TUNE_EN	Auto tuning enable (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Disable auto tuning 1b - Enable auto tuning
23 SMP_CLK_SEL	Clock selection When STD_TUNING_EN is 0, this field is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Fixed clock is used to sample data / cmd 1b - Tuned clock is used to sample data / cmd
22 EXE_TUNE	Execute tuning: (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) When STD_TUNING_EN is 0, this field is set to 1 to indicate the host driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 0b - Not tuned or tuning completed 1b - Execute tuning
21-8 —	Reserved
7 AC23EN	Auto CMD23 enable When this field is set to 1, the host controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	Nibble position indication In DDR 4-bit mode nibble position indication. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 MSBSEL	Multi / Single block select This field enables multiple block DATA line data transfers. For any other commands, this field can be set to 0. If this field is 0, it is not necessary to set the Block Count register. See Command Transfer Type (CMD_XFR_TYP). 0b - Single block 1b - Multiple blocks
4 DTDSEL	Data transfer direction select This field defines the direction of DATA line data transfers. The field is set to 1 by the host driver to transfer data from the SD card to uSDHC and is set to 0 for all other commands. 0b - Write (Host to card) 1b - Read (Card to host)
3 DDR_EN	Dual data rate mode selection
2 AC12EN	Auto CMD12 enable Multiple block transfers for memory require a CMD12 to stop the transaction. When this field is set to 1, uSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver is not set this field to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, uSDHC ignores this field no matter it is set or not. 0b - Disable 1b - Enable
1 BCEN	Block count enable This field is used to enable the Block Count register, which is only relevant for multiple block transfers. When this field is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 0b - Disable 1b - Enable
0 DMAEN	DMA enable This field enables DMA functionality. If this field is set to 1, a DMA operation begins when the host driver sets the DPSEL field of this register. Whether the simple DMA or the advanced DMA is active depends on the DMA Select field of the Protocol Control register. 0b - Disable 1b - Enable

50.9.1.21 Force Event (FORCE_EVENT)

Offset

Register	Offset
FORCE_EVENT	50h

Function

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status register can be written if the corresponding field of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding field of Interrupt Status register. A read from this register always results in 0's. to change the corresponding status fields in the Interrupt Status register, make sure to set IPGEN field in System Control Register so that peripheral clock is always active.

Forcing a card interrupt generates a short pulse on the DATA1 line, and the driver might treat this interrupt as a normal interrupt. The interrupt service routine might skip polling the card interrupt factor as the interrupt is self cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W	FEVT CINT			FEVT DMAE		FEVT NE		FEVT AC1...		FEVT DEBE	FEVT DCE	FEVT DCE	FEVT DCE	FEVT CIE	FEVT CEBE	FEVT CCE	FEVT CTOE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	0	0	0	0	0	0	0
W									FEVT CNI...			FEVT C1...	FEVT C1...	FEVT C1...	FEVT C1...	FEVT C1...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 FEVTCINT	Force event card interrupt Writing 1 to this field generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT field is set and the interrupt service routine might treat this interrupt as a normal interrupt from the external card.
30-29 —	Reserved
28 FEVTDMAE	Force event DMA error Forces the DMAE field of Interrupt Status register to be set.
27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26 FEVTTNE	Force tuning error Forces the TNE field of Interrupt Status register to be set.
25 —	Reserved
24 FEVTAC12E	Force event Auto Command 12 error Forces the AC12E field of Interrupt Status register to be set.
23 —	Reserved
22 FEVTDEBE	Force event data end bit error Forces the DEBE field of Interrupt Status register to be set.
21 FEVTDCE	Force event data CRC error Forces the DCE field of Interrupt Status register to be set.
20 FEVTDTOE	Force event data time out error Force the DTOE field of Interrupt Status register to be set.
19 FEVTCIE	Force event command index error Forces the CCE field of Interrupt Status register to be set.
18 FEVTCBE	Force event command end bit error Forces the CEBE field of Interrupt Status register to be set.
17 FEVTCCE	Force event command CRC error Forces the CCE field of Interrupt Status register to be set.
16 FEVTCCTOE	Force event command time out error Forces the CTOE field of Interrupt Status register to be set.
15-8 —	Reserved
7 FEVTCNIBAC12E	Force event command not executed by Auto Command 12 error Forces the CNIBAC12E field in the Auto Command12 Error Status register to be set.
6-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4 FEVTAC12IE	Force event Auto Command 12 index error Forces the AC12IE field in the Auto Command12 Error Status register to be set.
3 FEVTAC12EBE	Force event Auto Command 12 end bit error Forces the AC12EBE field in the Auto Command12 Error Status register to be set.
2 FEVTAC12CE	Force event auto command 12 CRC error Forces the AC12CE field in the Auto Command12 Error Status register to be set.
1 FEVTAC12TOE	Force event auto command 12 time out error Forces the AC12TOE field in the Auto Command12 Error Status register to be set.
0 FEVTAC12NE	Force event auto command 12 not executed Forces the AC12NE field in the Auto Command12 Error Status register to be set.

50.9.1.22 ADMA Error Status (ADMA_ERR_STATUS)

Offset

Register	Offset
ADMA_ERR_STATUS	54h

Function

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the host driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and does not generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

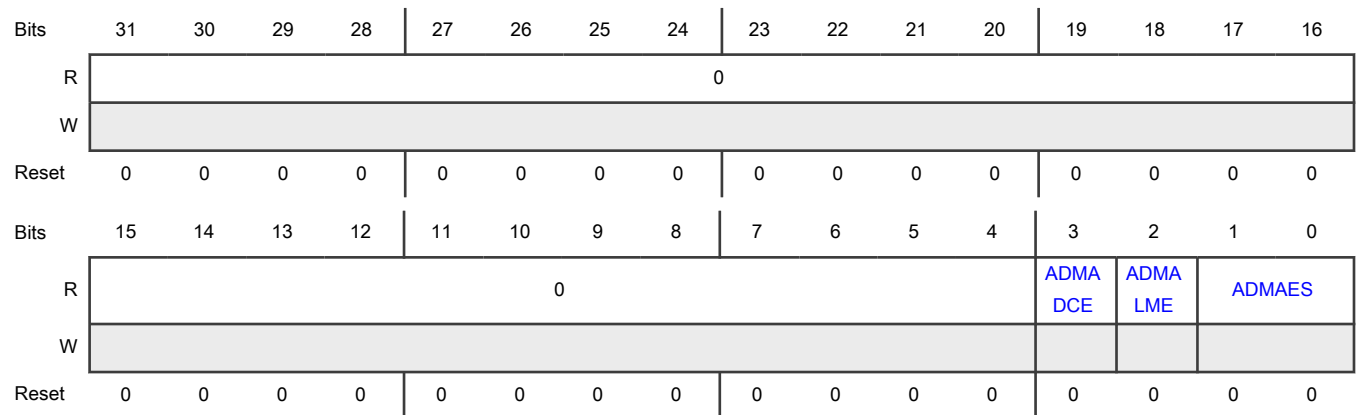
In case of a write operation, the host driver should use the ACMD22 to get the number of the written block, rather than using this information, because unwritten data might exist in the host controller.

The host controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The host driver can distinguish this error by reading the Valid field of the error descriptor.

Table 510. ADMA error state coding

D01-D00	ADMA error state (when error has occurred)	Contents of ADMA System Address register
00	ST_STOP (Stop DMA)	Holds the address of the next executable descriptor command
01	ST_FDS (Fetch descriptor)	Holds the valid descriptor address
10	ST_CADR (Change address)	No ADMA error is generated
11	ST_TFR (Transfer data)	Holds the address of the next executable descriptor command

Diagram



Fields

Field	Function
31-4 —	Reserved
3 ADMADCE	ADMA descriptor error This error occurs when invalid descriptor fetched by ADMA. 0b - No error 1b - Error
2 ADMALME	ADMA length mismatch error This error occurs in the following two cases: - While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length. - Total data length cannot be divided by the block length.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No error 1b - Error
1-0 ADMAES	ADMA error state (when ADMA error is occurred) This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. See ADMA Error Status (ADMA_ERR_STATUS) for more details.

50.9.1.23 ADMA System Address (ADMA_SYS_ADDR)

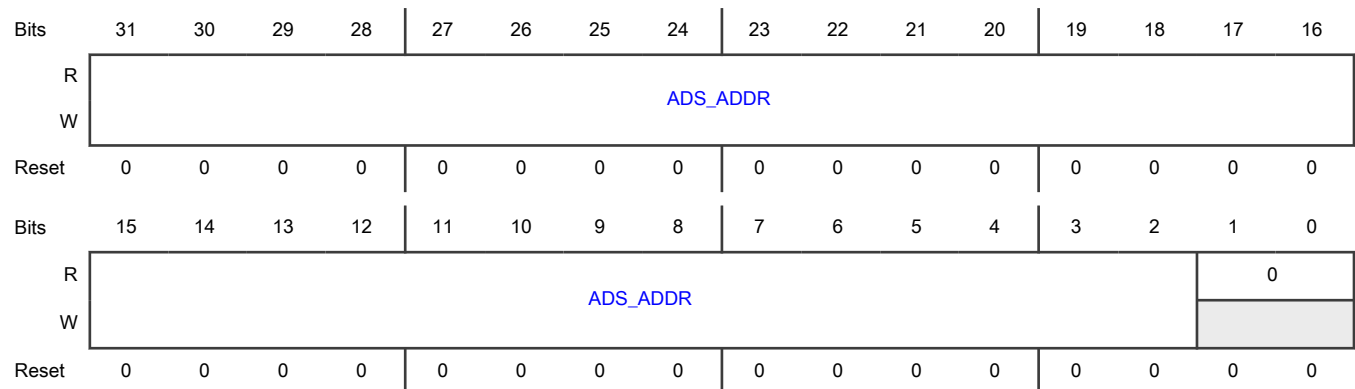
Offset

Register	Offset
ADMA_SYS_ADDR	58h

Function

This register contains the physical system memory address used for ADMA transfers.

Diagram



Fields

Field	Function
31-2 ADS_ADDR	ADMA system address This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the host driver sets the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions .
1-0 —	Reserved

50.9.1.24 DLL (Delay Line) Control (DLL_CTRL)

Offset

Register	Offset
DLL_CTRL	60h

Function

This register contains control fields for DLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DLL_CTRL_REF_UPDATE_INT				DLL_CTRL_SLV_UPDATE_INT								0	DLL_CTRL_SLV_DLY_TARGET1			
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	DLL_CTRL_SLV_OVERRIDE_VAL								DLL_C	DLL_C	DLL_CTRL_SLV_DLY_TARGET0				DLL_C	DLL_C	DLL_C
W										TR...					TR...		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31-28 DLL_CTRL_REF_UPDATE_INT	DLL control loop update interval The interval cycle is $(2 + \text{REF\_UPDATE\_INT}) * \text{REF\_CLOCK}$. By default, the DLL control loop updates every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that might effect the delay (such as voltage and temperature)
27-20	Slave delay line update interval

Table continues on the next page...

Table continued from the previous page...

Field	Function
DLL_CTRL_SLV_UPDATE_INT	If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line also updates automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 —	Reserved
18-16 DLL_CTRL_SLV_DLY_TARGET1	DLL slave delay target1 See DLL_CTRL_SLV_DLY_TARGET0 below.
15-9 DLL_CTRL_SLV_OVERRIDE_VAL	DLL slave override val When SLV_OVERRIDE = 1, this field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.
8 DLL_CTRL_SLV_OVERRIDE	DLL slave override Set this field to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE field. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7 DLL_CTRL_GATE_UPDATE	DLL gate update Set this field to 1 to prevent the DLL from updating (because when clock_in exists, glitches might appear during DLL updates). This field might be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6-3 DLL_CTRL_SLV_DLY_TARGET0	DLL slave delay target0 The delay target for uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is $((\{DLL\_CTRL\_SLV\_DLY\_TARGET1, DLL\_CTRL\_SLV\_DLY\_TARGET0\} + 1) * REF\_CLOCK / 2) / 16$ So the input read-clock can be delayed relative input data from $(REF\_CLOCK / 2) / 16$ to $REF\_CLOCK * 4$.
2 DLL_CTRL_SLV_FORCE_UPDATE	DLL slave delay line Setting this field to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line updates automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function might not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_RESET	DLL reset Setting this field to 1 force a reset on DLL. This causes the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so to create a subsequent reset, RESET must be taken low and then asserted again.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0	DLL and delay chain
DLL_CTRL_ENABLE	Set this field to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

50.9.1.25 DLL Status (DLL_STATUS)

Offset

Register	Offset
DLL_STATUS	64h

Function

This register contains the DLL status information. All fields are read only and reads the same as the power-reset value.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_STS_REF_SEL								DLL_STS_SLV_SEL						DLL_S TS...	DLL_S TS...
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-9 DLL_STS_REF_SEL	Reference delay line select taps This is encoded by 7 fields for 127 taps.
8-2	Slave delay line select status

Table continues on the next page...

Table continued from the previous page...

Field	Function
DLL_STS_SLV_SEL	This is the instant value generated from reference chain. Because the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

50.9.1.26 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)

Offset

Register	Offset
CLK_TUNE_CTRL_STATUS	68h

Function

This register contains the Clock Tuning Control status information. All fields are read only and reads the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode and EMMC HS200 mode.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE								TAP_SEL_OUT				TAP_SEL_POST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE								DLY_CELL_SET_OUT				DLY_CELL_SET_POST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PRE_ERR	PRE error PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
30-24 TAP_SEL_PRE	TAP_SEL_PRE Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE is updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23-20 TAP_SEL_OUT	Delay cells added on the feedback clock between CLK_PRE and CLK_OUT Reflects the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19-16 TAP_SEL_POST	Delay cells added on the feedback clock between CLK_OUT and CLK_POST Reflects the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14-8 DLY_CELL_SET_PRE	delay cells on the feedback clock between the feedback clock and CLK_PRE Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7-4 DLY_CELL_SET_OUT	Delay cells on the feedback clock between CLK_PRE and CLK_OUT Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
3-0 DLY_CELL_SET_POST	Delay cells on the feedback clock between CLK_OUT and CLK_POST Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

50.9.1.27 Strobe DLL control (STROBE_DLL_CTRL)**Offset**

Register	Offset
STROBE_DLL_CTRL	70h

Function

This register contains the strobe DLL control information.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STROBE_DLL_CTRL_REF_UPDATE_INT				STROBE_DLL_CTRL_SLV_UPDATE_INT								0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STROBE_DLL_CTRL_SLV_OVERRIDE_VAL								STROBE...	STROBE...	STROBE...	STROBE_DLL_CTRL_SLV_DELAY...		STROBE...	STROBE...	STROBE...
W									BE...	BE...	BE...			BE...	BE...	BE...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 STROBE_DLL_CTRL_REF_UPDATE_INT	Strobe DLL control reference update interval The interval cycle is $(2 + \text{STROBE\_REF\_UPDATE\_INT}) * \text{STROBE\_REF\_CLOCK}$. By default, the DLL control loop updates every two STROBE_REF_CLOCK cycles. NOTE Increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that might effect the delay (such as voltage and temperature)
27-20 STROBE_DLL_CTRL_SLV_UPDATE_INT	Strobe DLL control slave update interval Slave delay line update interval. If default 0 is used, it means 256 cycles of STROBE_REF_CLOCK. A value of 0x0F results in 15 cycles and so on. NOTE software can always cause an update of the slave-delay line using the STROBE_SLV_FORCE_UPDATE register. The slave delay line also updates automatically when the reference DLL transitions to a locked state (from an un-locked state).
19-16 —	This field is reserved. This read-only field is reserved and always has the value 0.
15-9 STROBE_DLL_CTRL_SLV_OVERRIDE_VAL	Strobe DLL control slave Override value When STROBE_SLV_OVERRIDE = 1, this field is used to manually select one of 128 physical taps. A value of 0 selects tap 1, and a value of 0x7F selects tap 128.
8 STROBE_DLL_CTRL_SLV_OVERRIDE	Strobe DLL control slave override Set this field to 1 to Enable manual override for slave delay chain using STROBE_SLV_OVERRIDE_VAL; set this field to 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE field. In fact to reduce power, if STROBE_SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 STROBE_DLL_CTRL_GATE_UPDATE_1	Strobe DLL control gate update Set this field to 1 to prevent the DLL from updating (because when STROBE_CLOCK_IN exists, glitches might appear during DLL updates). This field can be used by software if such a condition occurs. Clear the field to 0 to allow the DLL to update automatically.
6 STROBE_DLL_CTRL_GATE_UPDATE_0	Strobe DLL control gate update Set this field to 1 to prevent the DLL from updating (because when STROBE_CLOCK_IN exists, glitches might appear during DLL updates). This field can be used by software if such a condition occurs. Clear the field to 0 to allow the DLL to update automatically.
5-3 STROBE_DLL_CTRL_SLV_DELAY_TARGET	Strobe DLL Control Slave Delay Target The delay target for uSDHC loopback read clock can be programmed in 1/16th increments of an STROBE_REF_CLOCK half-period. The delay is $((\text{STROBE\_DLL\_CTRL\_SLV\_DELAY\_TARGET} + 1) * \text{STROBE\_REF\_CLOCK} / 2) / 16$. So the input read-clock can be delayed relative input data from $(\text{STROBE\_REF\_CLOCK} / 2) / 16$ to $(\text{STROBE\_REF\_CLOCK} * 4) / 16$.
2 STROBE_DLL_CTRL_SLV_FORCE_UPDATE	Strobe DLL control slave force updated Setting this field to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line should automatically update the STROBE_SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if STROBE_SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function might not work when uSDHC is working on data / cmd / response.
1 STROBE_DLL_CTRL_RESET	Strobe DLL control reset Setting this field to 1 to force a reset on DLL. This causes the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, to create a subsequent reset, RESET must be taken low and then asserted again.
0 STROBE_DLL_CTRL_ENABLE	Strobe DLL control enable Set this field to 1 to enable the DLL and delay chain; otherwise, set to 0 to bypasses DLL. NOTE Using the slave delay line override feature with STROBE_SLV_OVERRIDE and STROBE_SLV_OVERRIDE_VAL, the DLL does not need to be enabled.

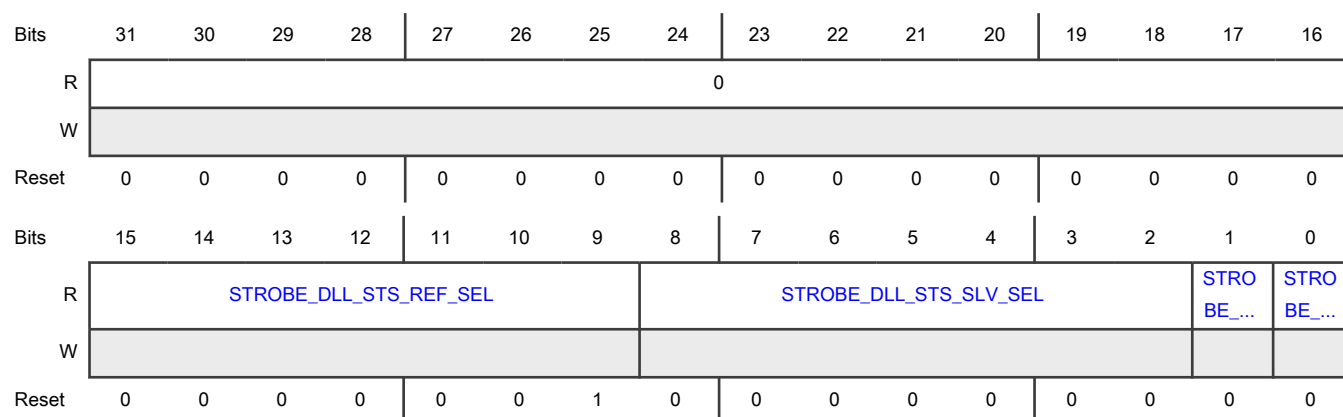
50.9.1.28 Strobe DLL status (STROBE_DLL_STATUS)

Offset

Register	Offset
STROBE_DLL_STATUS	74h

Function

This register contains the strobe DLL status information. All fields are read only and read the same as the power-reset value.

Diagram**Fields**

Field	Function
31-16 —	This field is reserved. This read-only field is reserved and always has the value 0.
15-9 STROBE_DLL_STS_REF_SEL	Strobe DLL status reference select Reference delay line select taps. This is encoded by 7 fields for 127 taps.
8-2 STROBE_DLL_STS_SLV_SEL	Strobe DLL status slave select Slave delay line select status. This is the instant value generated from reference chain. Because the reference chain can only be updated when STROBE_REF_CLOCK is detected, this value can be updated with the right value when the reference is locked.
1 STROBE_DLL_STS_REF_LOCK	Strobe DLL status reference lock This signifies that the DLL has detected and locked to a half-phase REF_CLOCK shift, it allows the slave delay-line to perform programmed clock delays.
0 STROBE_DLL_STS_SLV_LOCK	Strobe DLL status slave lock Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line, and the slave-delay line is implementing the programmed delay value.

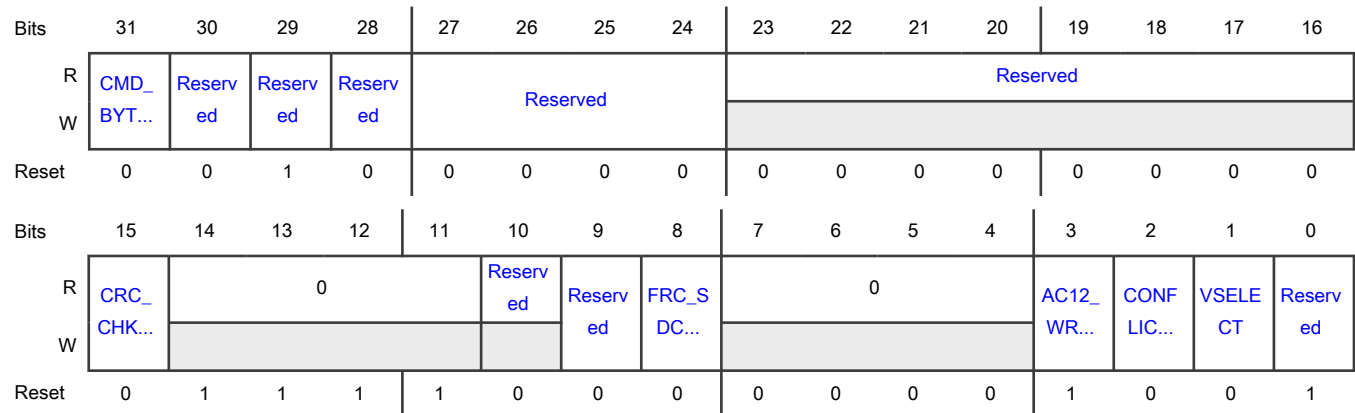
50.9.1.29 Vendor Specific Register (VEND_SPEC)**Offset**

Register	Offset
VEND_SPEC	C0h

Function

This register contains the vendor specific control / status register.

Diagram



Fields

Field	Function
31 CMD_BYTE_EN	Byte access This bit controls the byte access. 0b - Disable 1b - Enable
30 —	Reserved Always write as 0.
29 —	Reserved Always write as 1
28 —	Reserved Always write as 0.
27-24 —	Reserved Always write as 4'b0000.
23-16 —	Reserved Always write as 8'h00.
15 CRC_CHK_DIS	CRC Check Disable 0b - Check CRC16 for every read data packet and check CRC fields for every write data packet 1b - Ignore CRC16 check for every read data packet and ignore CRC fields check for every write data packet

Table continues on the next page...

Table continued from the previous page...

Field	Function
14-11 —	Reserved Reserved
10 —	Reserved Always write as 0.
9 —	Reserved Always write as 0.
8 FRC_SDCLK_ON	Force CLK Force CLK output active 0b - CLK active or inactive is fully controlled by the hardware. 1b - Force CLK active
7-4 —	Reserved
3 AC12_WR_CHK_BUSY_EN	Check busy enable Check busy enable after auto CMD12 for write data packet 0b - Do not check busy after auto CMD12 for write data packet 1b - Check busy after auto CMD12 for write data packet
2 CONFLICT_CHK_EN	Conflict check enable It is not implemented in uSDHC IP. 0b - Conflict check disable 1b - Conflict check enable
1 VSELECT	Voltage selection Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 0b - Change the voltage to high voltage range, around 3.0 V 1b - Change the voltage to low voltage range, around 1.8 V
0 —	Reserved Always write as 0.

50.9.1.30 MMC Boot (MMC_BOOT)

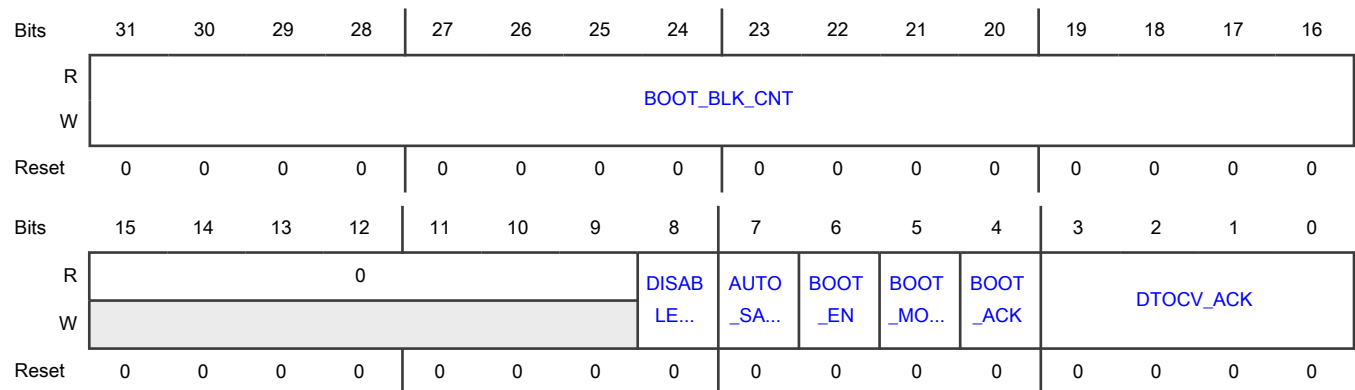
Offset

Register	Offset
MMC_BOOT	C4h

Function

This register contains the MMC Fast Boot control register.

Diagram



Fields

Field	Function
31-16 BOOT_BLK_CNT	Stop At Block Gap value of automatic mode The value defines the Stop At Block Gap value of automatic mode. When received, card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Attributes Register, field 31 - 16 of 0x04.
15-9 —	Reserved
8 DISABLE_TIME_OUT	Time out NOTE When this field is set, there is no timeout check no matter whether BOOT_EN is set or not. 0b - Enable time out 1b - Disable time out
7 AUTO_SABG_EN	Auto stop at block gap During boot, enable auto stop at block gap function. This function is triggered, and host stops at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 BOOT_EN	Boot enable Boot mode enable 0b - Fast boot disable 1b - Fast boot enable
5 BOOT_MODE	Boot mode Boot mode select 0b - Normal boot 1b - Alternative boot
4 BOOT_ACK	BOOT ACK Boot ACK mode select 0b - No ack 1b - Ack
3-0 DTCV_ACK	Boot ACK time out Boot ACK time out counter value. 0000b - SDCLK x 2 ¹⁴ 0001b - SDCLK x 2 ¹⁵ 0010b - SDCLK x 2 ¹⁶ 0011b - SDCLK x 2 ¹⁷ 0100b - SDCLK x 2 ¹⁸ 0101b - SDCLK x 2 ¹⁹ 0110b - SDCLK x 2 ²⁰ 0111b - SDCLK x 2 ²¹ 1110b - SDCLK x 2 ²⁸ 1111b - SDCLK x 2 ²⁹

50.9.1.31 Vendor Specific 2 Register (VEND_SPEC2)

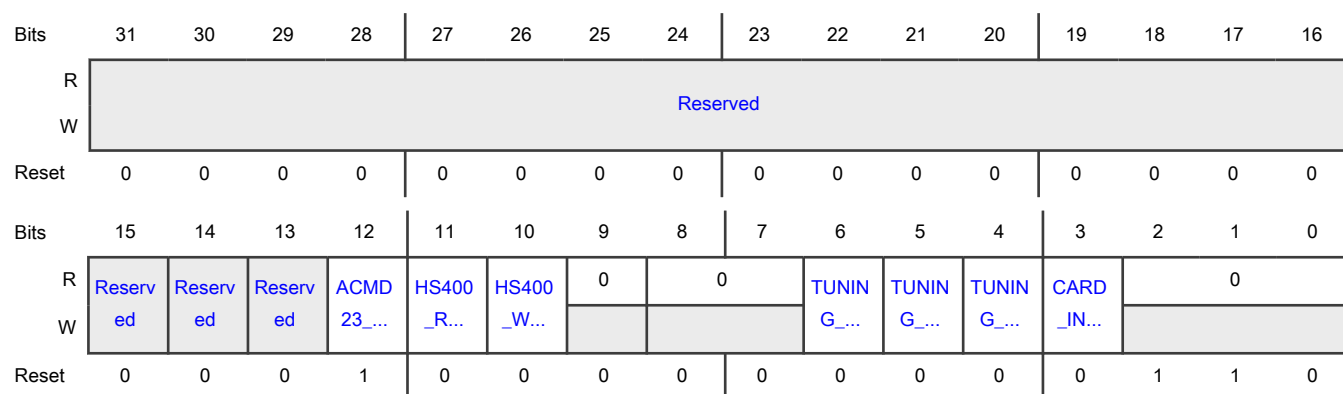
Offset

Register	Offset
VEND_SPEC2	C8h

Function

This register contains the vendor specific control 2 register.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 —	Reserved
13 —	Reserved
12 ACMD23_ARG U2_EN	Argument2 register enable for ACMD23 0b - Disable 1b - Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enabled.
11 HS400_RD_CL K_STOP_EN	HS400 read clock stop enable Only stop clock at read block gap.
10 HS400_WR_CL K_STOP_EN	HS400 write clock stop enable Only stop clock at write block gap.
9 —	Reserved
8-7 —	Reserved
6	Tuning command enable Enable the auto tuning circuit to check the CMD line.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TUNING_CMD_EN	0b - Auto tuning circuit does not check the CMD line. 1b - Auto tuning circuit checks the CMD line.
5 TUNING_1bit_EN	Tuning 1bit enable Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.
4 TUNING_8bit_EN	Tuning 8bit enable Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. 0b00 - Tuning circuit only checks the DATA[3:0] 0b01 - Tuning circuit only checks the DATA0 0b10 - Tuning circuit checks the whole DATA[7:0] 0b11 - Invalid NOTE The format of these two fields are [TUNNING_8bit_EN:TUNNING_1bit_EN].
3 CARD_INT_D3_TEST	Card interrupt detection test This field only uses for debugging. 0b - Check the card interrupt only when DATA3 is high. 1b - Check the card interrupt by ignoring the status of DATA3.
2-0 —	Reserved

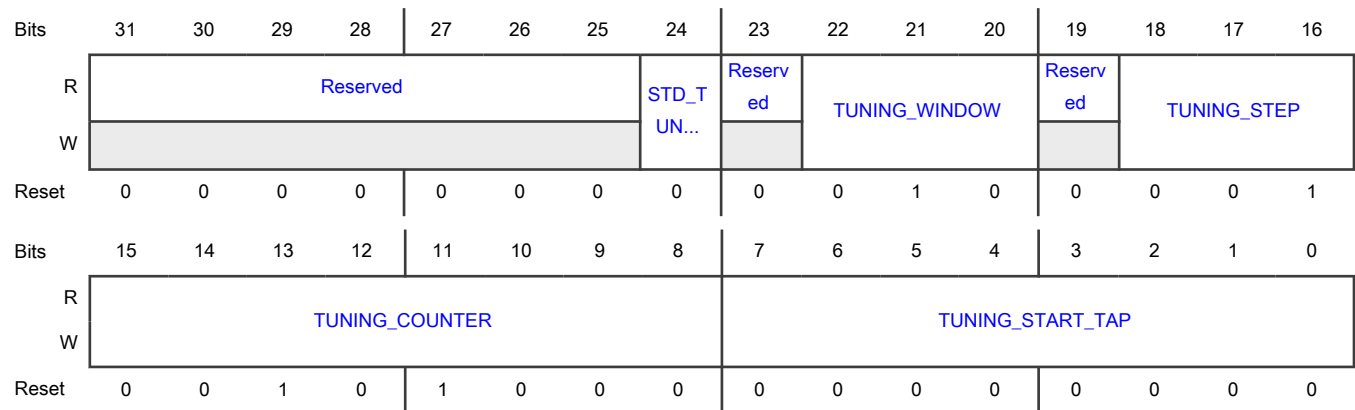
50.9.1.32 Tuning Control (TUNING_CTRL)

Offset

Register	Offset
TUNING_CTRL	CCh

Function

The register contains configuration of tuning circuit.

Diagram**Fields**

Field	Function
31-25 —	Reserved
24 STD_TUNING_ EN	Standard tuning circuit and procedure enable This field is used to enable standard tuning circuit and procedure.
23 —	Reserved
22-20 TUNING_WIND OW	Data window Select data window value for auto tuning
19 —	Reserved
18-16 TUNING_STEP	TUNING_STEP The increasing delay cell steps in tuning procedure.
15-8 TUNING_COU NTER	Tuning counter The MAX repeat CMD19 times in tuning procedure.
7-0 TUNING_STAR T_TAP	Tuning start The start delay cell point when send first CMD19 in tuning procedure.

50.9.2 uSDHC register descriptions

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map of uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

50.9.2.1 uSDHC memory map

USDHC1 base address: 4013_7000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA System Address (DS_ADDR)	32	RW	0000_0000h
4h	Block Attributes (BLK_ATT)	32	RW	0001_0000h
8h	Command Argument (CMD_ARG)	32	RW	0000_0000h
Ch	Command Transfer Type (CMD_XFR_TYP)	32	RW	0000_0000h
10h	Command Response0 (CMD_RSP0)	32	R	0000_0000h
14h	Command Response1 (CMD_RSP1)	32	R	0000_0000h
18h	Command Response2 (CMD_RSP2)	32	R	0000_0000h
1Ch	Command Response3 (CMD_RSP3)	32	R	0000_0000h
20h	Data Buffer Access Port (DATA_BUFF_ACC_PORT)	32	RW	0000_0000h
24h	Present State (PRES_STATE)	32	R	See section
28h	Protocol Control (PROT_CTRL)	32	RW	0880_0020h
2Ch	System Control (SYS_CTRL)	32	RW	0080_800Fh
30h	Interrupt Status (INT_STATUS)	32	RW	0000_0000h
34h	Interrupt Status Enable (INT_STATUS_EN)	32	RW	0000_0000h
38h	Interrupt Signal Enable (INT_SIGNAL_EN)	32	RW	0000_0000h
3Ch	Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)	32	RW	0000_0000h
40h	Host Controller Capabilities (HOST_CTRL_CAP)	32	RW	07F3_B407h
44h	Watermark Level (WTMK_LVL)	32	RW	0810_0810h
48h	Mixer Control (MIX_CTRL)	32	RW	8000_0000h
50h	Force Event (FORCE_EVENT)	32	RW	0000_0000h
54h	ADMA Error Status (ADMA_ERR_STATUS)	32	R	0000_0000h
58h	ADMA System Address (ADMA_SYS_ADDR)	32	RW	0000_0000h
60h	DLL (Delay Line) Control (DLL_CTRL)	32	RW	0000_0000h
64h	DLL Status (DLL_STATUS)	32	R	0000_0200h
68h	CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)	32	RW	0000_0000h
C0h	Vendor Specific Register (VEND_SPEC)	32	RW	2000_7809h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C4h	MMC Boot (MMC_BOOT)	32	RW	0000_0000h
C8h	Vendor Specific 2 Register (VEND_SPEC2)	32	RW	0000_1006h
CCh	Tuning Control (TUNING_CTRL)	32	RW	0021_2800h

50.9.2.2 DMA System Address (DS_ADDR)

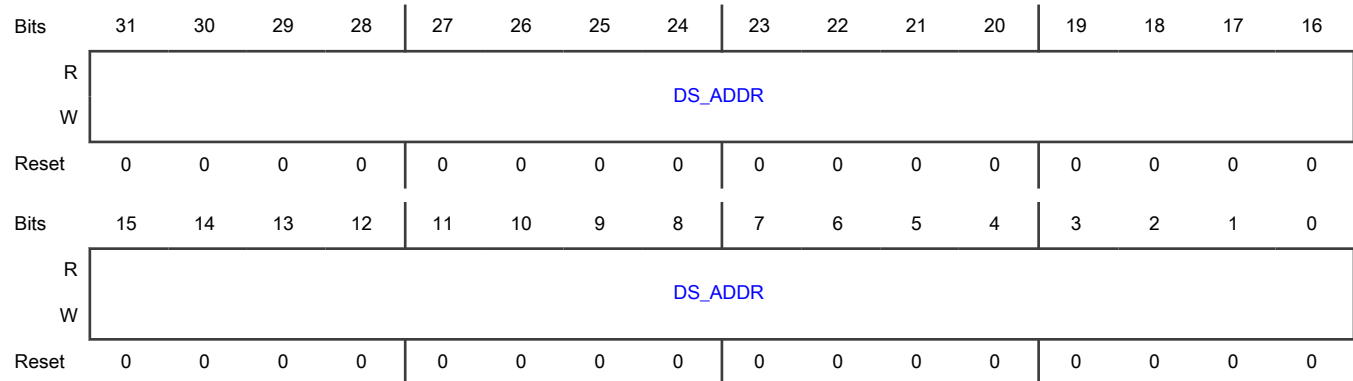
Offset

Register	Offset
DS_ADDR	0h

Function

This register contains the physical system memory address used for DMA transfers.

Diagram



Fields

Field	Function
31-0 DS_ADDR	System address DMA system address / argument 2 When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23. 1. SDMA system address

Table continues on the next page...

Field	Function
	Because the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The host driver waits until the DLA bit in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC automatically changes SEQ burst type to NSEQ. Because this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC bit is set. Such restriction is also listed in Software restrictions. 2. Argument 2 This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

50.9.2.3 Block Attributes (BLK_ATT)

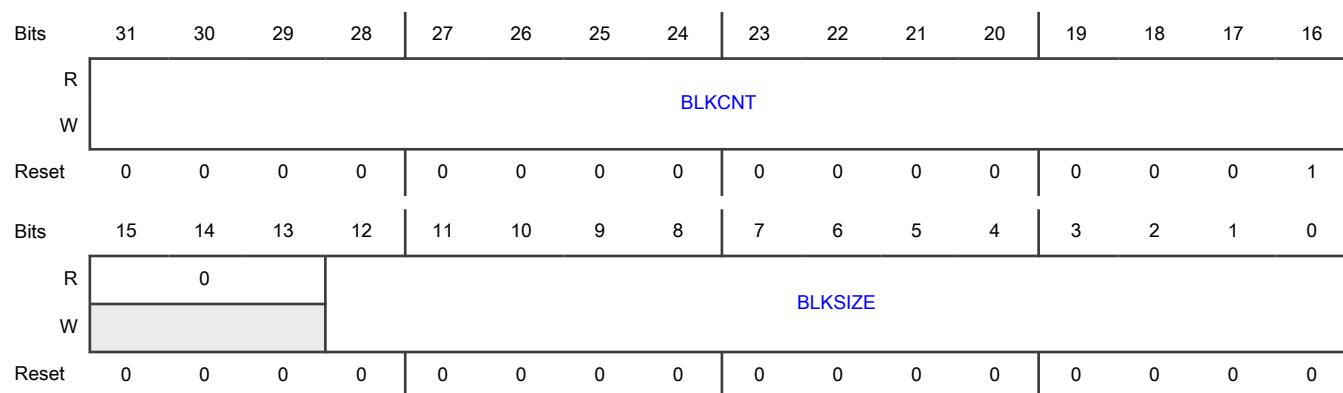
Offset

Register	Offset
BLK_ATT	4h

Function

This register is used to configure the number of data blocks and the number of bytes in each block.

Diagram



Fields

Field	Function
31-16 BLKCNT	Blocks count for current transfer This register is enabled when the Block Count Enable field in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register always reads as 1. The host driver sets this register to a value between 1 and the maximum block count. The uSDHC module decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to zero results in no data blocks being transferred. This register should be accessed only when no transaction is executing (that is, after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When saving transfer content because of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when the Suspend command is sent out, uSDHC treats the current transfer as aborted and change the BLKCNT register back to its original value instead of keeping the dynamical indicator of the remaining block count. When restoring transfer content prior to issuing a Resume command, the host driver restores the previously saved block count. NOTE Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL field is indicating a single block transfer, the read value of BLKCNT is always 1. 0000_0000_0000_0000b - Stop count 0000_0000_0000_0001b - 1 block 0000_0000_0000_0010b - 2 blocks 1111_1111_1111_1111b - 65535 blocks
15-13 —	Reserved
12-0 BLKSIZE	Transfer block size This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations are ignored. 0_0000_0000_0000b - No data transfer 0_0000_0000_0001b - 1 byte 0_0000_0000_0010b - 2 bytes 0_0000_0000_0011b - 3 bytes 0_0000_0000_0100b - 4 bytes

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0001_1111_1111b - 511 bytes 0_0010_0000_0000b - 512 bytes 0_1000_0000_0000b - 2048 bytes 1_0000_0000_0000b - 4096 bytes

50.9.2.4 Command Argument (CMD_ARG)

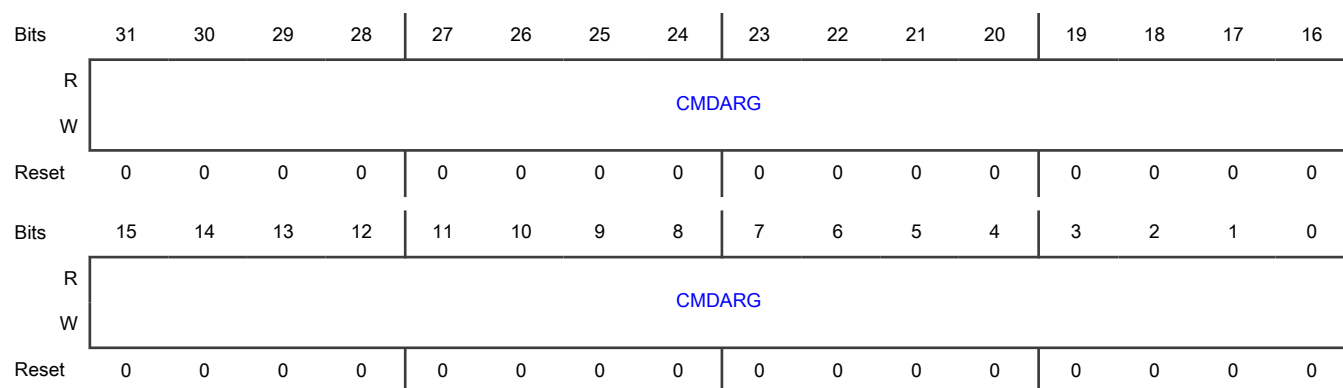
Offset

Register	Offset
CMD_ARG	8h

Function

This register contains the SD/MMC command argument.

Diagram



Fields

Field	Function
31-0	Command argument
CMDARG	The SD/MMC command argument is specified as bits 39-8 of the command format in the SD or MMC specification. This register is write protected when the Command Inhibit (CMD) field in the Present State register is set.

50.9.2.5 Command Transfer Type (CMD_XFR_TYP)

Offset

Register	Offset
CMD_XFR_TYP	Ch

Function

This register is used to control the operation of data transfers. The host driver sets this register before issuing a command followed by a data transfer or before issuing a Resume command. To prevent data loss, uSDHC prevents writing to the bits, which are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver checks the Command Inhibit DAT field ([PRES_STATE\[CDIHB\]](#)) and the Command Inhibit CMD field ([PRES_STATE\[CIHB\]](#)) in the Present State register before writing to this register. When the CDIHB field in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB field is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as a single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC ignores the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active ([PRES_STATE\[WPSPL\]](#) field of Present State register is '1'); otherwise, uSDHC also ignores the command.

If the commands with data transfer do not receive the response in 64 clock cycles, that is, if response time-out happens, uSDHC treats the external device, does not accept the command, and aborts the data transfer. In this scenario, the driver should issue the command again to retry the transfer. It is also possible that for some reason the card responds to the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 511. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, regarding the Response Type bits as well as the name of the response type.

Table 512. Relationship between parameters and the name of the response type

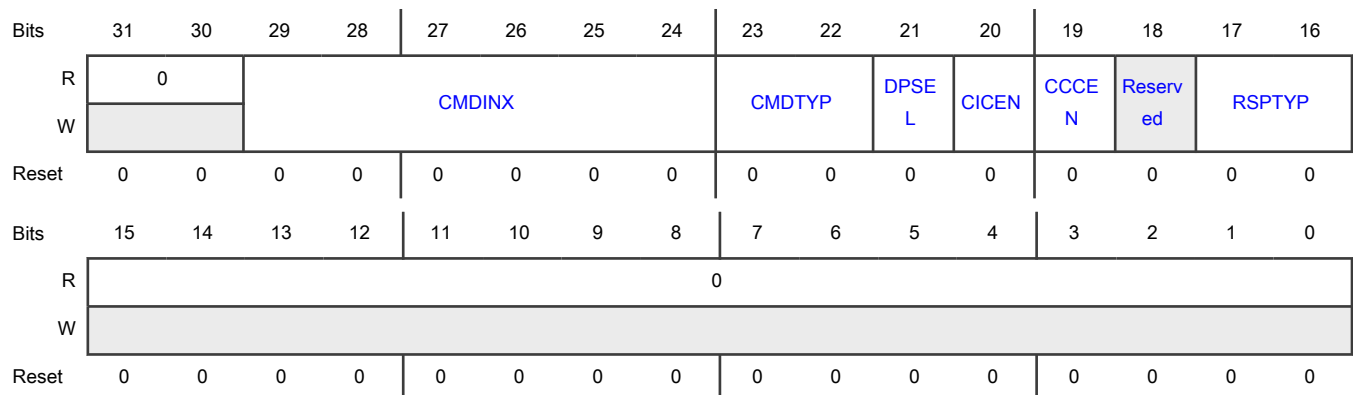
Response type	Index check enable	CRC check enable	Name of response type
00	0	0	No response

Table continues on the next page...

Table 512. Relationship between parameters and the name of the response type (continued)

Response type	Index check enable	CRC check enable	Name of response type
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification, but R5b is defined in this specification to specify that uSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command is used with R5b.
- The CRC fields for R3 and R4 are expected to be all 1 bits. The CRC check is disabled for these response types.

Diagram**Fields**

Field	Function
31-30 —	Reserved
29-24 CMDINX	Command index These bits are set to the command number that is specified in bits 45-40 of the command-format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23-22 CMDTYP	Command type There are three types of special commands: Suspend, Resume, and Abort. These bits are set to 00b for all other commands. • Suspend command: If the Suspend command succeeds, uSDHC assumes that the card bus has been released and that it is possible to issue the next command that uses the DATA line. Because uSDHC does not monitor the content of command response, it does not know if the Suspend

Table continues on the next page...

Table continued from the previous page...

Field	Function
	command succeeded or not. It is the host driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform uSDHC that a Suspend command was successfully issued. See Suspend Resume for more details. After the end bit of command is sent, uSDHC deasserts Read Wait for read transactions and stops checking busy for write transactions. In a 4-bit mode, the interrupt cycle starts. If the Suspend command fails, uSDHC maintains its current state, and the host driver restarts the transfer by setting the Continue Request field in the Protocol Control register. • Resume command: The host driver re-starts the data transfer by restoring the registers saved before sending the Suspend command and then sends the Resume command. The uSDHC module checks for a pending busy state before starting write transfers. • Abort command: If this command is set when executing a read transfer, uSDHC stops reads to the buffer. If this command is set when executing a write transfer, uSDHC stops driving the DATA line. After issuing the Abort command, the host driver should issue a software reset (Abort Transaction). 00b - Normal other commands 01b - Suspend CMD52 for writing bus suspend in CCCR 10b - Resume CMD52 for writing function select in CCCR 11b - Abort CMD12, CMD52 for writing I/O Abort in CCCR
21 DPSEL	Data present select This field is set to 1 to indicate that data is present and is transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52) • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b (for example, CMD38)) NOTE In resume command, this field is set, and other bits in this register is set the same as when the transfer was initially launched. When the write protect switch is on, (that is, the WPSPL field is active as '0'), any command with a write operation ignored. When this field is set, while the DTDSEL field is 0, writes to the register Transfer Type are ignored. 0b - No data present 1b - Data present
20 CICEN	Command index check enable If this field is set to 1, uSDHC checks the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this field is set to 0, the Index field is not checked. 0b - Disable command index check 1b - Enables command index check
19 CCEN	Command CRC check enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is set to 1, uSDHC checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this field is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. See RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP). 0b - Disables command CRC check 1b - Enables command CRC check
18 —	Reserved
17-16 RSPTYP	Response type select 00b - No response 01b - Response length 136 10b - Response length 48 11b - Response length 48, check busy after response
15-0 —	Reserved

50.9.2.6 Command Response0 (CMD_RSP0)

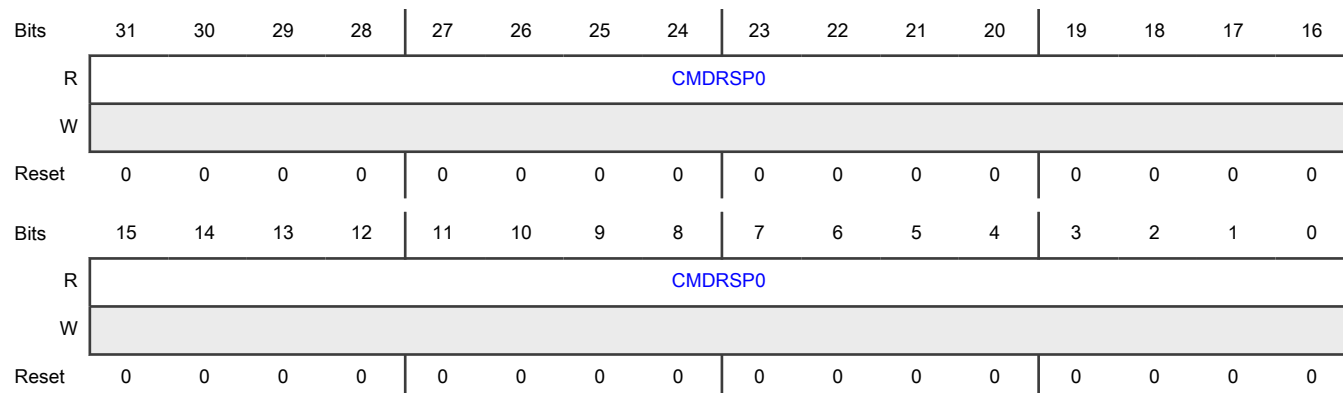
Offset

Register	Offset
CMD_RSP0	10h

Function

This register is used to store part 0 of the response bits from the card.

Diagram



Fields

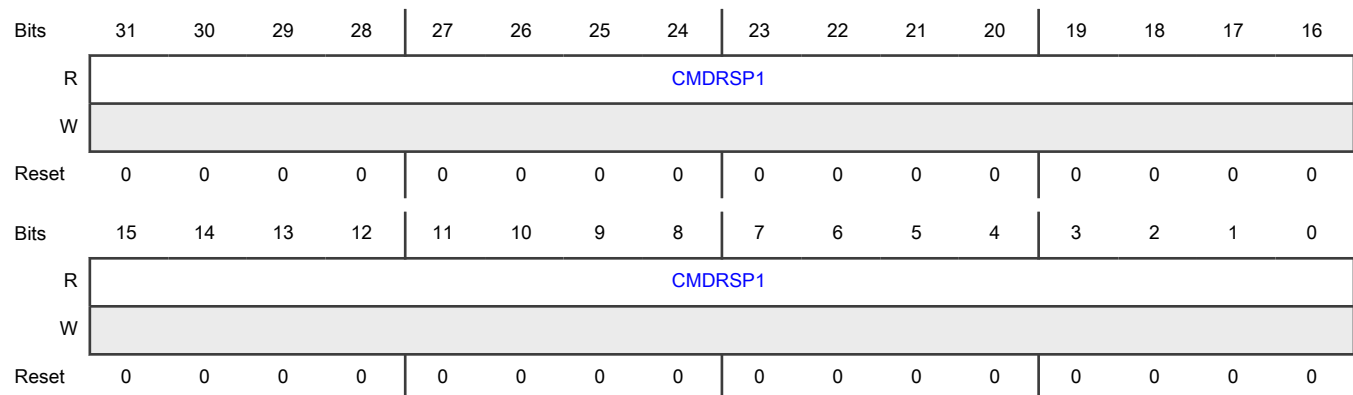
Field	Function
31-0 CMDRSP0	Command response 0 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.2.7 Command Response1 (CMD_RSP1)**Offset**

Register	Offset
CMD_RSP1	14h

Function

This register is used to store part 1 of the response bits from the card.

Diagram**Fields**

Field	Function
31-0 CMDRSP1	Command response 1 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

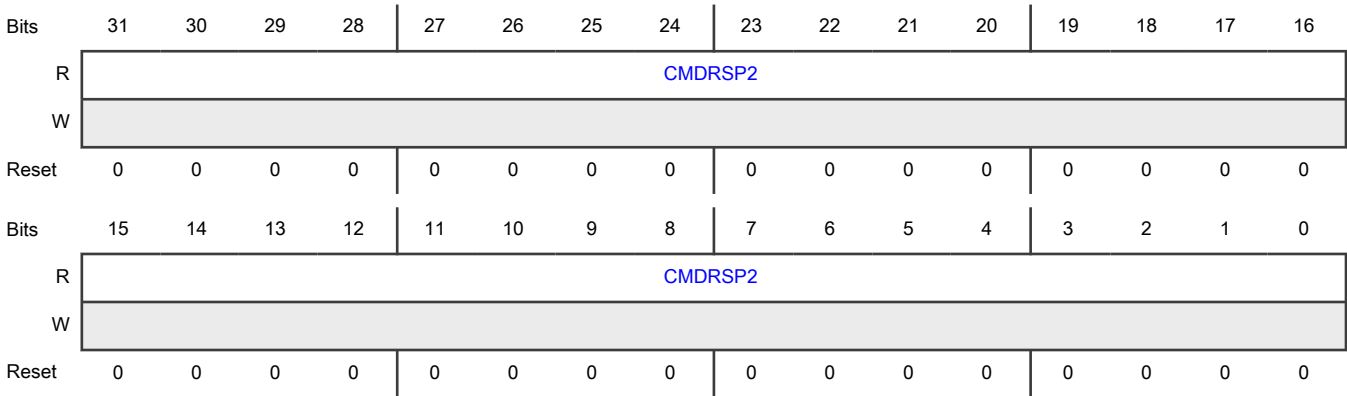
50.9.2.8 Command Response2 (CMD_RSP2)**Offset**

Register	Offset
CMD_RSP2	18h

Function

This register is used to store part 2 of the response bits from the card.

Diagram



Fields

Field	Function
31-0 CMDRSP2	Command response 2 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.2.9 Command Response3 (CMD_RSP3)

Offset

Register	Offset
CMD_RSP3	1Ch

Function

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD bus to Command Response registers for each response type. In this table, R[] refers to a bit range within the response data as transmitted on the SD bus.

Table 513. Response bit definition for each response type

Response type	Meaning of response	Response field	Response register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0
R1b (auto CMD12 response)	Card status for auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2,

Table continues on the next page...

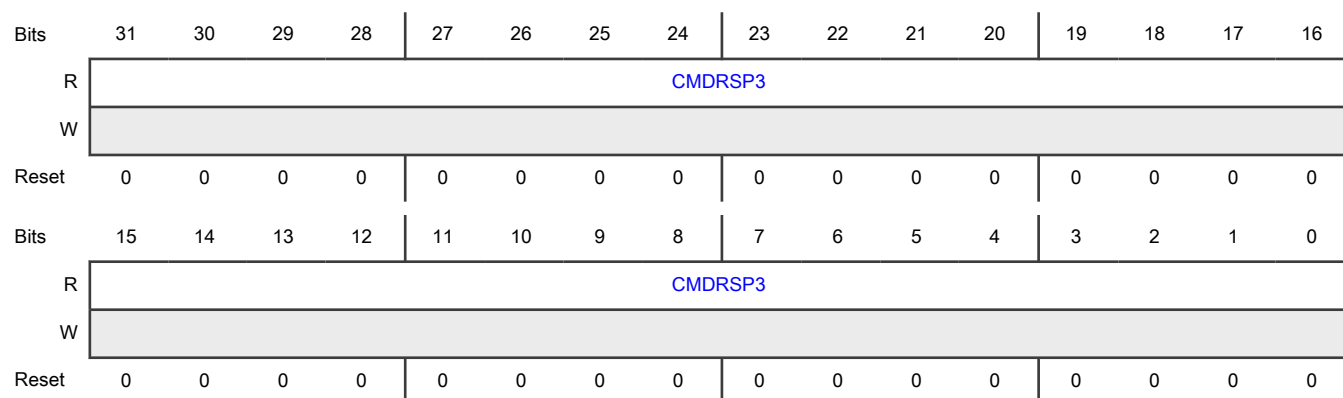
Table 513. Response bit definition for each response type (continued)

Response type	Meaning of response	Response field	Response register
			CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (publish RCA)	New published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, uSDHC only stores part of the response data in the Command Response registers. This enables the host driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, uSDHC checks R[47:1], and if the response length is 136 the uSDHC checks R[119:1].

Because uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Diagram**Fields**

Field	Function
31-0	Command response 3

Table continues on the next page...

Field	Function
CMDRSP3	See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this register for each response type.

50.9.2.10 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

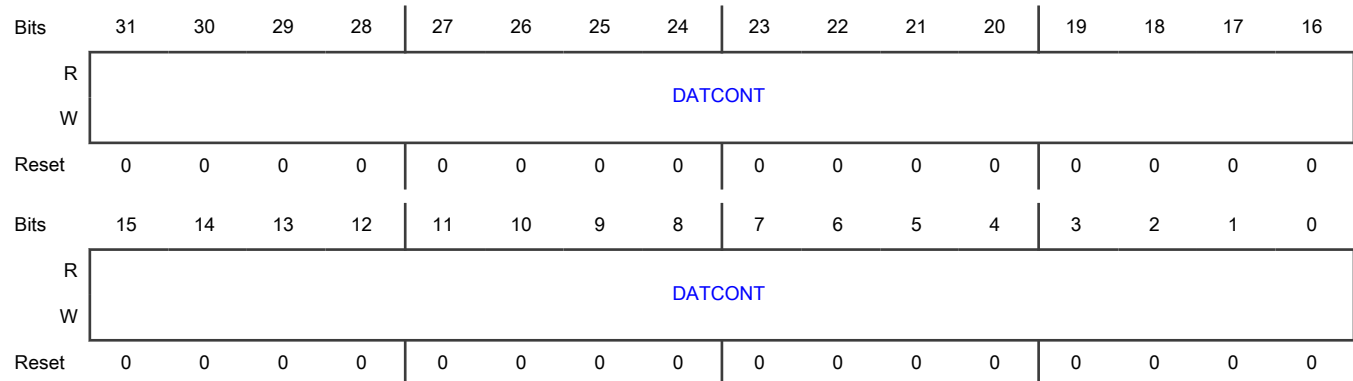
Offset

Register	Offset
DATA_BUFF_ACC_PORT	20h

Function

This is a 32-bit data port register used to access the internal buffer.

Diagram



Fields

Field	Function
31-0 DATCONT	Data content The Buffer Data Port register is for 32-bit data access by the ARM platform. When the internal DMA is enabled, any write to this register is ignored, and any read from this register always yields 0s.

50.9.2.11 Present State (PRES_STATE)

Offset

Register	Offset
PRES_STATE	24h

Function

The host driver can get status of uSDHC from this 32-bit read only register.

The host driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands are issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

NOTE

The reset value of Present State register depends on board connectivity.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DLSL								CLSL	0				WPSP L	CDPL	0	CINST
W																	
Reset	u	u	u	u	u	u	u	u	u	0	0	0	u	u	0	u	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	TSCD	0		RTR	BREN	BWEN	RTA	WTA	SDOF F	PERO FF	HCKO FF	IPGOF F	SDST B	DLA	CDIHB	CIHB	
W																	
Reset	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

Fields

Field	Function
31-24 DLSL	DATA[7:0] line signal level This status is used to check the DATA line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. 0000_0000b - Data 0 line signal level 0000_0001b - Data 1 line signal level 0000_0010b - Data 2 line signal level 0000_0011b - Data 3 line signal level 0000_0100b - Data 4 line signal level 0000_0101b - Data 5 line signal level 0000_0110b - Data 6 line signal level 0000_0111b - Data 7 line signal level
23 CLSL	CMD line signal level

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this field after reset is 1'b1, when the command line is pulled up.
22-20 —	Reserved
19 WPSPL	Write protect switch pin level The Write Protect switch is supported for memory and combo cards. This field reflects the inverted value of the WP pin of the card socket. A software reset does not affect this field. The reset value is affected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this field is high and write is enabled. 0b - Write protected (WP = 1) 1b - Write enabled (WP = 0)
18 CDPL	Card detect pin level This field reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this field. This field may be valid, but is not guaranteed, because of propagation delay. Use of this field is limited to testing because it must be debounced by software. A software reset does not affect this field. A write to the Force Event Register does not affect this field. The reset value is affected by the external card detection pin. This field shows the value on the CD_B pin (that is, when a card is inserted in the socket, it is 0 on the CD_B input, and consequently, the CDPL reads 1. 0b - No card present (CD_B = 1) 1b - Card present (CD_B = 0)
17 —	Reserved
16 CINST	Card inserted This field indicates whether a card has been inserted. The uSDHC module debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not affect this field. The Software Reset For All in the System Control register does not affect this field. A software reset does not affect this field. 0b - Power on reset or no card 1b - Card inserted
15 TSCD	Tape select change done This field indicates the delay setting is effective after write CLK_TUNE_CTRL_STATUS register. 0b - Delay cell select change is not finished. 1b - Delay cell select change is finished.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14-13 —	Reserved
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode and EMMC HS200 mode) Host controller may request host driver to execute re-tuning sequence by setting this field when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This field is cleared when a command is issued with setting Execute Tuning field in MIXER_CTRL register. Changing of this field from 0 to 1 generates Re-Tuning Event. See Interrupt status registers for more detail. This field isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock). 0b - Fixed or well tuned sampling clock 1b - Sampling clock needs re-tuning
11 BREN	Buffer read enable This status field is used for non-DMA read transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this field is high, valid data greater than the watermark level exist in the buffer. A change of this field from 1 to 0 occurs when some reads from the buffer (read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this field from 0 to 1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 0b - Read disable 1b - Read enable
10 BWEN	Buffer write enable This status field is used for non-DMA write transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this field is 1, valid data greater than the watermark level can be written to the buffer. A change of this field from 1 to 0 occurs when some writes to the buffer (write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this field from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 0b - Write disable 1b - Write enable
9 RTA	Read transfer active This status field is used for detecting completion of a read transfer. This field is set for either of the following conditions: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer

Table continues on the next page...

Table continued from the previous page...

Field	Function
	A transfer complete interrupt is generated when this field changes to 0. This field is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent because of the Stop At Block Gap Request being set to 1. 0b - No valid data 1b - Transferring data
8 WTA	Write transfer active This status field indicates a write transfer is active. If this field is 0, it means no valid write data exists in uSDHC. This field is set in either of the following cases: • After the end field of the write command • When writing 1 to the Continue Request field in the Protocol Control register to restart a write transfer This field is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request During a write transaction, a Block Gap Event interrupt is generated when this field is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the host driver in determining when to issue commands during Write Busy state. 0b - No valid data 1b - Transferring data
7 SDOFF	SD clock gated off internally This status field indicates that the SD clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON field is 0 to stop the SD clock in idle status. This field is for the host driver to debug data transaction on the SD bus. 0b - SD clock is active. 1b - SD clock is gated off.
6 PEROFF	IPG_PERCLK gated off internally This status field indicates that the IPG_PERCLK is internally gated off. This field is for the host driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK is gated off, otherwise IPG_PERCLK is always active. 0b - IPG_PERCLK is active. 1b - IPG_PERCLK is gated off.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 HCKOFF	HCLK gated off internally This status field indicates that the HCLK is internally gated off. This field is for the host driver to debug during a data transfer. 0b - HCLK is active. 1b - HCLK is gated off.
4 IPGOFF	Peripheral clock gated off internally This status field indicates that the peripheral clock is internally gated off. This field is for the host driver to debug. 0b - Peripheral clock is active. 1b - Peripheral clock is gated off.
3 SDSTB	SD clock stable This status field indicates that the internal card clock is stable. This field is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON field in System Control register to remove glitches on the card clock when the frequency is changing. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. 0b - Clock is changing frequency and not stable. 1b - Clock is stable.
2 DLA	Data line active This status field indicates whether one of the DATA lines on the SD bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer This field is cleared in either of the following cases: • When the end field of the last data block is sent from the SD bus to uSDHC. • When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC module waits at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait to use the suspend / resume function. This field remains 1 during Read Wait. In the case of write transactions:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This status indicates that a write transfer is executing on the SD bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the write command • When writing to 1 to the Continue Request field in the Protocol Control register to continue a write transfer This field is cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, uSDHC also detects if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, uSDHC assumes the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and because of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This field is cleared when the DATA0 line is released. 0b - DATA line inactive 1b - DATA line active
1 CDIHB	Command inhibit (DATA) This status field is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this field is 0, it indicates that uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. NOTE The SD host driver can save registers for a suspend transaction after this field has changed from 1 to 0. 0b - Can issue command that uses the DATA line 1b - Cannot issue command that uses the DATA line
0 CIHB	Command inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and uSDHC can issue a SD / MMC command using the CMD line. This field is set also immediately after the Transfer Type register is written. This field is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, commands using only the CMD line can be issued if this field is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If uSDHC cannot issue the command because of a command conflict error (see Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this field remains 1 and the Command Complete is not set. The Status of issuing an auto CMD12 does not show on this field. 0b - Can issue command using only CMD line 1b - Cannot issue command

50.9.2.12 Protocol Control (PROT_CTRL)

Offset

Register	Offset
PROT_CTRL	28h

Function

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether uSDHC issues a Suspend command or the SD card accepts the Suspend command.

- If the host driver does not issue a Suspend command, the Continue request is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card accepts it, a Resume command is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card does not accept it, the Continue request is used to restart the transfer.

Any time stop at block gap request stops the data transfer, the host driver waits for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the host driver clears the Stop At Block Gap Request before or simultaneously.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	NON_	BURST_LEN_EN			WECR	WEI	WEI	Reserved				RD_D	IABG	RWCT	SABG
W	ed	EXA...				M	NS	NT					ONE...	L	CREQ	REQ
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W					DMASEL				CDSS	CDTL	EMODE		D3CD	DTW		Reserv
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Fields

Field	Function
31 —	Reserved Always write as 0
30 NON_EXACT_B LK_RD	Non-exact block read Current block read is non-exact block read. It is only used for SDIO. 0b - The block read is exact block read. Host driver does not need to issue abort command to terminate this multi-block read.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read.
29-27 BURST_LEN_EN	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLES on the AXI side. 1xxb - Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP. x1xb - Burst length is enabled for INCR4 / INCR8 / INCR16. xx1b - Burst length is enabled for INCR.
26 WECRM	Wakeup event enable on SD card removal This field enables a wakeup event, via a card removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this field. When this field is set, the Card Removal Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Removal Status and uSDHC interrupt. 0b - Disables wakeup event enable on SD card removal 1b - Enables wakeup event enable on SD card removal
25 WECINS	Wakeup event enable on SD card insertion This field enables a wakeup event, via a card insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this field. When this field is set, the Card Insertion Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Insertion Status and uSDHC interrupt. 0b - Disable wakeup event enable on SD card insertion 1b - Enable wakeup event enable on SD card insertion
24 WECINT	Wakeup event enable on card interrupt This field enables a wakeup event, via a card interrupt, in the Interrupt Status register. This field can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this field is set, the Card Interrupt Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Interrupt Status and uSDHC interrupt. 0b - Disables wakeup event enable on card interrupt 1b - Enables wakeup event enable on card interrupt
23-21 —	Reserved Always write as 3'b100
20 RD_DONE_NO_8CLK	Read performed number 8 clock

Table continues on the next page...

Table continued from the previous page...

Field	Function
	According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end field of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks are active after the end field of the last read data transaction. However, these 8 clocks should not be active if user wants to use stop at block gap (include the auto stop at block gap in boot mode) feature for read and the RWCTL field (bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid these 8 clocks. Otherwise, the device might send extra data to uSDHC while uSDHC ignores these data. In a summary, this field should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.
19 IABG	Interrupt at block gap This field is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this field should be set to 0 to avoid an inadvertent interrupt. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. 0b - Disables interrupt at block gap 1b - Enables interrupt at block gap
18 RWCTL	Read wait control The read wait function is optional for SDIO cards. If the card supports read wait, set this field to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise, uSDHC has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. If the card does not support read wait, this field should never be set to 1; otherwise, DATA line conflicts might occur. If this field is set to 0, stop at block gap during read operation is also supported, but uSDHC stops the SD clock to pause reading operation. 0b - Disables read wait control and stop SD clock at block gap when SABGREQ field is set 1b - Enables read wait control and assert read wait without stopping SD clock at block gap when SABGREQ field is set
17 CREQ	Continue request This field is used to restart a transaction which was stopped using the stop at block gap request. When a suspend operation is not accepted by the card, it is also by setting this field to restart the paused transfer. To cancel stop at the block gap, set stop at block gap request to 0 and set this field to 1 to restart the transfer. The uSDHC module automatically clears this field, therefore it is not necessary for the host driver to set this field to 0. If both stop at block gap request and this field are 1, the continue request is ignored. 0b - No effect 1b - Restart
16 SABGREQ	Stop at block gap request This field is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion, the host driver leaves this field set to 1. Clearing both the stop at block gap request and continue request does not cause the transaction to

Table continues on the next page...

Table continued from the previous page...

Field	Function
	restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC module supports the stop at block gap request for write transfers, but for read transfers it requires that the SDIO card support read wait. Therefore, the host driver does not set this field during read transfers unless the SDIO card supports Read Wait and has set the read wait control to 1; otherwise, uSDHC stops the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the host driver writes data to the Data Port register, the host driver sets this field after all block data is written. If this field is set to 1, the host driver does not write data to the Data Port register after a block is sent. Once this field is set, the host driver does not clear this field before the Transfer Complete field in Interrupt Status register is set, otherwise uSDHC's behavior is undefined. This field effects read transfer active, write transfer active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 0b - Transfer 1b - Stop
15-10 —	Reserved
9-8 DMASEL	DMA select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00b - No DMA or simple DMA is selected. 01b - ADMA1 is selected. 10b - ADMA2 is selected. 11b - Reserved
7 CDSS	Card detect signal selection This field selects the source for the card detection. 0b - Card detection level is selected (for normal purpose). 1b - Card detection test level is selected (for test purpose).
6 CDTL	Card detect test level This bit is enabled while the card detection signal selection is set to 1 and it indicates card insertion. 0b - Card detect test level is 0, no card inserted 1b - Card detect test level is 1, card inserted
5-4 EMODE	Endian mode The uSDHC module supports all three endian modes in data transfer. See Data buffer for more details. 00b - Big endian mode 01b - Half word big endian mode 10b - Little endian mode 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 D3CD	DATA3 as card detection pin If this field is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 might set the card into the SPI mode, which uSDHC does not support. 0b - DATA3 does not monitor card insertion 1b - DATA3 as card detection pin
2-1 DTW	Data transfer width This field selects the data width of the SD bus for a data transfer. The host driver sets it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits. 00b - 1-bit mode 01b - 4-bit mode 10b - 8-bit mode 11b - Reserved
0 —	Reserved

50.9.2.13 System Control (SYS_CTRL)

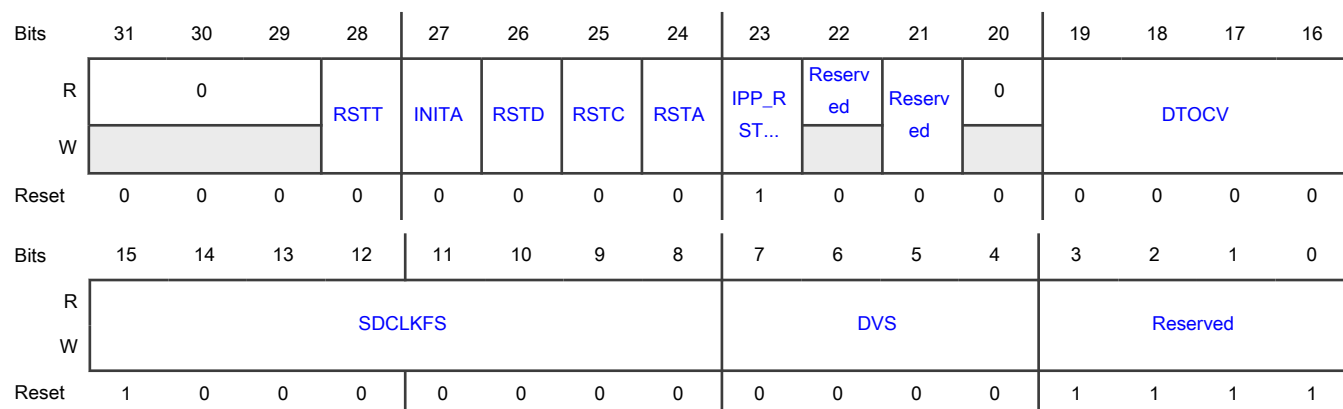
Offset

Register	Offset
SYS_CTRL	2Ch

Function

This register provides control of the system. See detail in the field description.

Diagram



Fields

Field	Function
31-29 —	Reserved
28 RSTT	Reset tuning When set this field to 1, it resets tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning field in AUTOCMD12_ERR_STATUS also sets this field to 1 to reset tuning circuit
27 INITA	Initialization active When this field is set, 80 SD-clocks are sent to the card. After the 80 clocks are sent, this field is self cleared. This field is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this field is already 1 has no effect. Writing 0 to this field at any time has no effect. When either of the CIHB and CDIHB fields in the Present State register are set, writing 1 to this field is ignored (that is, when command line or data lines are active, write to this field is not allowed). On the other-hand, when this field is set, that is, during initialization active period, it is allowed to issue command, and the command bit stream appears on the CMD pad after all 80 clock cycles are done. So, when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs to send 80 cycles to the card and does not want to wait till this field is self cleared.
26 RSTD	Software reset for data line Only part of the data circuit is reset. DMA circuit is also reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Data Port register • Buffer is cleared and initialized • Present State register • Buffer read enable • Buffer write enable • Read transfer active • Write transfer active • DATA line active • Command Inhibit (DATA) Protocol Control register • Continue request • Stop At Block Gap Request Interrupt Status register • Buffer read ready • Buffer write ready • DMA interrupt • Block gap event • Transfer complete

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
25 RSTC	Software reset for CMD line Only part of the command circuit is reset. After this field is set, the software waits for self-clear. The following registers and bits are cleared by this field: • Present State Register Command Inhibit (CMD) • Interrupt Status register Command Complete 0b - No reset 1b - Reset
24 RSTA	Software reset for all This reset effects the entire host controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the host driver is set this field to 1 to reset uSDHC. The uSDHC module resets this field to 0 when the capabilities registers are valid and the host driver can read them. Additional use of Software Reset For All does not affect the value of the capabilities registers. After this field is set, it is recommended that the host driver reset the external card and re-initialize it. After this field is set, the software should wait for self-clear. In tuning process, after every CMD19 is finished, this field is set to retest uSDHC. NOTE When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
23 IPP_RST_N	Hardware reset This register's value is output to card through pad directly to hardware reset pin of the card if the card supports this feature.
22 —	Reserved
21 —	Reserved
20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-16 DTCV	Data timeout counter value This value determines the interval by which DAT line timeouts are detected. See the Data Timeout Error field in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency is generated by dividing the base clock SDCLK value by this value. The host driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 0000b - $\text{SDCLK} \times 2^{14}$ 0001b - $\text{SDCLK} \times 2^{15}$ 0010b - $\text{SDCLK} \times 2^{16}$ 0011b - $\text{SDCLK} \times 2^{17}$ 0100b - $\text{SDCLK} \times 2^{18}$ 0101b - $\text{SDCLK} \times 2^{19}$ 0110b - $\text{SDCLK} \times 2^{20}$ 0111b - $\text{SDCLK} \times 2^{21}$ 1000b - $\text{SDCLK} \times 2^{22}$ 1001b - $\text{SDCLK} \times 2^{23}$ 1010b - $\text{SDCLK} \times 2^{24}$ 1011b - $\text{SDCLK} \times 2^{25}$ 1100b - $\text{SDCLK} \times 2^{26}$ 1101b - $\text{SDCLK} \times 2^{27}$ 1110b - $\text{SDCLK} \times 2^{28}$ 1111b - $\text{SDCLK} \times 2^{29} + \text{SDCLK} \times 2^{28} + \text{SDCLK} \times 2^{27} + \text{SDCLK} \times 2^{26}$
15-8 SDCLKFS	SDCLK frequency select This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. In Single Data Rate mode (DDR_EN field of MIXERCTRL is '0') Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01h) Base clock divided by 2 00h) Base clock divided by 1 While in Dual Data Rate mode (DDR_EN field of MIXERCTRL is '1') Only the following settings are allowed: 80h) Base clock divided by 512 40h) Base clock divided by 256 20h) Base clock divided by 128 10h) Base clock divided by 64 08h) Base clock divided by 32 04h) Base clock divided by 16 02h) Base clock divided by 8 01h) Base clock divided by 4 00h) Base clock divided by 2 When the software changes the DDR_EN field, SDCLKFS might need to be changed also. In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h yields 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD clock frequency is 50 MHz and is never exceed this limit. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. If setting SDCLKFS and DVS can generate the same clock frequency,(for example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.
7-4	Divisor
DVS	This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. Before changing clock divisor value (SDCLKFS or DVS), Host driver should make sure the SDSTB field is high.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The settings are as follows: 0000b - Divide-by-1 0001b - Divide-by-2 1110b - Divide-by-15 1111b - Divide-by-16
3-0	Reserved
—	Always write as 1.

50.9.2.14 Interrupt Status (INT_STATUS)

Offset

Register	Offset
INT_STATUS	30h

Function

An interrupt is generated when the normal interrupt signal enable is enabled and at least one of the status fields is set to 1. For all fields, writing 1 to a bit clears it; writing 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For card interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the card driver services the interrupt condition; otherwise, the CINT field is asserted again.

The table below shows the relationship between the command timeout error and the command complete.

Table 514. uSDHC status for command timeout error/command complete bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the transfer complete and the data timeout error.

Table 515. uSDHC status for data timeout error/transfer complete bit combinations

Transfer complete	Data timeout error	Meaning of the status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data transfer complete

The table below shows the relationship between the command CRC error and command timeout error.

Table 516. uSDHC status for command CRC error/command timeout error bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				W1C		W1C		W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TP	0	RTE	0			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		W1C		W1C				W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 DMAE	DMA error Occurs when an Internal DMA transfer has failed. This field is set to 1, when some error occurs in the data transfer. This error can be caused by either simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Because any error corrupts the whole data block, the host driver restarts the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 0b - No error 1b - Error
27 —	Reserved
26	Tuning error: (only for SD3.0 SDR104 mode and EMMC HS200 mode)

Table continues on the next page...

Table continued from the previous page...

Field	Function
TNE	This field is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, host driver needs to abort a command executing and perform tuning.
25 —	Reserved
24 AC12E	Auto CMD12 error Occurs when detecting that one of the fields in the Auto CMD12 Error Status register has changed from 0 to 1. This field is set to 1, not only when the errors in Auto CMD12 occur, but also, when the Auto CMD12 is not executed due to the previous command error. 0b - No error 1b - Error
23 —	Reserved
22 DEBE	Data end bit error Occurs either when detecting 0 at the end field position of read data that uses the DATA line, or at the end field position of the CRC. This field is not asserted in tuning process. 0b - No error 1b - Error
21 DCE	Data CRC error Occurs when detecting a CRC error when transferring read data that uses the DATA line, or when detecting the Write CRC status having a value other than 010. This field is not asserted in tuning process. 0b - No error 1b - Error
20 DIOE	Data timeout error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This field is not asserted in tuning process. 0b - No error 1b - Time out
19	Command index error

Table continues on the next page...

Table continued from the previous page...

Field	Function
CIE	Occurs if a command index error occurs in the command response. This field is not asserted in tuning process. 0b - No error 1b - Error
18 CEBE	Command end bit error Occurs when detecting that the end field of a command response is 0. This field is not asserted in tuning process. 0b - No error 1b - End bit error generated
17 CCE	Command CRC error Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this field is set when detecting a CRC error in the command response. • The uSDHC module detects a CMD line conflict by monitoring the CMD line when a command is issued. If uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then uSDHC aborts the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error should also be set to 1 to distinguish CMD line conflict. This field is not asserted in tuning process. 0b - No error 1b - CRC error generated
16 CTOE	Command timeout error Occurs only if no response is returned within 64 SDCLK cycles from the end field of the command. If uSDHC detects a CMD line conflict, in which case a Command CRC Error is also be set (as shown in Interrupt Status (INT_STATUS)), this field is set without waiting for 64 SDCLK cycles. This is because the command is aborted by uSDHC. This field is not asserted in tuning process. 0b - No error 1b - Time out
15 —	Reserved
14 TP	Tuning pass:(only for SD3.0 SDR104 mode and EMMC HS200 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 RTE	Re-tuning event: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. host controller requests host driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 0b - Re-tuning is not required. 1b - Re-tuning should be performed.
11-9 —	Reserved
8 CINT	Card interrupt This status field is set when an interrupt signal is detected from the external card. In 1-bit mode, uSDHC detects the Card Interrupt without the SD clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the host system. Writing this field to 1 can clear this field, but as the interrupt source from the SDIO card does not clear, this field is set again. to clear this field, it is required to reset the interrupt source from the external card followed by a writing 1 to this field. When this status has been set, and the host driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the host system. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal might not be asserted), write 1 to clear this field, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 0b - No card interrupt 1b - Generate card interrupt
7 CRM	Card removal This status field is set if the Card Inserted field in the Present State register changes from 1 to 0. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if no card is inserted. to leave it cleared, clear the Card Removal Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or inserted 1b - Card removed
6 CINS	Card insertion This status field is set if the Card Inserted field in the Present State register changes from 0 to 1. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if a card is inserted. to leave it cleared, clear the Card Inserted Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or removed

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Card inserted
5 BRR	Buffer read ready This status field is set if the Buffer Read Enable field, in the Present State register, changes from 0 to 1. See the Buffer Read Enable field in the Present State register for additional information. This field indicates that cmd19 is finished in tuning process. 0b - Not ready to read buffer 1b - Ready to read buffer
4 BWR	Buffer write ready This status field is set if the Buffer Write Enable field, in the Present State register, changes from 0 to 1. See the Buffer Write Enable field in the Present State register for additional information. 0b - Not ready to write buffer 1b - Ready to write buffer
3 DINT	DMA interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this field does not be set. Instead, the DMAE field is set. Either Simple DMA or ADMA finishes data transferring, this field is set. 0b - No DMA interrupt 1b - DMA interrupt is generated.
2 BGE	Block gap event If the Stop At Block Gap Request field in the Protocol Control register is set, this field is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this field is not set to 1. In the case of a Read Transaction: This field is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD bus timing). The Read Wait must be supported to use this function. In the case of Write Transaction: This field is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 0b - No block gap event 1b - Transaction stopped at block gap
1 TC	Transfer complete This field is set when a read or write transfer is completed. In the case of a Read Transaction: This field is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the host system). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request field in the Protocol Control register (after valid data has been read to the host system).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	In the case of a Write Transaction: This field is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request field in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this field is set when busy is deasserted. This field is not asserted in tuning process. 0b - Transfer does not complete 1b - Transfer complete
0 CC	Command complete This field is set when you receive the end field of the command response (except auto CMD12). See the Command Inhibit (CMD) in the Present State register. This field is not asserted in tuning process. 0b - Command not complete 1b - Command complete

50.9.2.15 Interrupt Status Enable (INT_STATUS_EN)

Offset

Register	Offset
INT_STATUS_EN	34h

Function

Setting the fields in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any field is cleared, the corresponding Interrupt Status field is also cleared (that is, when the field in this register is cleared, the corresponding field in Interrupt Status register is always 0).

- Depending on IABG field setting, uSDHC might be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There are some delays on the Card Interrupt, asserted from the card, to the time the host system is informed.
- To detect a CMD line conflict, the host driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE SEN	0	TNES EN	0	AC12E SEN	0	DEBE SEN	DCES EN	DTOE SEN	CIESE N	CEBE SEN	CCES EN	CTOE SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPSE N	0	RTES EN	0			CINTS EN	CRMS EN	CINSS EN	BRRS EN	BWRS EN	DINTS EN	BGES EN	TCSE N	CCSE N
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 DMAESEN	DMA error status enable 0b - Masked 1b - Enabled
27 —	Reserved
26 TNESEN	Tuning error status enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12ESEN	Auto CMD12 error status enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBESEN	Data end bit error status enable 0b - Masked 1b - Enabled
21	Data CRC error status enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
DCESEN	0b - Masked 1b - Enabled
20 DTESEN	Data timeout error status enable 0b - Masked 1b - Enabled
19 CIESEN	Command index error status enable 0b - Masked 1b - Enabled
18 CEBESEN	Command end bit error status enable 0b - Masked 1b - Enabled
17 CCESEN	Command CRC error status enable 0b - Masked 1b - Enabled
16 CTOSEN	Command timeout error status enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPSEN	Tuning pass status enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTESEN	Re-tuning event status enable 0b - Masked 1b - Enabled
11-9 —	Reserved
8 CINTSEN	Card interrupt status enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If this field is set to 0, uSDHC clears the interrupt request to the system. The Card Interrupt detection is stopped when this field is cleared and restarted when this field is set to 1. The host driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this field again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 0b - Masked 1b - Enabled
7 CRMSEN	Card removal status enable 0b - Masked 1b - Enabled
6 CINSEN	Card insertion status enable 0b - Masked 1b - Enabled
5 BRRSEN	Buffer read ready status enable 0b - Masked 1b - Enabled
4 BWRSEN	Buffer write ready status enable 0b - Masked 1b - Enabled
3 DINTSEN	DMA interrupt status enable 0b - Masked 1b - Enabled
2 BGESEN	Block gap event status enable 0b - Masked 1b - Enabled
1 TCSEN	Transfer complete status enable 0b - Masked 1b - Enabled
0 CCSEN	Command complete status enable 0b - Masked 1b - Enabled

50.9.2.16 Interrupt Signal Enable (INT_SIGNAL_EN)

Offset

Register	Offset
INT_SIGNAL_EN	38h

Function

This register is used to select which interrupt status is indicated to the host system as the interrupt. These status fields all share the same interrupt line. Setting any of these fields to 1 enables interrupt generation. The corresponding Status register field generates an interrupt when the corresponding interrupt signal enable field is set.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0	TNEIE	0	AC12E	0	DEBEI	DCEIE	DTOEI	CIEIE	CEBEI	CCEIE	CTOEI
W						N		IEN		EN	N	EN	N	EN	N	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPIEN	0	RTEIE	0				CRMI	CINSI	BRRIE	BWRI	DINTI	BGEIE	TCIEN	CCIEN
W				N					EN	EN	N	EN	EN	N		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 DMAEIEN	DMA error interrupt enable 0b - Masked 1b - Enable
27 —	Reserved
26 TNEIEN	Tuning error interrupt enable 0b - Masked 1b - Enabled
25 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 AC12EIEN	Auto CMD12 error interrupt enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBEIEN	Data end bit error interrupt enable 0b - Masked 1b - Enabled
21 DCEIEN	Data CRC error interrupt enable 0b - Masked 1b - Enabled
20 DTOEIEN	Data timeout error interrupt enable 0b - Masked 1b - Enabled
19 CIEIEN	Command index error interrupt enable 0b - Masked 1b - Enabled
18 CEBEIEN	Command end bit error interrupt enable 0b - Masked 1b - Enabled
17 CCEIEN	Command CRC error interrupt enable 0b - Masked 1b - Enabled
16 CTOEIEN	Command timeout error interrupt enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPIEN	Tuning Pass interrupt enable 0b - Masked 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 —	Reserved
12 RTEIEN	Re-tuning event interrupt enable 0b - Masked 1b - Enabled
11-9 —	Reserved
8 CINTIEN	Card interrupt enable 0b - Masked 1b - Enabled
7 CRMIEN	Card removal interrupt enable 0b - Masked 1b - Enabled
6 CINSIEN	Card insertion interrupt enable 0b - Masked 1b - Enabled
5 BRRIEN	Buffer read ready interrupt enable 0b - Masked 1b - Enabled
4 BWRIEN	Buffer write ready interrupt enable 0b - Masked 1b - Enabled
3 DINTIEN	DMA interrupt enable 0b - Masked 1b - Enabled
2 BGEIEN	Block gap event interrupt enable 0b - Masked 1b - Enabled
1 TCIEN	Transfer complete interrupt enable 0b - Masked 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 CCIEN	Command complete interrupt enable 0b - Masked 1b - Enabled

50.9.2.17 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

Offset

Register	Offset
AUTOCMD12_ERR_STATUS	3Ch

Function

When the Auto CMD12 Error Status field in the Status register is set, the host driver checks this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in field 04-01. This register is valid only when the Auto CMD12 Error status field is set.

The table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 517. Relationship between command CRC error and command timeout error for auto CMD12

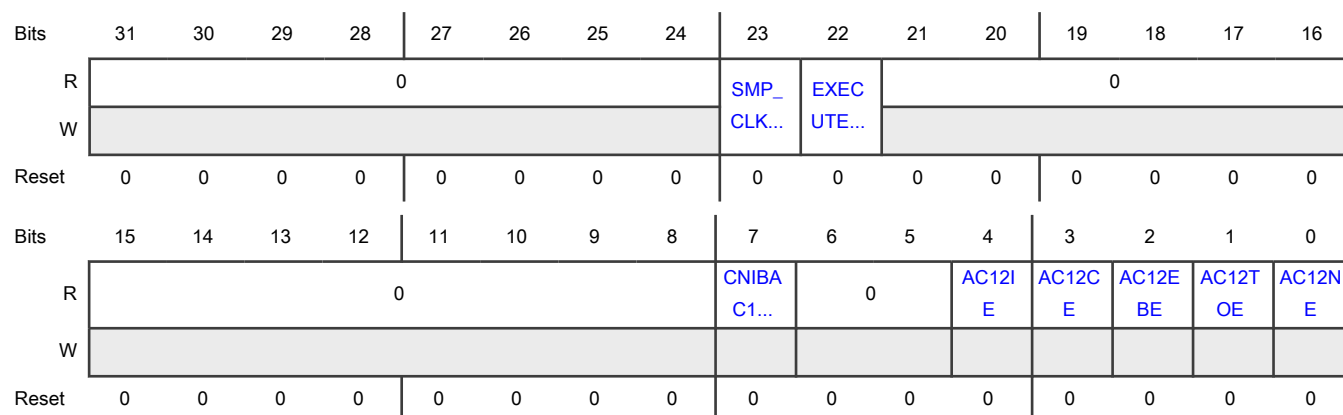
Auto CMD12 CRC error	Auto CMD12 timeout error	Type of error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

- When uSDHC is going to issue an Auto CMD12
 - Set field 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
 - Set field 0 to 0 if the Auto CMD12 is issued
- At the end field of an Auto CMD12 response
 - Check errors correspond to fields 1-4
 - Set fields 1-4 corresponding to detected errors
 - Clear fields 1-4 corresponding to detected errors
- Before reading the Auto CMD12 Error Status field 7
 - Set field 7 to 1 if there is a command that can't be issued
 - Clear field 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, field 7 is sampled when the driver is not writing to the Command register. So, it is suggested to read this register only when the AC12E field in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error fields (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Diagram



Fields

Field	Function
31-24 —	Reserved
23 SMP_CLK_SEL	Sample clock select When std_tuning_en field is set, this field is used to select sampling clock to receive CMD and DATA. Otherwise, this field is reserved. This field is set by ty tuning procedure and valid after the completion of tuning(When Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Writing 1 to this field is meaningless and ignored. A tuning circuit is reset by writing to 0. This field can be cleared with setting Execute Tuning. Once the tuning circuit is reset, it takes time to complete tuning sequence. Therefore, host driver should keep this field to 1 to perform re-tuning sequence to complete re-tuning sequence in a short time. Change of this field is not allowed while the Host controller us receiving response or a read data block. 0b - Fixed clock is used to sample data 1b - Tuned clock is used to sample data
22 EXECUTE_TUNING	Execute tuning When std_tuning_en field is set, this field is used to start tuning procedure. Otherwise, this field is reserved. This field is set to start tuning procedure and automatically cleared when runing procedure is completed. The result of tuning is indicated to sam_clk_sel field. Tuning procedure is aborted by writing 0.
21-8 —	Reserved
7 CNIBAC12E	Command not issued by Auto CMD12 error

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Setting this field to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04-D01) in this register. 0b - No error 1b - Not issued
6-5 —	Reserved
4 AC12IE	Auto CMD12 / 23 index error Occurs if the command index error occurs in response to a command. 0b - No error 1b - Error, the CMD index in response is not CMD12/23
3 AC12CE	Auto CMD12 / 23 CRC error Occurs when detecting a CRC error in the command response. 0b - No CRC error 1b - CRC error met in Auto CMD12/23 response
2 AC12EBE	Auto CMD12 / 23 end bit error Occurs when detecting that the end field of command response is 0 which should be 1. 0b - No error 1b - End bit error generated
1 AC12TOE	Auto CMD12 / 23 timeout error Occurs if no response is returned within 64 SDCLK cycles from the end field of the command. If this field is set to 1, the other error status fields (2-4) have no meaning. 0b - No error 1b - Time out
0 AC12NE	Auto CMD12 not executed If memory multiple block data transfer is not started, due to a command error, this field is not set because it is not necessary to issue an Auto CMD12. Setting this field to 1 means uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this field is set to 1, other error status fields (1-4) have no meaning. 0b - Executed 1b - Not executed

50.9.2.18 Host Controller Capabilities (HOST_CTRL_CAP)

Offset

Register	Offset
HOST_CTRL_CAP	40h

Function

This register provides the host driver with information specific to uSDHC implementation. The value in this register is the power-on-reset value and does not change with a software reset.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMA S	0	MBL		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RETUNING_M ODE		USE_T UN...	Reserv ed	TIME_COUNT_RETUNING				Reserved				DDR5 0_S...		SDR10 4_...	SDR50 _S...
W																
Reset	1	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1

Fields

Field	Function
31-27 —	Reserved
26 VS18	Voltage support 1.8 V This field depends on the host system ability. 0b - 1.8 V not supported 1b - 1.8 V supported
25 VS30	Voltage support 3.0 V This field depends on the host system ability. 0b - 3.0 V not supported 1b - 3.0 V supported
24 VS33	Voltage support 3.3 V This field depends on the host system ability.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - 3.3 V not supported 1b - 3.3 V supported
23 SRS	Suspend / resume support This field indicates whether uSDHC supports Suspend / Resume functionality. If this field is 0, the Suspend and Resume mechanism, as well as the read wait, are not supported, and the host driver does not issue either Suspend or Resume commands. 0b - Not supported 1b - Supported
22 DMAS	DMA support This field indicates whether uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 0b - DMA not supported 1b - DMA supported
21 HSS	High speed support This field indicates whether uSDHC supports High Speed mode and the host system can supply a SD clock frequency from 25 MHz to 50 MHz. 0b - High speed not supported 1b - High speed supported
20 ADMAS	ADMA support This field indicates whether uSDHC supports the ADMA feature. 0b - Advanced DMA not supported 1b - Advanced DMA supported
19 —	Reserved
18-16 MBL	Max block length This value indicates the maximum block size that the host driver can read and write to the buffer in uSDHC. The buffer transfers block size without wait cycles. 000b - 512 bytes 001b - 1024 bytes 010b - 2048 bytes 011b - 4096 bytes
15-14	Retuning Mode This field selects retuning method.

Table continues on the next page...

Table continued from the previous page...

Field	Function
RETUNING_MODE	00b - Mode 1 01b - Mode 2 10b - Mode 3 11b - Reserved
13 USE_TUNING_SDR50	Use Tuning for SDR50 This field is set to 1. Host controller requires tuning to operate SDR50. 0b - SDR does not require tuning. 1b - SDR50 requires tuning.
12 —	Reserved
11-8 TIME_COUNT_RETUNING	Time counter for retuning This field indicates an initial value of the Retuning Timer for Re-Tuning Mode1 and 3. Setting to 0 disables Retuning Timer.
7-3 —	Reserved
2 DDR50_SUPPORT	DDR50 support This field indicates support of DDR50 mode.
1 SDR104_SUPPORT	SDR104 support This field indicates support of SDR104 mode.
0 SDR50_SUPPORT	SDR50 support This field indicates support of SDR50 mode.

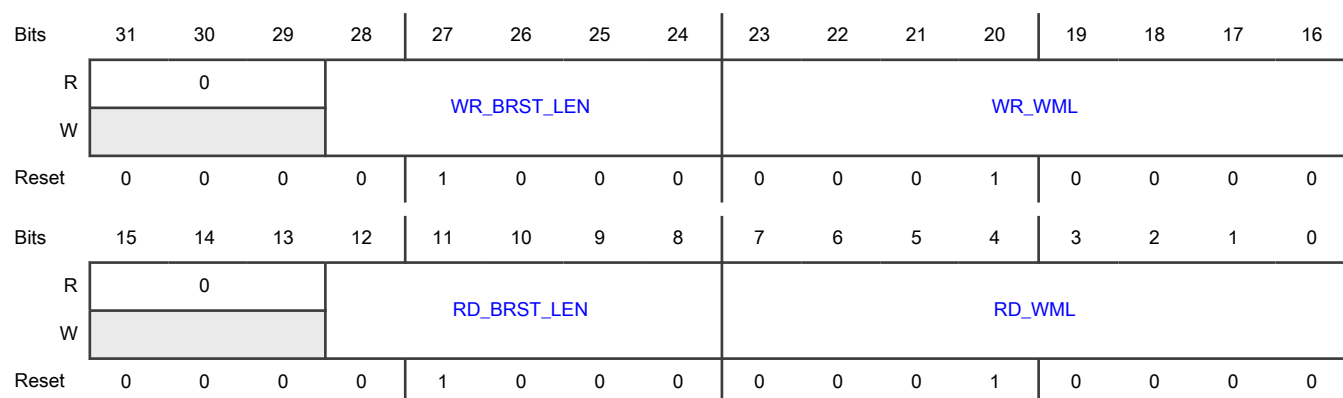
50.9.2.19 Watermark Level (WTMK_LVL)

Offset

Register	Offset
WTMK_LVL	44h

Function

Both write and read watermark levels (FIFO threshold) are configurable. Their value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. Their value can range from 1 to 31 words.

Diagram**Fields**

Field	Function
31-29 —	Reserved
28-24 WR_BRST_LEN	Write burst length due to system restriction, the actual burst length might not exceed 16 The number of words uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
23-16 WR_WML	Write watermark level The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also, the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15-13 —	Reserved
12-8 RD_BRST_LEN	Read burst length due to system restriction, the actual burst length might not exceed 16 The number of words uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
7-0 RD_WML	Read watermark level The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also, the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

50.9.2.20 Mixer Control (MIX_CTRL)

Offset

Register	Offset
MIX_CTRL	48h

Function

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 518. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed	Reserv ed	Reserv ed	0	0	0	FBCLK _S...	AUTO _TU...	SMP_ CLK...	EXE_ TUNE	0					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AC23E N	NIBBL E_...	MSBS EL	DTDS EL	DDR_ EN	AC12E N	BCEN	DMAE N
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Reserved
—	Always write as 1.
30	Reserved
—	Always write as 0.
29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Always write as 0.
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 FBCLK_SEL	Feedback clock source selection (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Feedback clock comes from the loopback CLK 1b - Feedback clock comes from the ipp_card_clk_out
24 AUTO_TUNE_EN	Auto tuning enable (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Disable auto tuning 1b - Enable auto tuning
23 SMP_CLK_SEL	Clock selection When STD_TUNING_EN is 0, this field is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) 0b - Fixed clock is used to sample data / cmd 1b - Tuned clock is used to sample data / cmd
22 EXE_TUNE	Execute tuning: (Only used for SD3.0, SDR104 mode and EMMC HS200 mode) When STD_TUNING_EN is 0, this field is set to 1 to indicate the host driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 0b - Not tuned or tuning completed 1b - Execute tuning
21-8 —	Reserved
7 AC23EN	Auto CMD23 enable When this field is set to 1, the host controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	Nibble position indication In DDR 4-bit mode nibble position indication. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 MSBSEL	Multi / Single block select This field enables multiple block DATA line data transfers. For any other commands, this field can be set to 0. If this field is 0, it is not necessary to set the Block Count register. See Command Transfer Type (CMD_XFR_TYP). 0b - Single block 1b - Multiple blocks
4 DTDSEL	Data transfer direction select This field defines the direction of DATA line data transfers. The field is set to 1 by the host driver to transfer data from the SD card to uSDHC and is set to 0 for all other commands. 0b - Write (Host to card) 1b - Read (Card to host)
3 DDR_EN	Dual data rate mode selection
2 AC12EN	Auto CMD12 enable Multiple block transfers for memory require a CMD12 to stop the transaction. When this field is set to 1, uSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver is not set this field to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, uSDHC ignores this field no matter it is set or not. 0b - Disable 1b - Enable
1 BCEN	Block count enable This field is used to enable the Block Count register, which is only relevant for multiple block transfers. When this field is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 0b - Disable 1b - Enable
0 DMAEN	DMA enable This field enables DMA functionality. If this field is set to 1, a DMA operation begins when the host driver sets the DPSEL field of this register. Whether the simple DMA or the advanced DMA is active depends on the DMA Select field of the Protocol Control register. 0b - Disable 1b - Enable

50.9.2.21 Force Event (FORCE_EVENT)

Offset

Register	Offset
FORCE_EVENT	50h

Function

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status register can be written if the corresponding field of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding field of Interrupt Status register. A read from this register always results in 0's. to change the corresponding status fields in the Interrupt Status register, make sure to set IPGEN field in System Control Register so that peripheral clock is always active.

Forcing a card interrupt generates a short pulse on the DATA1 line, and the driver might treat this interrupt as a normal interrupt. The interrupt service routine might skip polling the card interrupt factor as the interrupt is self cleared.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	FEVT CINT			FEVT DMAE		FEVT NE		FEVT AC1...		FEVT DEBE	FEVT DCE	FEVT DCE	FEVT CIE	FEVT CEBE	FEVT CCE	FEVT CTOE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	0	0	0	0	0	0	0
W									FEVT CNI...			FEVT C1...	FEVT C1...	FEVT C1...	FEVT C1...	FEVT C1...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 FEVTCINT	Force event card interrupt Writing 1 to this field generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT field is set and the interrupt service routine might treat this interrupt as a normal interrupt from the external card.
30-29 —	Reserved
28 FEVTDMAE	Force event DMA error Forces the DMAE field of Interrupt Status register to be set.
27	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
26 FEVTTNE	Force tuning error Forces the TNE field of Interrupt Status register to be set.
25 —	Reserved
24 FEVTAC12E	Force event Auto Command 12 error Forces the AC12E field of Interrupt Status register to be set.
23 —	Reserved
22 FEVTDEBE	Force event data end bit error Forces the DEBE field of Interrupt Status register to be set.
21 FEVTDCE	Force event data CRC error Forces the DCE field of Interrupt Status register to be set.
20 FEVTDTOE	Force event data time out error Force the DTOE field of Interrupt Status register to be set.
19 FEVTCIE	Force event command index error Forces the CCE field of Interrupt Status register to be set.
18 FEVTCBE	Force event command end bit error Forces the CEBE field of Interrupt Status register to be set.
17 FEVTCCE	Force event command CRC error Forces the CCE field of Interrupt Status register to be set.
16 FEVTCCTOE	Force event command time out error Forces the CTOE field of Interrupt Status register to be set.
15-8 —	Reserved
7 FEVTCNIBAC12E	Force event command not executed by Auto Command 12 error Forces the CNIBAC12E field in the Auto Command12 Error Status register to be set.
6-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
4 FEVTAC12IE	Force event Auto Command 12 index error Forces the AC12IE field in the Auto Command12 Error Status register to be set.
3 FEVTAC12EBE	Force event Auto Command 12 end bit error Forces the AC12EBE field in the Auto Command12 Error Status register to be set.
2 FEVTAC12CE	Force event auto command 12 CRC error Forces the AC12CE field in the Auto Command12 Error Status register to be set.
1 FEVTAC12TOE	Force event auto command 12 time out error Forces the AC12TOE field in the Auto Command12 Error Status register to be set.
0 FEVTAC12NE	Force event auto command 12 not executed Forces the AC12NE field in the Auto Command12 Error Status register to be set.

50.9.2.22 ADMA Error Status (ADMA_ERR_STATUS)

Offset

Register	Offset
ADMA_ERR_STATUS	54h

Function

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the host driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and does not generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

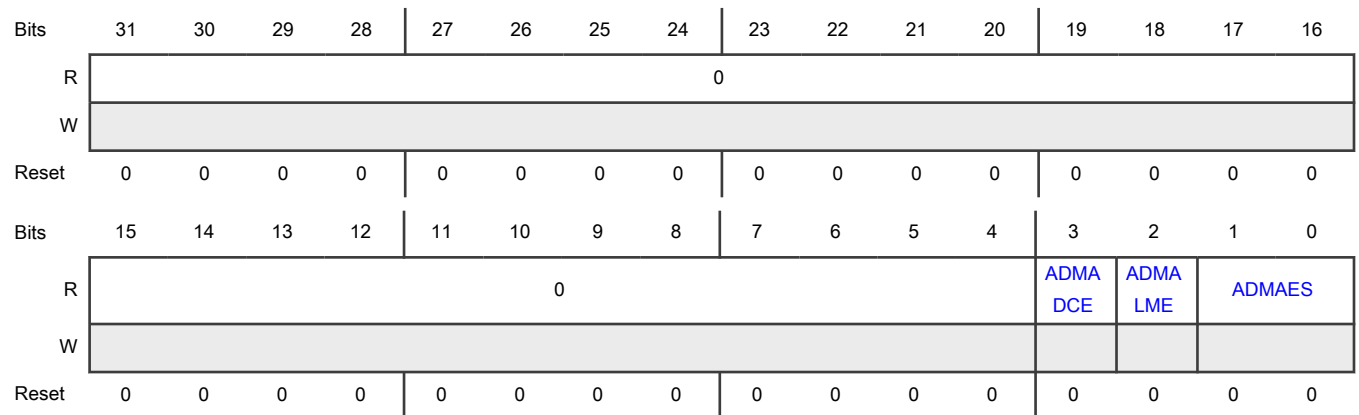
In case of a write operation, the host driver should use the ACMD22 to get the number of the written block, rather than using this information, because unwritten data might exist in the host controller.

The host controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The host driver can distinguish this error by reading the Valid field of the error descriptor.

Table 519. ADMA error state coding

D01-D00	ADMA error state (when error has occurred)	Contents of ADMA System Address register
00	ST_STOP (Stop DMA)	Holds the address of the next executable descriptor command
01	ST_FDS (Fetch descriptor)	Holds the valid descriptor address
10	ST_CADR (Change address)	No ADMA error is generated
11	ST_TFR (Transfer data)	Holds the address of the next executable descriptor command

Diagram



Fields

Field	Function
31-4 —	Reserved
3 ADMADCE	ADMA descriptor error This error occurs when invalid descriptor fetched by ADMA. 0b - No error 1b - Error
2 ADMALME	ADMA length mismatch error This error occurs in the following two cases: - While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length. - Total data length cannot be divided by the block length.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No error 1b - Error
1-0 ADMAES	ADMA error state (when ADMA error is occurred) This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. See ADMA Error Status (ADMA_ERR_STATUS) for more details.

50.9.2.23 ADMA System Address (ADMA_SYS_ADDR)

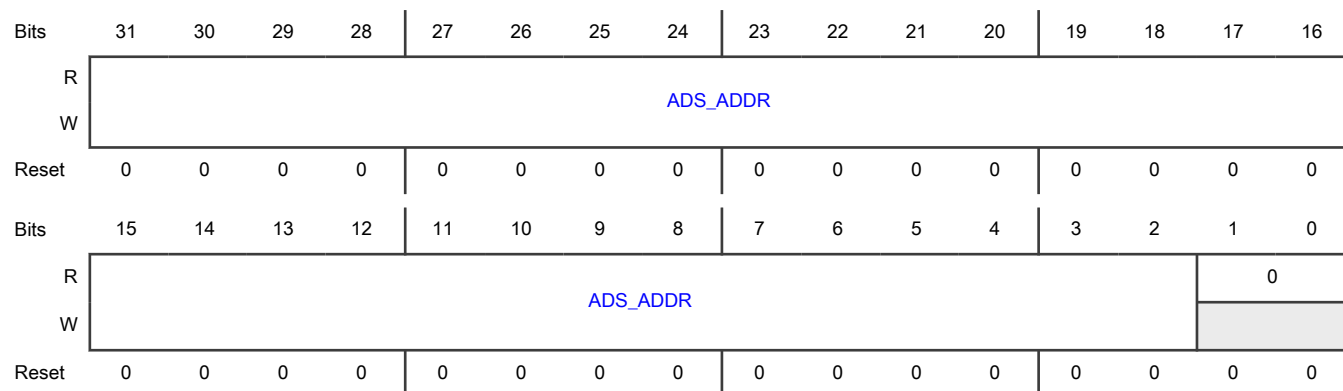
Offset

Register	Offset
ADMA_SYS_ADDR	58h

Function

This register contains the physical system memory address used for ADMA transfers.

Diagram



Fields

Field	Function
31-2 ADS_ADDR	ADMA system address This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the host driver sets the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions .
1-0 —	Reserved

50.9.2.24 DLL (Delay Line) Control (DLL_CTRL)

Offset

Register	Offset
DLL_CTRL	60h

Function

This register contains control fields for DLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLL_CTRL_REF_UPDATE_INT				DLL_CTRL_SLV_UPDATE_INT								0	DLL_CTRL_SLV_DLY_T ARGET1		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_CTRL_SLV_OVERRIDE_VAL								DLL_C TR...	DLL_C TR...	DLL_CTRL_SLV_DLY_TARGET0			DLL_C TR...	DLL_C TR...	DLL_C TR...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 DLL_CTRL_REF_UPDATE_INT	DLL control loop update interval The interval cycle is $(2 + \text{REF\_UPDATE\_INT}) * \text{REF\_CLOCK}$. By default, the DLL control loop updates every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that might effect the delay (such as voltage and temperature)
27-20	Slave delay line update interval

Table continues on the next page...

Table continued from the previous page...

Field	Function
DLL_CTRL_SLV_UPDATE_INT	If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line also updates automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 —	Reserved
18-16 DLL_CTRL_SLV_DLY_TARGET1	DLL slave delay target1 See DLL_CTRL_SLV_DLY_TARGET0 below.
15-9 DLL_CTRL_SLV_OVERRIDE_VAL	DLL slave override val When SLV_OVERRIDE = 1, this field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.
8 DLL_CTRL_SLV_OVERRIDE	DLL slave override Set this field to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE field. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7 DLL_CTRL_GATE_UPDATE	DLL gate update Set this field to 1 to prevent the DLL from updating (because when clock_in exists, glitches might appear during DLL updates). This field might be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6-3 DLL_CTRL_SLV_DLY_TARGET0	DLL slave delay target0 The delay target for uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is $((\{DLL\_CTRL\_SLV\_DLY\_TARGET1, DLL\_CTRL\_SLV\_DLY\_TARGET0\} + 1) * REF\_CLOCK / 2) / 16$ So the input read-clock can be delayed relative input data from $(REF\_CLOCK / 2) / 16$ to $REF\_CLOCK * 4$.
2 DLL_CTRL_SLV_FORCE_UPDATE	DLL slave delay line Setting this field to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line updates automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function might not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_RESET	DLL reset Setting this field to 1 force a reset on DLL. This causes the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so to create a subsequent reset, RESET must be taken low and then asserted again.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0	DLL and delay chain
DLL_CTRL_ENABLE	Set this field to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

50.9.2.25 DLL Status (DLL_STATUS)

Offset

Register	Offset
DLL_STATUS	64h

Function

This register contains the DLL status information. All fields are read only and reads the same as the power-reset value.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_STS_REF_SEL								DLL_STS_SLV_SEL						DLL_S TS...	DLL_S TS...
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15-9 DLL_STS_REF_SEL	Reference delay line select taps This is encoded by 7 fields for 127 taps.
8-2	Slave delay line select status

Table continues on the next page...

Table continued from the previous page...

Field	Function
DLL_STS_SLV_SEL	This is the instant value generated from reference chain. Because the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

50.9.2.26 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)

Offset

Register	Offset
CLK_TUNE_CTRL_STATUS	68h

Function

This register contains the Clock Tuning Control status information. All fields are read only and reads the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode and EMMC HS200 mode.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE								TAP_SEL_OUT				TAP_SEL_POST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE								DLY_CELL_SET_OUT				DLY_CELL_SET_POST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PRE_ERR	PRE error PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
30-24 TAP_SEL_PRE	TAP_SEL_PRE Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE is updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23-20 TAP_SEL_OUT	Delay cells added on the feedback clock between CLK_PRE and CLK_OUT Reflects the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19-16 TAP_SEL_POST	Delay cells added on the feedback clock between CLK_OUT and CLK_POST Reflects the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14-8 DLY_CELL_SET_PRE	delay cells on the feedback clock between the feedback clock and CLK_PRE Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7-4 DLY_CELL_SET_OUT	Delay cells on the feedback clock between CLK_PRE and CLK_OUT Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
3-0 DLY_CELL_SET_POST	Delay cells on the feedback clock between CLK_OUT and CLK_POST Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

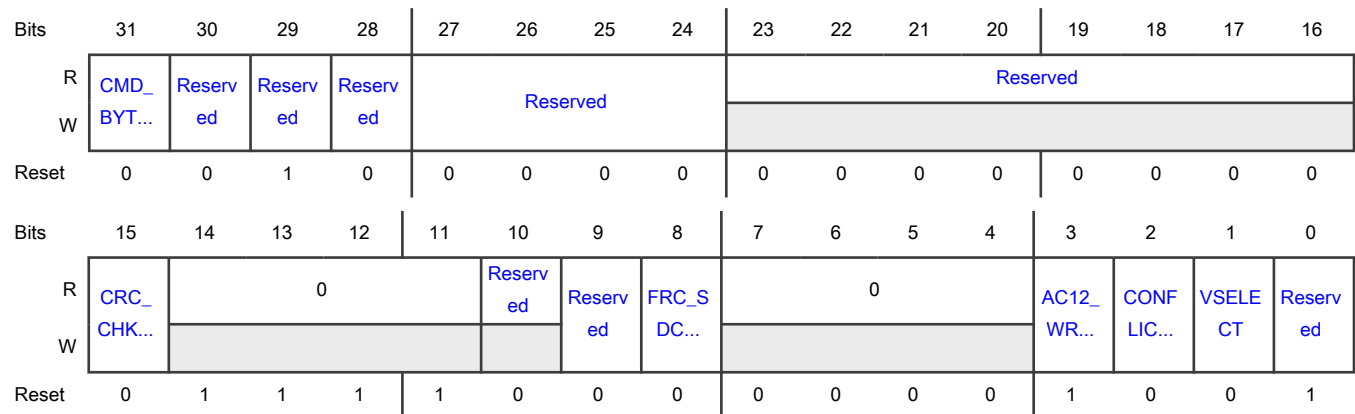
50.9.2.27 Vendor Specific Register (VEND_SPEC)**Offset**

Register	Offset
VEND_SPEC	C0h

Function

This register contains the vendor specific control / status register.

Diagram



Fields

Field	Function
31 CMD_BYTE_EN	Byte access This bit controls the byte access. 0b - Disable 1b - Enable
30 —	Reserved Always write as 0.
29 —	Reserved Always write as 1
28 —	Reserved Always write as 0.
27-24 —	Reserved Always write as 4'b0000.
23-16 —	Reserved Always write as 8'h00.
15 CRC_CHK_DIS	CRC Check Disable 0b - Check CRC16 for every read data packet and check CRC fields for every write data packet 1b - Ignore CRC16 check for every read data packet and ignore CRC fields check for every write data packet
14-11 —	Reserved Reserved
10	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Always write as 0.
9	Reserved
—	Always write as 0.
8	Force CLK
FRC_SDCLK_ON	Force CLK output active 0b - CLK active or inactive is fully controlled by the hardware. 1b - Force CLK active
7-4	Reserved
—	
3	Check busy enable
AC12_WR_CHK_BUSY_EN	Check busy enable after auto CMD12 for write data packet 0b - Do not check busy after auto CMD12 for write data packet 1b - Check busy after auto CMD12 for write data packet
2	Conflict check enable
CONFLICT_CHK_EN	It is not implemented in uSDHC IP. 0b - Conflict check disable 1b - Conflict check enable
1	Voltage selection
VSELECT	Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 0b - Change the voltage to high voltage range, around 3.0 V 1b - Change the voltage to low voltage range, around 1.8 V
0	Reserved
—	Always write as 0.

50.9.2.28 MMC Boot (MMC_BOOT)

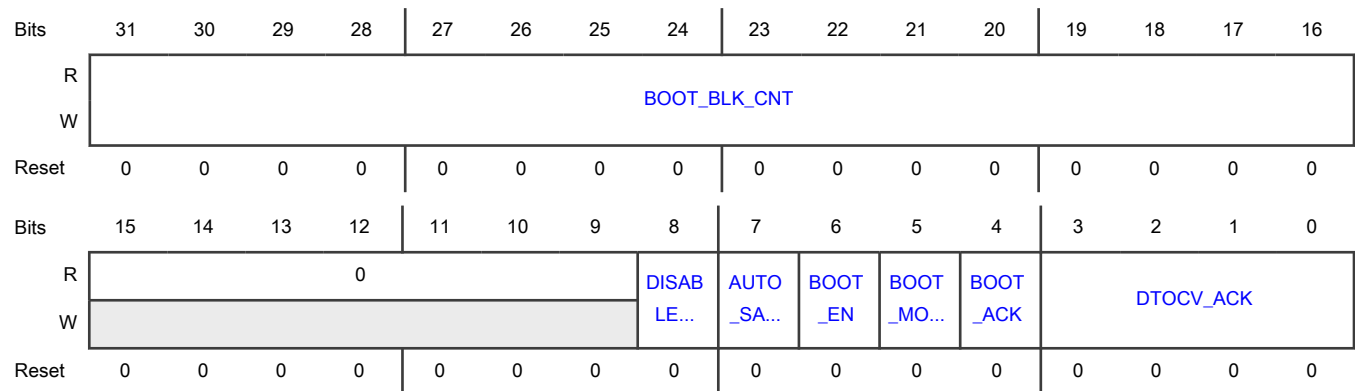
Offset

Register	Offset
MMC_BOOT	C4h

Function

This register contains the MMC Fast Boot control register.

Diagram



Fields

Field	Function
31-16 BOOT_BLK_CNT	Stop At Block Gap value of automatic mode The value defines the Stop At Block Gap value of automatic mode. When received, card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Attributes Register, field 31 - 16 of 0x04.
15-9 —	Reserved
8 DISABLE_TIME_OUT	Time out NOTE When this field is set, there is no timeout check no matter whether BOOT_EN is set or not. 0b - Enable time out 1b - Disable time out
7 AUTO_SABG_EN	Auto stop at block gap During boot, enable auto stop at block gap function. This function is triggered, and host stops at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot enable Boot mode enable 0b - Fast boot disable 1b - Fast boot enable
5 BOOT_MODE	Boot mode Boot mode select 0b - Normal boot 1b - Alternative boot

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 BOOT_ACK	BOOT ACK Boot ACK mode select 0b - No ack 1b - Ack
3-0 DTCV_ACK	Boot ACK time out Boot ACK time out counter value. 0000b - SDCLK x 2 ¹⁴ 0001b - SDCLK x 2 ¹⁵ 0010b - SDCLK x 2 ¹⁶ 0011b - SDCLK x 2 ¹⁷ 0100b - SDCLK x 2 ¹⁸ 0101b - SDCLK x 2 ¹⁹ 0110b - SDCLK x 2 ²⁰ 0111b - SDCLK x 2 ²¹ 1110b - SDCLK x 2 ²⁸ 1111b - SDCLK x 2 ²⁹

50.9.2.29 Vendor Specific 2 Register (VEND_SPEC2)

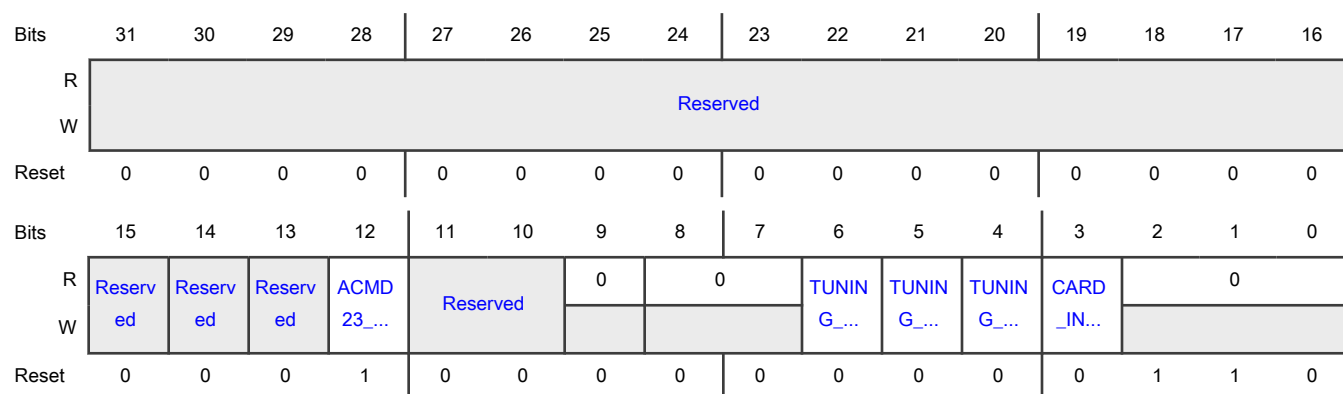
Offset

Register	Offset
VEND_SPEC2	C8h

Function

This register contains the vendor specific control 2 register.

Diagram



Fields

Field	Function
31-15 —	Reserved
14 —	Reserved
13 —	Reserved
12 ACMD23_ARG U2_EN	Argument2 register enable for ACMD23 0b - Disable 1b - Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enabled.
11-10 —	Reserved
9 —	Reserved
8-7 —	Reserved
6 TUNING_CMD_ EN	Tuning command enable Enable the auto tuning circuit to check the CMD line. 0b - Auto tuning circuit does not check the CMD line. 1b - Auto tuning circuit checks the CMD line.
5	Tuning 1bit enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TUNING_1bit_EN	Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.
4 TUNING_8bit_EN	Tuning 8bit enable Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. 0b00 - Tuning circuit only checks the DATA[3:0] 0b01 - Tuning circuit only checks the DATA0 0b10 - Tuning circuit checks the whole DATA[7:0] 0b11 - Invalid NOTE The format of these two fields are [TUNNING_8bit_EN:TUNNING_1bit_EN].
3 CARD_INT_D3_TEST	Card interrupt detection test This field only uses for debugging. 0b - Check the card interrupt only when DATA3 is high. 1b - Check the card interrupt by ignoring the status of DATA3.
2-0 —	Reserved

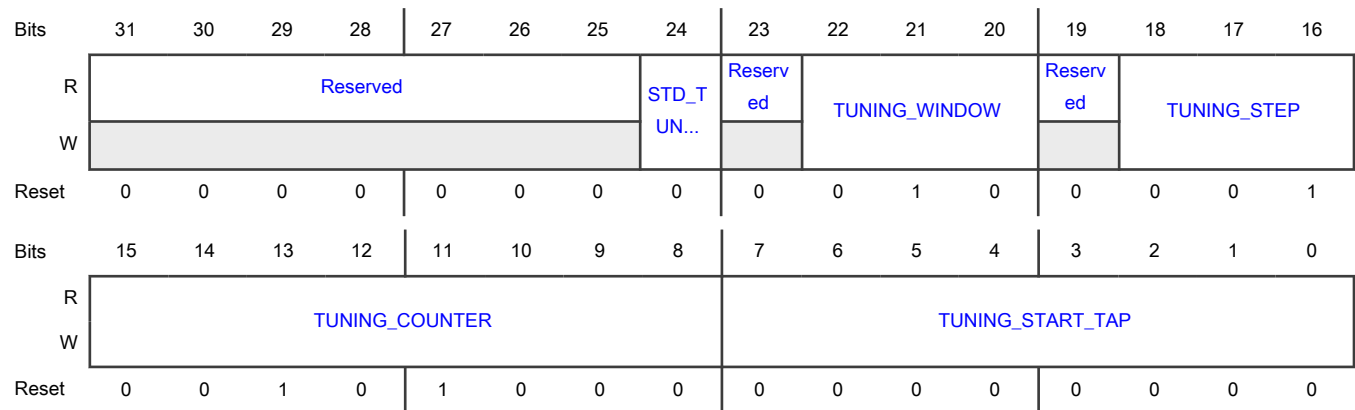
50.9.2.30 Tuning Control (TUNING_CTRL)

Offset

Register	Offset
TUNING_CTRL	CCh

Function

The register contains configuration of tuning circuit.

Diagram**Fields**

Field	Function
31-25 —	Reserved
24 STD_TUNING_EN	Standard tuning circuit and procedure enable This field is used to enable standard tuning circuit and procedure.
23 —	Reserved
22-20 TUNING_WINDOW	Data window Select data window value for auto tuning
19 —	Reserved
18-16 TUNING_STEP	TUNING_STEP The increasing delay cell steps in tuning procedure.
15-8 TUNING_COUNTER	Tuning counter The MAX repeat CMD19 times in tuning procedure.
7-0 TUNING_START_TAP	Tuning start The start delay cell point when send first CMD19 in tuning procedure.

Chapter 51

FlexIO

51.1 Chip-specific FLEXIO information

Table 520. Reference links to related information

Topic	Related module	Reference
Full description	FLEXIO	FLEXIO
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

51.1.1 Low Frequency Display Clock

The FLEXIO interface includes a low frequency clock-out (LOW_FREQ_CLKOUT) and its compliment (LOW_FREQ_CLKOUT_N). The low frequency clock is created by routing the 1 KHz output from the 32 KHz clock module through a standard, 8-bit clock divider. This provides a clock in the range of 4Hz to 1 KHz. The two clock pins are multiplexed with two of the sixteen FLEXIO pins.

NOTE

The 1 KHz clock output must be enabled by setting RTC1KHZ_EN bit in the RTC Control register in order to utilize the low frequency clock-out feature.

51.1.2 DMA Support

On this device, the standard DMA is not supported for use with the FLEXIO module. The SmartDMA should be used instead. Any reference to the standard DMA should be disregarded in the chapter.

51.1.3 Module instances

This device has one instance of the FLEXIO module, FLEXIO0.

51.2 Overview

The FlexIO is a highly configurable module providing a wide range of functionality including:

- Emulation of a variety of serial/parallel communication protocols
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions
- Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules
- Programmable state machine for offloading basic system control functions from CPU

51.2.1 Block Diagram

The [following diagram](#) gives a high-level overview of the configuration of FlexIO timers and shifters.

The FlexIO uses shifters, timers and external triggers to shift data into or out of the FlexIO. The timing of the data shift is controlled by the timers, as shown in the block diagram. The timers can be configured to use generic timer functions, external triggers, or various other conditions to determine this control.

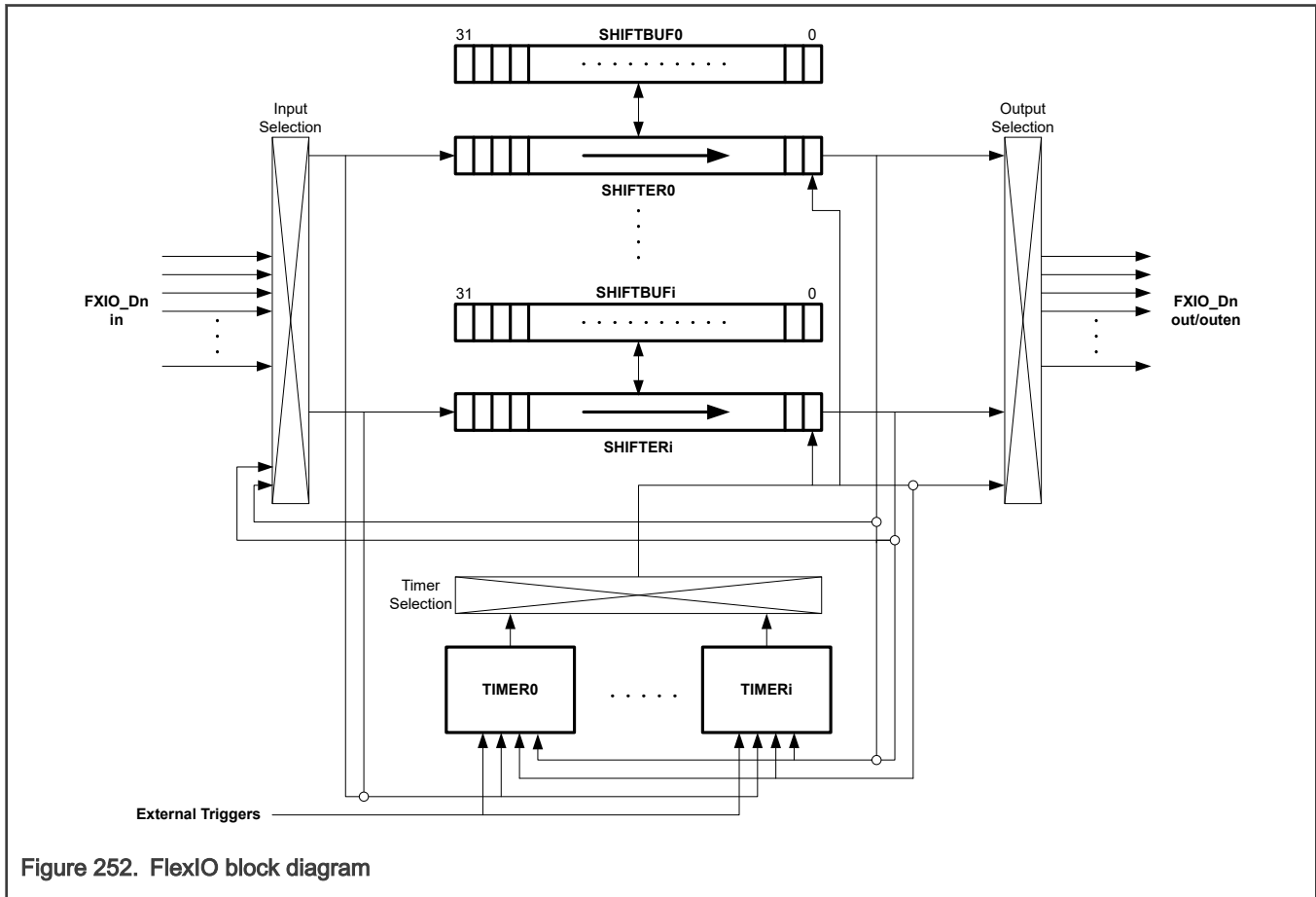


Figure 252. FlexIO block diagram

51.2.2 Features

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI
- I2S
- Camera IF
- Motorola 68K/Intel 8080 bus
- PWM/Waveform generation
- Input-capture (pulse edge interval measurement) - Example: SENT

The following key features are provided:

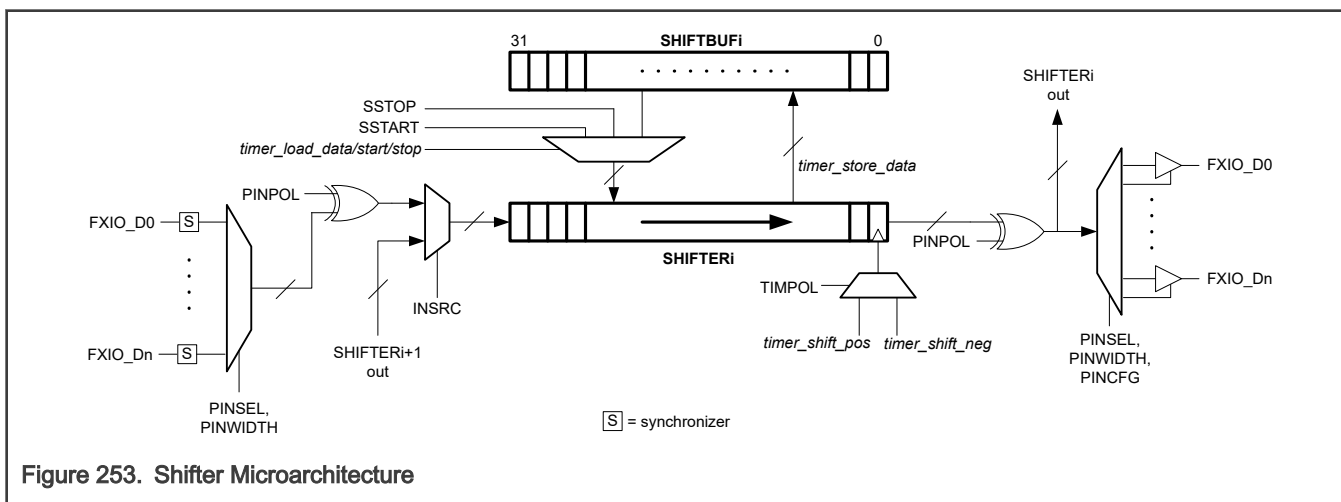
- Array of 32-bit shift registers with transmit, receive, data match, logic mode, and state modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation

- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support
- Interrupt, DMA or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions
- Programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs
- Programmable state machine for offloading basic system control functions from CPU with support for up to 8 states, 8 outputs and 3 selectable inputs per state
- Integrated general purpose input/output registers and pin rising/falling edge interrupts to simplify software support

51.3 Functional description

51.3.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of the FlexIO. The timing of shift, load and store events are controlled by the Timer assigned to the Shifter via the [SHIFTCTL\[TIMSEL\]](#) register. The Shifters are designed to support either DMA, interrupt or polled operation. The following block diagram provides a detailed view of the Shifter microarchitecture.



Transmit Mode

When configured for Transmit mode ([SHIFTCTL\[SMOD\]](#)=Transmit), the shifter loads data from the [SHIFTBUF](#) register and shift data out when a load event is signalled by the assigned Timer. An optional start/stop bit can also be automatically loaded before/after [SHIFTBUF](#) data by configuring the [SHIFTCFG\[SSTART\]](#), [TIMCFG\[TSTART\]](#) or [SHIFTCFG\[SSTOP\]](#), [TIMCFG\[TSTOP\]](#) registers in the Shifter and Timer.

NOTE

The shifter immediately loads a stop bit when the Shifter is initially configured for Transmit mode if a stop bit is enabled.

The Shifter Status Flag ([SHIFTSTAT\[SSF\]](#)) and any enabled interrupts or DMA requests are set when data has been loaded from the [SHIFTBUF](#) register into the Shifter or when the Shifter is initially configured into Transmit mode. The flag clears when new data has been written into the [SHIFTBUF](#) register or a logic 1 is written to this flag. In transmit mode, any write of the [SHIFTBUF](#) register clears the corresponding Shifter Status Flag. It does not matter what is writing or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/cleared for each mode is documented in the SSF register description.

The Shifter Error Flag ([SHIFTErr\[SEF\]](#)) and any enabled interrupts are set when an attempt to load data from an empty [SHIFTBUF](#) register occurs (buffer underrun). The flag can be cleared by writing it with logic 1.

Receive Mode

When configured for Receive mode ([SHIFTCTL\[SMOD\]](#)=Receive), the shifter shifts data in and store data into the [SHIFTBUF](#) register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the [SHIFTCFG\[SSTART\]](#), [TIMCFG\[TSTART\]](#) or [SHIFTCFG\[SSTOP\]](#), [TIMCFG\[TSTOP\]](#) registers in the Shifter and Timer.

The Shifter Status Flag ([SHIFTSTAT\[SSF\]](#)) and any enabled interrupts or DMA requests are set when data has been stored into the [SHIFTBUF](#) register from the Shifter. The flag clears when the data has been read from the [SHIFTBUF](#) register or a logic 1 is written to this flag. Any read of the [SHIFTBUF](#) register clears the corresponding Shifter Status Flag when Shifter is configured in Receive mode. It does not matter what is reading or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/clear for each mode is documented in the SSF register description.

The Shifter Error Flag ([SHIFTErr\[SEF\]](#)) and any enabled interrupts are set when an attempt to store data into a full [SHIFTBUF](#) register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

Match Store Mode

When configured for Match Store mode ([SHIFTCTL\[SMOD\]](#)=Match Store), the shifter shifts data in, check for a match result and store matched data into the [SHIFTBUF](#) register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the [SHIFTCFG\[SSTART\]](#), [TIMCFG\[TSTART\]](#) or [SHIFTCFG\[SSTOP\]](#), [TIMCFG\[TSTOP\]](#) registers in the Shifter and Timer. Up to 16-bits of data can be compared using [SHIFTBUF\[31:16\]](#) to configure the data to be matched and [SHIFTBUF\[15:0\]](#) to mask the match result.

The Shifter Status Flag ([SHIFTSTAT\[SSF\]](#)) and any enabled interrupts or DMA requests are set when a match occurs and matched data has been stored into the [SHIFTBUF](#) register from the Shifter. The flag clears when the matched data has been read from the [SHIFTBUF](#) register or a logic 1 is written to this flag. Any read of the [SHIFTBUF](#) register clears the corresponding Shifter Status Flag when Shifter is configured in Match Store mode. It does not matter what is reading or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/clear for each mode is documented in the SSF register description.

The Shifter Error Flag ([SHIFTErr\[SEF\]](#)) and any enabled interrupts are set when an attempt to store matched data into a full [SHIFTBUF](#) register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

Match Continuous Mode

When configured for Match Continuous mode ([SHIFTCTL\[SMOD\]](#)=Match Continuous), the shifter shifts data in and continuously check for a match result whenever a shift event is signalled by the assigned Timer. Up to 16-bits of data can be compared using [SHIFTBUF\[31:16\]](#) to configure the data to be matched and [SHIFTBUF\[15:0\]](#) to mask the match result.

The Shifter Status Flag ([SHIFTSTAT\[SSF\]](#)) and any enabled interrupts or DMA requests are set when a match occurs. The flag clears automatically as soon as there is no longer a match between Shifter data and [SHIFTBUF](#) register. **The flag cannot be cleared by reading the [SHIFTBUF](#) register.**

The Shifter Error Flag ([SHIFTErr\[SEF\]](#)) and any enabled interrupts are set when a match occurs. The flag clears when there is a read from the [SHIFTBUF](#) register or it written with logic 1.

State Mode

Using State mode enables the user to implement any state machine with up to 8 states, 8 outputs and 3 selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU, which could potentially remain in a low power mode.

When configured for State mode ([SHIFTCTL\[SMOD\]](#)=State), the [SHIFTBUF](#) register is used to drive the output and compute next state values when the Shifter has been selected by the current state pointer ([SHIFTSTATE\[STATE\]](#)). The following diagram provides a detailed view of Shifter microarchitecture when configured for State mode.

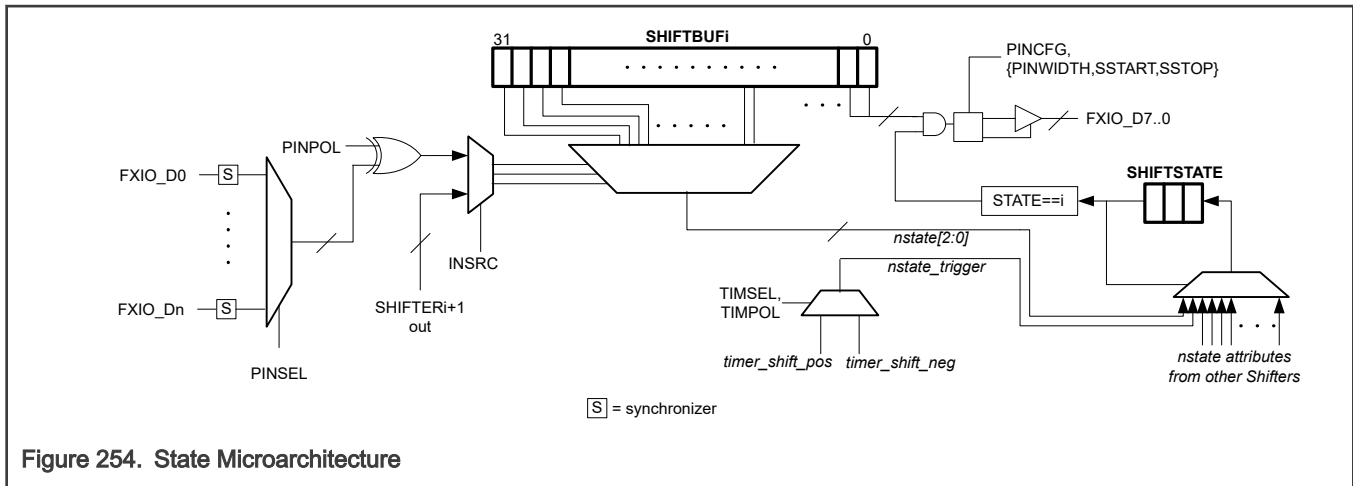


Figure 254. State Microarchitecture

When Shifter *i* has been selected by the current state pointer, output pins FXIO_D[7:0] are driven by SHIFTBUF[i][31:24] using the configuration set by SHIFTCTL[PINCFG]. When set, SHIFTCFGi[PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]] are respectively used to disable the output drive on pins FXIO_D[7:0] for state machine applications which require less than 8 output pins.

The next state value is computed using the 3 input pins selected by SHIFTCTL[PINSEL] together with SHIFTBUF[i][23:0].

NOTE

Each state could potentially use a different set of 3 input pins.

The following table details how the next state value is computed when the current state pointer is pointing to Shifter *i*.

Table 521. Next State computation for SHIFTSTATE[STATE]=i

FXIO_D[PINSEL+2]	FXIO_D[PINSEL+1]	FXIO_D[PINSEL]	Next State Value
0	0	0	SHIFTBUF[i][2:0]
0	0	1	SHIFTBUF[i][5:3]
0	1	0	SHIFTBUF[i][8:6]
0	1	1	SHIFTBUF[i][11:9]
...	...	...	...
1	1	1	SHIFTBUF[i][23:21]

Note that other shifters/timers could potentially be configured to drive the input pins of a given state, allowing the user to create complex combinations of shifters/timers as desired e.g. the output of a Shifter configured for logic mode could potentially be used to drive a state machine input.

The next state transition is triggered using the Timer output selected by SHIFTCTL[TIMSEL] with polarity controlled by SHIFTCTL[TIMPOL]. Note that each state could potentially use a different Timer to trigger each next state transition, allowing a variety of internal/external trigger sources and clocking configurations to be used (see [Timer section](#) for more detail).

The current state pointer defaults to Shifter 0 at reset, however it can be written by the user to select a different Shifter for the initial state. If the current state pointer selects a Shifter which is not configured for State mode, then outputs are not driven and a next state transition is never triggered.

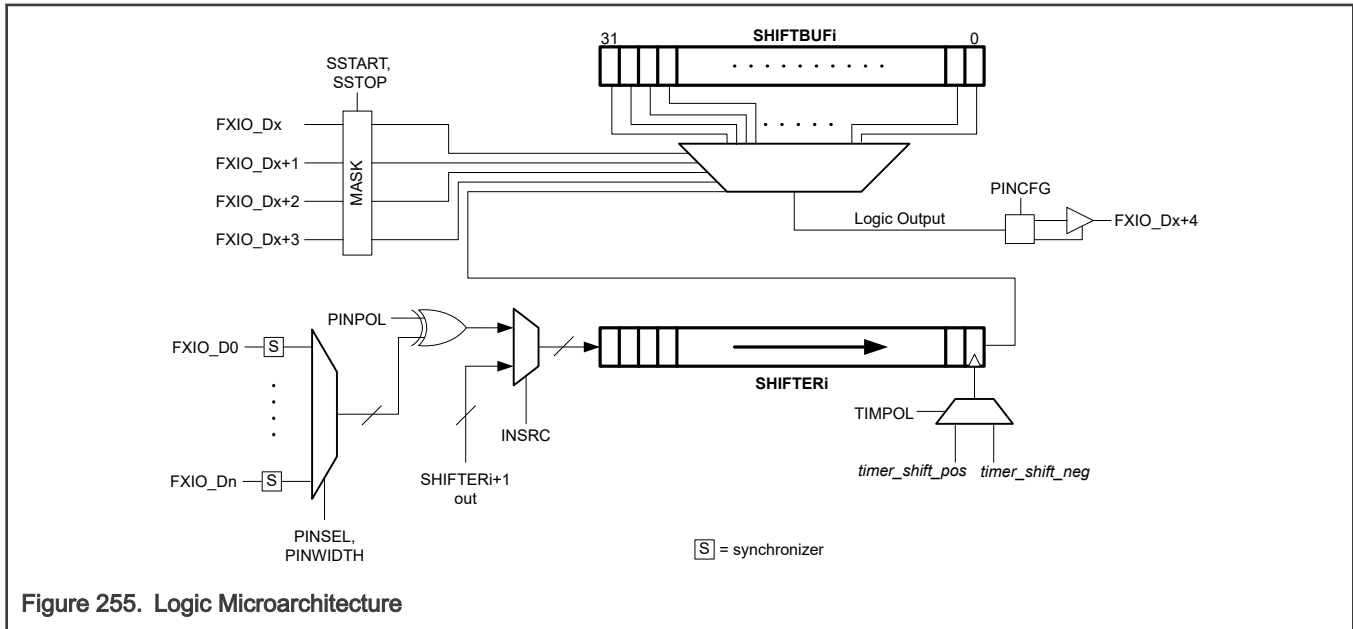
The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set whenever the Shifter has been selected by the current state pointer. The flag clears when the current state pointer is updated to a different Shifter.

Logic Mode

Using logic mode enables the user to implement a small amount of programmable digital logic within a FlexIO Shifter.

When configured for Logic mode ([SHIFTCTL\[SMOD\]=Logic](#)), the [SHIFTBUFi](#) register is used to implement a 5-input, 32-bit programmable logic look-up table. The following diagram provides a detailed view of Shifter microarchitecture when configured for Logic mode.

The [SHIFTBUFi](#) configures the lookup table for the 4 pin inputs and the [SHIFTERi](#) can optionally be used to configure a feedback or delayed pin source as the 5th input to the look up table.



The look-up table is driven using 4 pin inputs (maskable using [SHIFTCFG\[SSTOP\]](#) and [SHIFTCFG\[SSTART\]](#)) plus 1 input from the internal shifter and can be configured to drive an output pin using the [SHIFTCTL\[PINCFG\]](#) field. Pin inputs and outputs are fixed for each logic look-up table and are not selectable. The following table lists the logic output value selected by the look-up table for Shifter 'i'.

Table 522. Logic Look-up table for Shifter 'i'

SHIFTERi[0]	FXIO_D[x+3] ¹	FXIO_D[x+2]	FXIO_D[x+1]	FXIO_D[x]	Logic Output to FXIO_D[x+4]
0	0	0	0	0	SHIFTBUFi[0]
0	0	0	0	1	SHIFTBUFi[1]
0	0	0	1	0	SHIFTBUFi[2]
0	0	0	1	1	SHIFTBUFi[3]
...	...	...	...	...	...
1	1	1	1	1	SHIFTBUFi[31]

1. for Shifter i=0...3, x=i
for Shifter i=4...7, x=i+4

To minimize output glitches, [SHIFTCFG\[SSTOP\]](#) and [SHIFTCFG\[SSTART\]](#) can be used to mask unused input pins. When set, {SSTOP[1:0], SSTART[1:0]} mask FXIO_D[x+3]...FXIO_D[x] inputs respectively, so that any transitions on these pins does not cause the logic output to glitch.

Note that other shifters/timers could potentially be configured to drive the input pins of a given look-up table, allowing the user to concatenate look-up tables or create complex combinations of shifters/timers as desired.

SHIFTCFG[PWIDTH] controls the number of delay stages introduced by the internal shifter input (SHIFTERi[0]). For example, when configured for 1-bit shift (PWIDTH=0), the internal shifter introduces a 32 Shift clock delay before passing its input (selected by **SHIFCTL[PINSEL]**) to the look-up table. When configured for 32-bit shift (PWIDTH=16...31), the internal shifter introduces a 1 Shift clock delay to its input.

The Shifter Status Flag (**SHIFTSTAT[SSF]**) and any enabled interrupts or DMA requests are set whenever the output pin allocated to the logic look-up table has a value of 1 (after being synchronized to the FlexIO clock). The flag clears when the output pin has a value of 0. This also allows the SSF flag to be used as a trigger to a Timer if desired.

The Shifter Error Flag (**SHIFTErr[SEF]**) and any enabled interrupts are set when the output pin allocated to the logic look-up table has asserted. The flag can be cleared by writing it with logic 1.

The logic mode input pins (including pins driven by other shifters/timers) are first synchronized to the FlexIO functional clock before they are input to the programmable logic look-up table.

51.3.2 Timer Operation

The FlexIO 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock. They can perform generic timer functions such as generating a clock or select output or a PWM waveform. Timers can be configured to enable in response to a trigger, pin or shifter condition; decrement always or only on a trigger or pin edge; reset in response to a trigger or pin condition; and disable on a trigger or pin condition or on a timer compare. Timers can optionally include a start condition and/or stop condition.

Each timer operates independently, although a timer can be configured to enable or disable at the same time as the previous timer (eg: timer1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently and can be configured to be a timer output, shifter status flag, pin input or an external trigger input (refer to the chip-specific FlexIO information for details on the external trigger connections). The trigger configuration is separate from the pin configuration, which can be configured for input, output data or output enable.

The Timer Configuration Register (**TIMCFGn**) should be configured before setting the Timer Mode (**TIMOD**).

Timer 8-bit Baud Counter Mode

In 8-bit Baud Counter Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the baud rate of the shift clock and the upper 8-bits are used to configure the number of shift clock edges in the transfer. When the lower 8-bits decrement to zero, the timer output is toggled and the lower 8-bits reload from the compare register. The upper 8-bits decrement when the lower 8-bits equal zero and decrement.

Note that a timer reset event in 8-bit Baud Counter Mode only resets the lower 8-bit counter, the upper 8-bit counter is not affected and can decrement if the timer reset is configured to update the state of the timer output, and the timer output toggles as a result of the timer reset event.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

Timer 8-bit High PWM Mode

In 8-bit High PWM Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the timer output high period and the upper 8-bits are used to configure the timer output low period. The lower 8-bits decrement when the output is high. When the lower 8-bits equal zero and decrement, the timer output is cleared and the lower 8-bits are reloaded from the compare register. The upper 8-bits decrement when the output is low. When the upper 8-bits equal zero and decrement, the timer output is set and the upper 8-bits are reloaded from the compare register.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

Timer 16-bit Counter Mode

In 16-bit Counter Mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (eg: `TIMDEC[1:0] != 10` or 11) or the number of shift clock edges in the transfer (eg: `TIMDEC[1:0] = 10` or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer compare event.

Timer 16-bit Counter Disable Mode

In 16-bit Counter Disable Mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (eg: `TIMDEC[1:0] != 10` or 11) or the number of shift clock edges in the transfer (eg: `TIMDEC[1:0] = 10` or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer disable event.

Timer 8-bit Word Counter Mode

In 8-bit Word Counter Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the number of shift clock edges in each word and the upper 8-bits are used to configure the number number of words in the transfer. When the lower 8-bits decrement to zero, the timer output is toggled and the lower 8-bits reload from the compare register. The upper 8-bits only decrement when the lower 8-bits equal zero and decrement.

A timer compare event occurs when the lower 8-bits equal zero and decrements. The timer status flag is set when the upper 8-bits equal zero and decrements.

Timer 8-bit Low PWM Mode

In 8-bit Low PWM Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the timer output low period and the upper 8-bits are used to configure the timer output high period. The lower 8-bits decrement when the output is low. When the lower 8-bits equal zero and decrement, the timer output is set and the lower 8-bits are reloaded from the compare register. The upper 8-bits decrement when the output is high. When the upper 8-bits equal zero and decrement, the timer output is cleared and the upper 8-bits are reloaded from the compare register.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

Timer 16-bit Input Capture Mode

In 16-bit Input Capture Mode, the 16-bit counter uses a fixed initial value of 0xFFFF to load the counter. Whenever a timer counter disable condition is detected (as configured by `TIMDIS`), the timer status flag is set and the inverted timer counter value is stored in the timer compare register, then the timer counter is immediately restarted from 0xFFFF. The timer output is always output on the FlexIO trigger outputs (which can also be used by other timers as an input trigger) and can also be output to a pin (`PINCFG/PINPOL/PINSEL`) or a shifter.

If the timer status flag is already set, then the input capture event is missed. When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads with 0xFFFF. The timer output can be used to count the number of counter overflows between two input capture events. Input capture mode is not intended to be used to control shift registers, it is to enable software to count the number of bits in a variable length transfer or to monitor the period or frequency of an input.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer disable event. Whenever the status flag asserts, the inverted timer counter value is stored in the timer compare register.

Timer Enable and Start Bit

When the `TIMOD` is configured for the desired mode, and the condition configured by timer enable (`TIMENA`) is detected then the following events occur. When `ONETIM` is set, then the timer status flag must also be clear to generate a timer enable event, otherwise the timer enable event is blocked. This can be used to enforce software intervention after each timer iteration.

- Timer counter loads the current value of the compare Register and start decrementing as configured by [TIMDEC](#).
- Timer output may update to its initial state depending on the [TIMOUT](#) configuration. Shifters that are controlled by this timer do not see this as a rising edge on the timer shift clock.
- Transmit shifters controlled by this timer either output their start bit value, or load the shift register from the shift buffer and output the first bit, as configured by [SSTART](#).

If the Timer start bit is enabled, the timer counter reloads with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when [TIMOUT](#)=1), a shifter that is configured to shift on falling edge and load on the first shift does not load correctly.

Timer Decrement and Reset

The Timer then generates the timer output and timer shift clock depending on the [TIMOD](#) and [TIMDEC](#) fields. The shifter clock is either equal to the timer output (when [TIMDEC](#) != 10 or 11) or equal to the decrement clock (when [TIMDEC](#) = 10 or 11). When [TIMDEC](#) is configured to decrement from a pin or trigger, the timer decrements on both rising and falling edges.

If a Timer is configured to decrement on the FlexIO functional clock divided by 16 or 256 (when [TIMDEC](#) = 100 or 101), then a common prescaler that is shared by all Timers is used to generate the two divide ratios. This prescaler is reset when all Timers are either idle or configured to not use the prescaler ([TIMDEC](#) != 100 or 101).

When the Timer is configured to reset as configured in the [TIMRST](#) field then the Timer counter loads the current value of the compare register again. The timer output and timer shift clock can be configured to update on timer reset, as configured by [TIMOUT](#). This can result in a timer shift clock edge if the timer output toggles as a result of the timer reset. In 8-bit Baud Counter mode this would also decrement the upper 8-bits of the counter.

In general, when the timer counter decrements to zero a timer compare event is triggered. The timer compare event causes the timer counter to load the contents of the timer compare register, the timer output to toggle, any configured transmit shift registers to load, and any configured receive shift registers to store. Depending on the timer mode, the timer status flag may also be set.

Timer Disable and Stop Bit

When the is Timer is configured to add a stop bit on each compare, the following additional events occur.

- Transmit shifters controlled by this timer output their stop bit value (if configured by [SSTOP](#)).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by [SSTOP](#).
- On the first rising edge of the shifter clock after the compare, the timer counter reloads the current value of the Compare Register.

Transmit shifters must be configured to load on the first shift when the timer is configured to insert a stop bit on each compare.

When the condition configured by timer disable ([TIMDIS](#)) is detected, the following events occur.

- Timer counter reloads the current value of the Compare Register and start decrementing as configured by [TIMDEC](#).
- Timer output clears. Shifters that are controlled by this timer do not see this as a falling edge on the timer shift clock, but can generate a shift event if the timer shift clock would otherwise generate one.
- Transmit shifters controlled by this timer output their stop bit value (if configured by [SSTOP](#)).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by [SSTOP](#).

If the timer stop bit is enabled, the timer counter continues decrementing until the next rising edge of the shift clock is detected, at which point it finishes. Although the timer output is forced low during the stop bit, the timer shift clock can toggle during the stop bit, but does not generate shift events.

A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). When [ONETIM](#) is set, the timer status flag must be clear before the next timer enable condition is detected. Receive shift registers with stop bit enabled store the contents of the shift register into the shift buffer and verify the state of the input data on the configured shift edge while the timer is in the stop state.

condition. If there is no configured edge between the timer disable and the next rising edge of the shift clock then the final store and verify do not occur.

51.3.3 Pin operation

The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity. Each timer and shifter can be configured as an input, output data, output enable or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity, since the output enable assertion would cause logic zero to be output on the pin) or to control the enable on the bidirectional output. Any timer or shifter could be configured to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

Parallel Interface

Shifters can be configured to use multiple FlexIO pins in parallel using the [SHIFTCFG\[PWIDTH\]](#) field. [PWIDTH](#) is used to configure the following settings of a shifter:

1. Number of bits shifted per Shift clock.
2. Number of pins driven by the shifter per Shift clock (only on shifters supporting parallel transmit i.e. SHIFTER0, SHIFTER4).
3. Number of pins sampled by the shifter per Shift clock (only on Shifter supporting parallel receive i.e. SHIFTER3, SHIFTER7).

When configured for parallel shift, either 4, 8, 16 or 32-bits can be shifted on every Shift clock. If an adjacent shifter is selected as the input source ([SHIFTCFG\[INSRC\]=1](#)), the least significant 4, 8, 16 or 32-bits from the adjacent shifter is sampled on each Shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), the shifter can be configured to sample multiple pins ([SHIFTCFG\[INSRC\]=0](#)), with [PWIDTH](#) and [PINSEL](#) selecting the pins as follows: `FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]`. Note that if [PWIDTH](#) is less than the number of bits being shifted on each Shift clock, then the most significant bits are masked with 0 e.g. if [PINSEL=7](#) and [PWIDTH=6](#), then `SHIFTER[31:24]` samples `{0,0,FXIO_D[12:7]}` on each Shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), the shifter can be configured to drive multiple pins using [SHIFCTL\[PINCFG\]](#), with [PWIDTH](#) and [PINSEL](#) selecting the pins as follows: `FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]`. Note that if [PWIDTH](#) is less than the number of bits being shifted on each Shift clock, then the most significant pins are not driven e.g. if [PINSEL=7](#) and [PWIDTH=6](#), then `SHIFTER[5:0]` drives only `FXIO_D[12:7]` on each Shift clock.

Pin Synchronization

When configuring a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized to the FlexIO clock before the signal is used by a timer or shifter. This introduces a small latency of between 0.5 to 1.5 FlexIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FlexIO clock cycles.

If an input is used by more than one timer or shifter then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

Note that FlexIO pins are also connected internally, configuring a FlexIO shifter or timer to output data on an unused pin makes an internal connection that allows other shifters and timer to use this pin as an input. This allows a shifter output to be used to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized to the FlexIO clock and therefore incurs a 1 cycle latency.

So when using a Pin input as a Timer Trigger, Timer Clock or Shifter Data Input, the following synchronization delays occur:

1. 0.5 to 1.5 FlexIO clock cycles for external pin
2. 1 FlexIO clock cycle for an internally driven pin

For timing considerations such as output valid time and input setup time for specific applications (SPI Master, SPI Slave, I2C Master, I2S Master, I2S Slave) please refer to the [FlexIO Application Information Section](#).

Pin Override

The state of any FlexIO pin can be overridden at any time by software. The Pin Output Enable register configures any pin as an output and drives that pin with the value in the Pin Output Data Register.

Alias registers for the Pin Output Enable/Data Registers also exist, writing a logic one to an alias register updates the corresponding register bits in both the Pin Output Enable Register and the Pin Output Data Register as follows.

- Pin Output Disable Register clears Output Enable Register and clear Output Data Register.
- Pin Output Clear Register sets Output Enable Register and clear Output Data Register.
- Pin Output Set Register sets Output Enable Register and set Output Data Register.
- Pin Output Toggle Register sets Output Enable Register and toggle Output Data Register.

Pin Interrupt

The state of any FlexIO pin can be read by software at any time. Software can also configure any pin to set a status flag when either a rising and/or falling edge is detected on that pin. The pin status flag can also be configured to generate an interrupt.

51.3.4 Low Power Modes

The FlexIO remains functional during low power modes, provided the Doze Enable bit ([CTRL\[DOZEN\]](#)) is clear and the FlexIO functional clock remains enabled.

The exception to this is in Low Leakage Stop mode. In this case, the Doze Enable ([CTRL\[DOZEN\]](#)) bit is ignored and the FlexIO waits for all Timers to complete any pending operation before acknowledging Low Leakage Stop mode entry.

51.3.5 Debug Mode

The FlexIO remains functional in debug mode provided the Debug Enable bit ([CTRL\[DBGGE\]](#)) is set.

51.3.6 Clocks

Functional clock

The FlexIO functional clock is asynchronous to the bus clock and can remain enabled in low power modes. The FlexIO functional clock must be enabled before accessing any FlexIO registers. Provided the FlexIO functional clock is at least equal to the bus clock, the [CTRL\[FASTACC\]](#) bit can be set to support fast register accesses.

Bus clock

The bus clock is only used for bus accesses to the control and configuration registers.

51.3.7 Reset

Chip reset

The logic and registers for the FlexIO are reset to their default state on a chip reset.

Software reset

The FlexIO implements a software reset bit in its Control Register. The [CTRL\[SWRST\]](#) resets all logic and registers to their default state, except for the CTRL itself.

51.3.8 Interrupts and DMA Requests

The following table illustrates the status flags that can generate the FlexIO interrupt and DMA requests.

Table 523. FlexIO Interrupts and DMA Requests

Flag	Description	Interrupt	DMA Request	Low Power Wakeup
SSF	Shifter Status Flag.	Y	Y	Y
SEF	Shifter Error Flag.	Y	N	Y
TSF	Timer Status Flag.	Y	Y	Y
PSF	Pin Status Flag.	Y	N	Y
ETSF	External Trigger Status Flag.	Y	N	Y

51.3.9 Peripheral Triggers

The connection of the FlexIO peripheral triggers with other peripherals are device specific.

Output Triggers

Each FlexIO Timer generates an output trigger equal to the timer output. The output trigger is not affected by the timer pin polarity configuration.

Input Trigger

FlexIO supports multiple external trigger inputs that can be used to trigger one or more FlexIO timers. If a rising edge is detected on an external trigger when the FlexIO is enabled, then the external trigger status flag is set. The external triggers are synchronized to the FlexIO functional clock and must assert for at least two cycles of the FlexIO functional clock to be sampled correctly.

51.4 Application Information

This section provides examples for a variety of FlexIO module applications.

See the register descriptions for bitfield details.

51.4.1 UART Transmit

UART transmit can be supported using one Timer, one Shifter and one Pin (two Pins if supporting CTS). The start and stop bit insertion is handled automatically and multiple transfers can be supported using DMA controller. The timer status flag can be used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention, before transmitting a break or idle character the **SSTART** and **SSTOP** fields should be altered to transmit the required state and the data to transmit must equal 0xFF or 0x00. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). Note that when performing byte writes to **SHIFTBUF_n** (or **SHIFTBUFBIS** for transmitting MSB first), the rest of the register remains unaltered allowing an address mark bit or additional stop bit to remain undisturbed.

FlexIO does not support automatic insertion of parity bits.

Table 524. UART Transmit Configuration

Register	Value	Comments
SHIFTCFG _n	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTL _n	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting

Table continues on the next page...

Table 524. UART Transmit Configuration (continued)

Register	Value	Comments
		PINPOL , or can support open drain by setting PINPOL =0x1 and PINCFG =0x1.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP [15:8] = (number of bits x 2) - 1. Set TIMCMP [7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_2222	Configure start bit, stop bit, enable on trigger asserted and disable on compare. Can support CTS by configuring TIMEN =0x3.
TIMCTLn	0x01C0_0001	Configure dual 8-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL =0x1 (for Pin 1) and PINPOL =0x1.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF [7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS [7:0] register instead.

An alternative configuration is possible that supports slower baud rates but requires two Timers.

Table 525. UART Transmit Configuration for Slow Baud Rate

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL , or can support open drain by setting PINPOL =0x1 and PINCFG =0x1.
TIMCMPn	0x0000_000F	Configure for 8-bit transfer. Set TIMCMP [15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0030_2622	Configure start bit, stop bit, enable on trigger rising edge, decrement on trigger and disable on compare.
TIMCTLn	0x0740_0003	Configure 16-bit counter using Timer 1 output as internal trigger source.

Table continues on the next page...

Table 525. UART Transmit Configuration for Slow Baud Rate (continued)

Register	Value	Comments
TIMCMP(n+1)	0x0000_0001	Configure baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (baud rate divider / 2) - 1.
TIMCFG(n+1)	0x0000_1200	Configure enable on trigger asserted and disable on Timer 0 disable. Can support CTS by configuring TIMEN=0x3.
TIMCTL(n+1)	0x01C0_0003	Configure 16-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL=0x1 (for Pin 1) and PINPOL=0x1.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS[7:0] register instead.

51.4.2 UART Receive

UART receive can be supported using one Timer, one Shifter and one Pin (two Timers and two Pins if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers can be supported using the DMA controller. The timer status flag can be used to indicate when the stop bit of each word is received.

Triple voting of the received data is not supported by FlexIO, data is sampled only once in the middle of each bit. Another timer can be used to implement a glitch filter on the incoming data, another Timer can also be used to detect an idle line of programmable length. Break characters cause the error flag to set and the shifter buffer register returns 0x00.

FlexIO does not support automatic verification of parity bits.

Table 526. UART Receiver Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2422	Configure start bit, stop bit, enable on pin posedge and disable on compare. Enable resynchronization to

Table continues on the next page...

Table 526. UART Receiver Configuration (continued)

Register	Value	Comments
		received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x0000_0081	Configure dual 8-bit counter using inverted Pin 0 input.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUFBS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBS[7:0] register instead.

The UART Receiver with RTS configuration uses a 2nd Timer to generate the RTS output. The RTS asserts when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit is detected while the RTS is asserted, the received data is simply ignored.

Table 527. UART Receiver with RTS Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2522	Configure start bit, stop bit, enable on pin posedge with trigger asserted and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x02C0_0081	Configure dual 8-bit counter using inverted Pin 0 input. Trigger is internal using inverted Pin 1 input.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0030_6100	Enable on Timer N enable and disable on trigger falling edge. Decrement on trigger to ensure no compare.
TIMCTL(n+1)	0x0143_0003	Configure 16-bit counter and output on Pin 1. Trigger is internal using Shifter 0 flag.

Table continues on the next page...

Table 527. UART Receiver with RTS Configuration (continued)

Register	Value	Comments
SHIFTBUF _n	Data to receive	Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

51.4.3 SPI Master

SPI master mode can be supported using two Timers, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of 1 clock cycle between the slave select negating and before the next transfer. Writing to the transmit buffer by either core or DMA is used to initiate each transfer.

NOTE

Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 528. SPI Master (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFG _n	0x0000_0000	Start and stop bit disabled.
SHIFCTL _n	0x0083_0002	Configure transmit using Timer 0 on negedge of clock with output data on Pin 0.
SHIFTCFG _(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFCTL _(n+1)	0x0000_0101	Configure receive using Timer 0 on posedge of clock with input data on Pin 1.
TIMCMP _n	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG _n	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0.
TIMCTL _n	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock.
TIMCMP _(n+1)	0x0000_FFFF	Never compare.

Table continues on the next page...

Table 528. SPI Master (CPHA=0) Configuration (continued)

Register	Value	Comments
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUF _n	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 529. SPI Master (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFG _n	0x0000_0021	Start bit loads data on first shift.
SHIFTCTL _n	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on negedge of clock with input data on Pin 1.
TIMCMP _n	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG _n	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0.
TIMCTL _n	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock. Set TIMDIS=3 to keep slave select asserted for as long as there is data in the transmit buffer.

Table continues on the next page...

Table 529. SPI Master (CPHA=1) Configuration (continued)

Register	Value	Comments
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUF _n	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

51.4.4 SPI Slave

SPI slave mode can be supported using one Timer, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The transmit data must be written to the transmit buffer register before the external slave select asserts, otherwise the shifter error flag is set.

NOTE

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

Table 530. SPI Slave (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFG _n	0x0000_0000	Start and stop bit disabled.
SHIFTCTL _n	0x0083_0002	Configure transmit using Timer 0 on falling edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on rising edge of shift clock with input data on Pin 1.
TIMCMP _n	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.

Table continues on the next page...

Table 530. SPI Slave (CPHA=0) Configuration (continued)

Register	Value	Comments
TIMCFGn	0x0120_0600	Configure enable on trigger rising edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF , use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 531. SPI Slave (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Shifter configured to load on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set $TIMCMP[15:0] = (\text{number of bits} \times 2) - 1$.
TIMCFGn	0x0120_6602	Configure start bit, enable on trigger rising edge, disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written

Table continues on the next page...

Table 531. SPI Slave (CPHA=1) Configuration (continued)

Register	Value	Comments
		using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF , use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

51.4.5 I2C Master

I2C master mode can be supported using two Timers, two Shifters and two Pins. One timer is used to generate the SCL output and one timer is used to control the shifters. The two shifters are used to transmit and receive for every word, when receiving the transmitter must transmit 0xFF to tristate the output. FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK. FlexIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (START to Repeated START/STOP), so the compare register needs to be programmed with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin equal to output (although this increases both the clock high and clock low periods by at least 1 FlexIO clock cycle each). The second timer uses the SCL input pin to control the transmit/receive shift registers, this enforces an SDA data hold time by an extra 2 FlexIO clock cycles.

Both the transmit and receive shifters need to be serviced for each word in the transfer, the transmit shifter must transmit 0xFF when receiving and the receive shifter returns the data actually present on the SDA pin. The transmit shifter loads 1 additional word on the last falling edge of SCL pin, this word should be 0x00 if generating a STOP condition or 0xFF if generating a repeated START condition. During the last word of a master-receiver transfer, the transmit SSTOP bit should be set by software to generate a NACK.

The receive shift register asserts an error interrupt if a NACK is detected, but software is responsible for generating the STOP or repeated START condition. If a NACK is detected during master-transmit, the interrupt routine should immediately write the transmit shifter register with 0x00 (if generating STOP) or 0xFF (if generating repeated START). Software should then wait for the next rising edge on SCL and then disable both timers. The transmit shifter should then be disabled after waiting the setup delay for a repeated START or STOP condition.

NOTE

Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The I2C master data valid is delayed 2 cycles because the clock output is passed through a synchronizer before clocking the transmit/receive shifter (to guarantee some SDA hold time). Since the SCL output is synchronous with FlexIO clock, the synchronization delay is 1 cycle and then 1 cycle to generate the output.

Table 532. I2C Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1).

Table continues on the next page...

Table 532. I2C Master Configuration (continued)

Register	Value	Comments
SHIFTCTLn	0x0101_0082	Configure transmit using Timer 1 on rising edge of clock with inverted output enable (open drain output) on Pin 0.
SHIFTCFG(n+1)	0x0000_0020	Start bit disabled and stop bit enabled (logic 0) for ACK/NACK detection.
SHIFTCTL(n+1)	0x0180_0001	Configure receive using Timer 1 on falling edge of clock with input data on Pin 0.
TIMCMPn	0x0000_2501	Configure 2 word transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of words x 18) + 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0102_2222	Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset.
TIMCTLn	0x01C1_0101	Configure dual 8-bit counter using Pin 1 output enable (SCL open drain), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_000F	Configure 8-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_1112	Enable when Timer 0 is enabled, disable when Timer 0 is disabled, enable start bit and stop bit at end of each word, decrement on pin input.
TIMCTL(n+1)	0x01C0_0183	Configure 16-bit counter using inverted Pin 1 input (SCL).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBBS[7:0], use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

51.4.6 I2S Master

I2S master mode can be supported using two Timers, two Shifters and four Pins. One timer is used to generate the bit clock and control the shifters and one timer is used to generate the frame sync. FlexIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FlexIO clock frequency, and the initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure the frame sync is generated one clock cycle before the first output data.

NOTE

Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 533. I2S Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Load transmit data on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_0202	Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1.
TIMCTLn	0x01C3_0281	Configure dual 8-bit counter using inverted Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Clear PINPOL to invert the polarity of the output shift clock.
TIMCMP(n+1)	0x0000_007F	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (number of bits x baud rate divider) - 1.
TIMCFG(n+1)	0x0000_0100	Enable when Timer 0 is enabled and never disable.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter using inverted Pin 3 output (as frame sync). Clear PINPOL to invert the polarity of the output frame sync
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFIS , use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

Table continues on the next page...

Table 533. I2S Master Configuration (continued)

Register	Value	Comments
		Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

51.4.7 I2S Slave

I2S slave mode can be supported using three Timers, two Shifters and four Pins (for single transmit and single receive, other combinations of transmit and receive are possible).

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag is set.

NOTE

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The output valid time of I2S slave is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data

Timer 2 detects the falling edge of frame sync (start of new frame) and asserts output until rising edge of bit clock (when frame sync is normally sampled). Timer 0 detects rising edge of bit clock with Timer 2 output asserted and asserts output for length of frame. Timer 1 detects falling edge of bit clock with Timer 0 output asserted and controls shift registers for 32-bit transfers.

Table 534. I2S Slave Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0103_0002	Configure transmit using Timer 1 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0180_0101	Configure receive using Timer 1 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_007F	Configure two 32-bit transfers per frame. Set TIMCMP[15:0] = (number of bits x 4) - 1.
TIMCFGn	0x0020_2500	Configure enable on pin rising edge (inverted bit clock) with trigger high (Timer 2) and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock).

Table continues on the next page...

Table 534. I2S Slave Configuration (continued)

Register	Value	Comments
TIMCTLn	0x0B40_0203	Configure 16-bit counter using Pin 2 input (bit clock), with Timer 2 output as the trigger.
TIMCMP(n+1)	0x0000_003F	Configure 32-bit transfers. Set $TIMCMP[15:0] = (\text{number of bits} \times 2) - 1$.
TIMCFG(n+1)	0x0020_2500	Configure enable on pin (bit clock) rising edge with trigger (Timer 0) high and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock).
TIMCTL(n+1)	0x0340_0283	Configure 16-bit counter using inverted Pin 2 input (bit clock), with Timer 0 output as the trigger.
TIMCMP(n+2)	0x0000_0000	Compare on zero (first edge).
TIMCFG(n+2)	0x0020_6400	Configure enable on inverted pin (frame sync) rising edge and disable on trigger falling edge (bit clock), initial clock state is logic 1 and decrement on inverted pin input (frame sync).
TIMCTL(n+2)	0x04C0_0383	Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 inverted input (bit clock) as the trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS , use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS , use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

51.4.8 SENT Receiver

SENT receiver can be supported by using one Timer and one Pin. The Timer is configured as Input-Capture mode, and captures the counter value at falling edge of the Pin input. Once the counter value is captured, the counter is automatically restarted from counter value 0. Therefore, the captured value is always indicating the period between previous falling edge and current falling edge. CPU interrupt or DMA trigger can be configured at each capture of the counter, and the entire SENT frame decoding with the latest tick width adjustment is performed by CPU software.

Table 535. SENT Receiver Configuration

Register	Value	Comments
TIMCMPn	-	Counter value at the falling edge of the Pin is stored in this register.
TIMCFGn	0x0000_6000	Timer always enabled, Timer disabled on Trigger falling edge, Decrement counter on FlexIO clock.
TIMCTLn	0xnn40_0007	Single 16-bit input capture mode, Timer pin input and output are selected by PINSEL, Timer pin output disabled, Internal trigger selected, Select Pin nn as Trigger.

51.4.9 Camera Interface

Camera Interface can be supported using one Timer, one or more Shifters and multiple Pins. Multiple transfers can be supported using DMA controller.

The example below describes FlexIO configuration for interfacing to an 8-bit CMOS sensor with PCLK, VSYNC, HREF and D[7:0] outputs. The example uses a 128-bit buffer to capture 16-pixels of image data before interrupt or DMA transfer, however a bigger or smaller buffer may be used depending on system DMA performance and FlexIO resource usage by other applications. Note that additional timers may be used to track number of pixels per row and number of rows per frame or HREF/VSYNC may be assigned as GPIO interrupts for software tracking.

Table 536. Camera Interface Configuration for 8-bit CMOS sensor

Register	Value	Comments
SHIFTCFGn...n+2 ¹	0x0007_0100	Configure 8-bit parallel shift in from adjacent shifter.
SHIFTCFGn+3	0x0007_0000	Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]).
SHIFTCTLn...n+3	0x0080_0001	Configure receive using Timer 0 on negedge of clock.
TIMCMPn	0x0000_001F	Configure 16-pixel (8 bits/pixel x 16 pixels = 128-bits) transfer. Set TIMCMP[15:0] = (number of pixels x 2) - 1.
TIMCFGn	0x0120_6600	Configure enable on trigger (HREF) rising edge and disable on trigger falling edge, initial Shift clock state is logic 0 and decrement on PCLK input.
TIMCTLn	0x12C0_0803	Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger.
SHIFTBUFn...n+3	Data to receive	Received data can be read from SHIFTBUFn...n+3, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

1. n=0 or 4

51.4.10 Motorola 68K/Intel 8080 Bus Interface

The Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by Smart/Asynchronous LCD controllers. In conjunction with GPIO, FlexIO is able to drive these interfaces using one Timer and one Shifter, although additional Shifters could be used to support large transfers via the DMA controller.

The configuration below provides an example of how to drive a 16-bit 68K or 8080 bus. For a 8080 bus, two GPIO are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

Table 537. Motorola 68K/Intel 8080 Write Configuration

Register	Value	Comments
SHIFTCFG0...7	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCTL0	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with data output to FXIO_D[15:0].
SHIFTCTL1...7	0x0000_0002	Configure transmit using Timer 0 on posedge of clock.
TIMCMP0	0x0000_0101 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_2200	Configure enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	0x01C3_1001 (Motorola 68K, 1-beat) 0x1DC3_1001 (Motorola 68K, 16-beats) 0x01C3_1081 (Intel 8080, 1-beat) 0x1DC3_1081 (Intel 8080, 16-beats)	Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, nWR pin for 8080), with Shifter 0 (1-beat) or Shifter 7 (16-beats) flag as the inverted trigger.
SHIFTBUF0...7	Data to transmit	Transmit data can be written to SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats) to initiate a transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

Table 538. Motorola 68K/Intel 8080 Read Configuration

Register	Value	Comments
SHIFTCFG0...6	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCFG7	0x000F_0000	Configure 16-bit parallel shift in from pin.

Table continues on the next page...

Table 538. Motorola 68K/Intel 8080 Read Configuration (continued)

Register	Value	Comments
SHIFTCTL0...7	0x0080_0001	Configure receive using Timer 0 on negedge of clock with data input from FXIO_D[15:0].
TIMCMP0	0x0000_0101 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_2220	Configure stop_bit, enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	0x1DC3_1001 (Motorola 68K, 1 beat) 0x01C3_1001 (Motorola 68K, 16 beats) 0x1DC3_1181 (Intel 8080, 1 beat) 0x01C3_1181 (Intel 8080, 16 beats)	Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (nRD pin for 8080), with Shifter 7 flag (1-beat) or Shifter 0 flag (16-beats) as the inverted trigger.
SHIFTBUF0...7	Data received	Received data can be read from SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats), use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

In general, any operation to a 68K/8080 bus slave begins with a register write cycle followed by one or more data read or write cycles. To accomplish this, the following program flow should be used:

1. Configure FlexIO with 1-beat write configuration
2. Configure GPIO to assert nCS, RS pins (and deassert RDWR pin for 68K)
3. Write register index data to SHIFTBUF0[15:0]
4. Configure GPIO to deassert RS pin (and assert RDWR pin for 68K data read)
5. Configure FlexIO with desired read or write configuration (e.g. 1 or 16-beats)
6. Use the Shifter Status Flag to trigger interrupt or DMA driven data transfers to/from SHIFTBUF registers
7. Configure GPIO to deassert nCS pin

51.4.11 Low Power State Machine

The configuration below details a hypothetical state machine example to illustrate the flexibility allowed when using Shifter state mode.

In this example, FlexIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted (This assumes comparator is connected to external trigger 15. See the chip-specific FlexIO information for actual FlexIO trigger mappings). Throughout this operation, the CPU can be kept in a STOP/VLPS mode by clearing the CTRL[DOZEN] bit and ensuring the FLEXIO_CLK is enabled. The state diagram below shows the states and transitions implemented by this example.

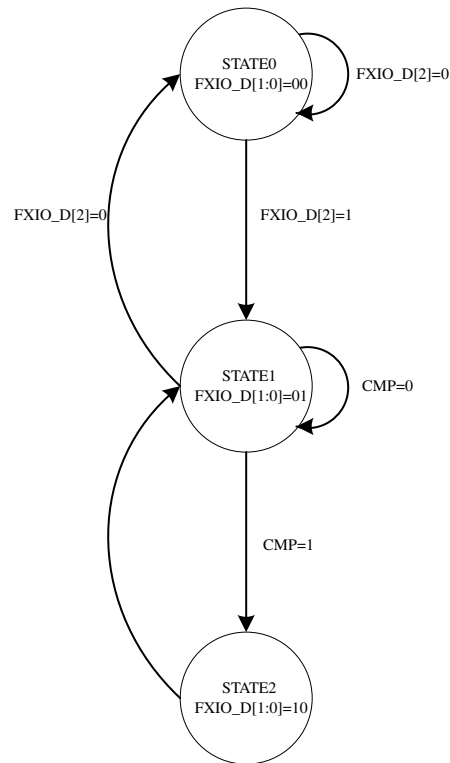


Figure 256. State Diagram

Table 539. State Machine Configuration

Register	Value	Comments
SHIFTCFG0...2	0x0000_0003	Enable FXIO_D[1:0] as outputs.
SHIFTCTL0	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF0	0x0020_8208	State0: Drive FXIO_D[1:0]=00, transition to State0 if FXIO_D[2]=0, State1 if FXIO_D[2]=1.
SHIFTCTL1	0x0000_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output high to trigger next state.
SHIFTBUF1	0x0140_8408	State1: Drive FXIO_D[1:0]=01, transition to State0 if FXIO_D[2]=0, State1 if CMP=0, State2 if CMP=1 (FXIO_D[3]=1)

Table continues on the next page...

Table 539. State Machine Configuration (continued)

Register	Value	Comments
SHIFTCTL2	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF2	0x0224_9249	State2: Drive FXIO_D[1:0]=10, transition to State1 when Timer0 output low
TIMCMP0	0x0000_FFFF	Configure baud rate of divide by 131072 of the FlexIO clock. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_0000	Configure timer always enabled.
TIMCTL0	0x0000_0003	Configure single 16-bit counter.
TIMCFG1	0x0010_7600	Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge.
TIMCTL1	0x0F03_0303	Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected.

51.4.12 Keypad Interface

The Keypad Interface can support a 3x4 keypad matrix using 3 timers and 3 shifters, although a larger matrix could be supported using additional shifters. The configuration is designed for 4 columns which are configured as active low open drain pins, and 3 rows which are configured as input pins with pullup resistors enabled.

One shifter is configured in logic mode and is configured to assert its output when any of the row inputs are low, indicating a key is pressed. A timer is used to filter the shifter output to ensure the key is pressed for a minimum amount of time before performing the column scan.

Another shifter is configured for parallel transmit, this shifter is used to scan each column when a key press is detected. Whenever not scanning, the shifter output is configured to assert all active low open drain column outputs to detect any keypress. A dedicated timer is used to control the transmit shifter.

The last shifter is configured for parallel receive, this shifter is used to capture the result of the column scanning for software to decode which key or keys was pressed. This configuration captures the state of both row and column pins for each scan, although the row state could also be deduced by the shift order. A dedicated timer is used to control the receive shifter, it shifts at half the frequency of the transmit shifter.

Once the result of the key scan is available in the receive shifter register, FlexIO continues to monitor the row inputs and can trigger multiple scans from the one key press. To support debouncing, software can decide how many consecutive scans should be considered a single key press.

Since the pins used in logic mode are fixed per shifter, and the shifters that support parallel shifts are limited, this configuration is restricted in what pins and shifters it can use. Increasing the matrix beyond 4 row inputs requires multiple shifters in logic mode and increasing beyond 4 column outputs requires concatenating transmit shifters to create a larger shift register.

Table 540. Keypad Interface Configuration

Register	Value	Comments
SHIFTCFG0	0x0003_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1), configured for 4-bit shift.
SHIFTCTL0	0x0101_0402	Configure transmit using Timer 1 on rising edge of clock generating open drain output on Pins [7:4] (column outputs).
SHIFTBUF0	0x0804_0201	Static data containing the column scan pattern, each column is scanned one-hot with deadline inbetween.
SHIFTCFG1	0x0000_0020	In logic mode, mask input 3.
SHIFTCTL1	0x0000_0007	Configure logic mode.
SHIFTBUF1	0x07ff_07ff	Static data configuring logic mode LUT. Output asserts if pins [3:1] are logic 0.
SHIFTCFG3	0x0007_0000	Configured for 8-bit shift.
SHIFTCTL3	0x0280_0001	Configure receive using Timer 2 on falling edge of clock with input data on Pins [7:0] (both rows and columns, pin 0 is don't care).
TIMCMP0	0x0000_00ff	Configures pre-scanning glitch filter to 256 FlexIO clock cycles. For different filter cycles, set TIMCMP[15:0] = (filter cycles) - 1.
TIMCFG0	0x0103_6600	Configure enable on trigger rising edge and disable on trigger falling edge. Initial clock state is logic 0 and is not affected by reset.
TIMCTL0	0x0540_0003	Configure 16-bit counter using Shifter 1 flag (logic state) as trigger.
TIMCMP1	0x0000_0f3f	Configure 8 shifts (twice for each column) at column scan rate of divide by 128. For different scan frequency, set TIMCMP[7:0] = (scan divider / 2) - 1.
TIMCFG1	0x0020_2622	Enable on trigger rising edge, disable on compare, start and stop bits are enabled.
TIMCTL1	0x0340_0001	Configure dual 8-bit counter using Timer 0 output as the trigger.
TIMCMP2	0x0000_0801	Configure 4 shifts at half the frequency of Timer 1 trigger.
TIMCFG2	0x0110_2100	Enable on Timer 1 enable, disable on compare, decrement on trigger input

Table continues on the next page...

Table 540. Keypad Interface Configuration (continued)

Register	Value	Comments
		with output initially negated and not affected by reset
TIMCTL2	0x0740_0001	Configure dual 8-bit counter using Timer 1 output as the trigger.
SHIFTBUF3	Keypad Scan Result	Keypad scan result can be read from SHIFTBUF3, each byte is the result of one scan with both row and column pin state from the scan (note that pin 0 is not used). Use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

51.5 FlexIO Signal Descriptions

Table 541.

Signal	Description	I/O
FXIO_Dn (n=0...15)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

51.6 Memory Map and Registers

51.6.1 FLEXIO register descriptions

NOTE

An invalid register access results in a bus error. This includes reading a write-only register, writing a read-only register or accessing an invalid address.

51.6.1.1 FLEXIO memory map

FLEXIO0 base address: 4003_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	R	0201_0003h
4h	Parameter Register (PARAM)	32	R	1010_0808h
8h	FlexIO Control Register (CTRL)	32	RW	0000_0000h
Ch	Pin State Register (PIN)	32	R	0000_0000h
10h	Shifter Status Register (SHIFTSTAT)	32	RW	0000_0000h
14h	Shifter Error Register (SHIFTErr)	32	RW	0000_0000h
18h	Timer Status Register (TIMSTAT)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
20h	Shifter Status Interrupt Enable (SHIFTSIEN)	32	RW	0000_0000h
24h	Shifter Error Interrupt Enable (SHIFTEIEN)	32	RW	0000_0000h
28h	Timer Interrupt Enable Register (TIMIEN)	32	RW	0000_0000h
30h	Shifter Status DMA Enable (SHIFTSDEN)	32	RW	0000_0000h
38h	Timer Status DMA Enable (TIMERSDEN)	32	RW	0000_0000h
40h	Shifter State Register (SHIFTSTATE)	32	RW	0000_0000h
48h	Trigger Status Register (TRGSTAT)	32	RW	0000_0000h
4Ch	External Trigger Interrupt Enable Register (TRIGIEN)	32	RW	0000_0000h
50h	Pin Status Register (PINSTAT)	32	RW	0000_0000h
54h	Pin Interrupt Enable Register (PINIEN)	32	RW	0000_0000h
58h	Pin Rising Edge Enable Register (PINREN)	32	RW	0000_0000h
5Ch	Pin Falling Edge Enable Register (PINFEN)	32	RW	0000_0000h
60h	Pin Output Data Register (PINOUTD)	32	RW	0000_0000h
64h	Pin Output Enable Register (PINOUTE)	32	RW	0000_0000h
68h	Pin Output Disable Register (PINOUTDIS)	32	RW	0000_0000h
6Ch	Pin Output Clear Register (PINOUTCLR)	32	RW	0000_0000h
70h	Pin Output Set Register (PINOUTSET)	32	RW	0000_0000h
74h	Pin Output Toggle Register (PINOUTTOG)	32	RW	0000_0000h
80h - 9Ch	Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)	32	RW	0000_0000h
100h - 11Ch	Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)	32	RW	0000_0000h
200h - 21Ch	Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)	32	RW	0000_0000h
280h - 29Ch	Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)	32	RW	0000_0000h
300h - 31Ch	Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)	32	RW	0000_0000h
380h - 39Ch	Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)	32	RW	0000_0000h
400h - 41Ch	Timer Control N Register (TIMCTL0 - TIMCTL7)	32	RW	0000_0000h
480h - 49Ch	Timer Configuration N Register (TIMCFG0 - TIMCFG7)	32	RW	0000_0000h
500h - 51Ch	Timer Compare N Register (TIMCMP0 - TIMCMP7)	32	RW	0000_0000h
680h - 69Ch	Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
700h - 71Ch	Shifter Buffer N Half Word Swapped Register (SHIFTBUFHWS0 - SHIFTBUFHWS7)	32	RW	0000_0000h
780h - 79Ch	Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)	32	RW	0000_0000h
800h - 81Ch	Shifter Buffer N Odd Even Swapped Register (SHIFTBUFOES0 - SHIFTBUFOES7)	32	RW	0000_0000h
880h - 89Ch	Shifter Buffer N Even Odd Swapped Register (SHIFTBUFEOS0 - SHIFTBUFEOS7)	32	RW	0000_0000h
900h - 91Ch	Shifter Buffer N Halfword Byte Swapped Register (SHIFTBUFHBS0 - SHIFTBUFHBS7)	32	RW	0000_0000h

51.6.1.2 Version ID Register (VERID)

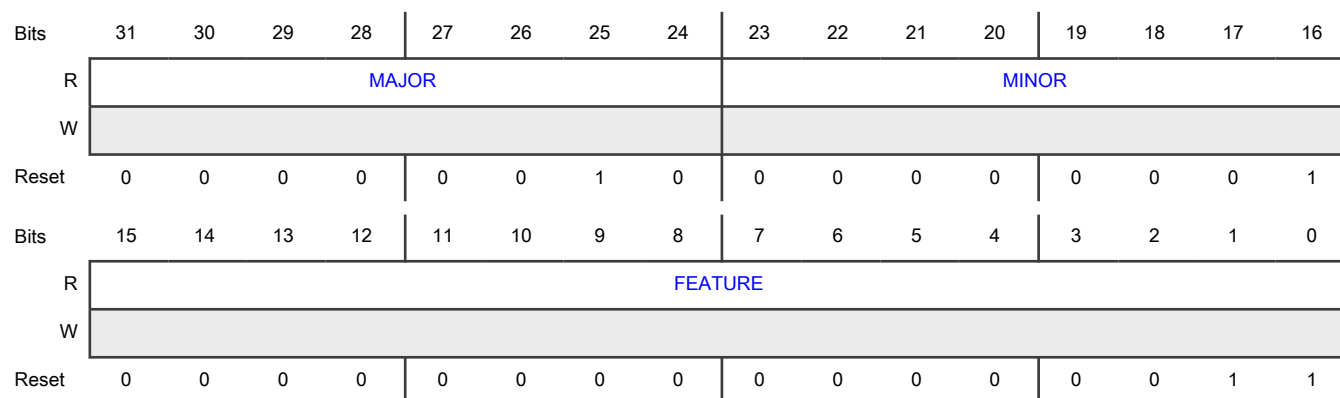
Offset

Register	Offset
VERID	0h

Function

Contains the version of the FlexIO.

Diagram



Fields

Field	Function
31-24	Major Version Number

Table continues on the next page...

Table continued from the previous page...

Field	Function
MAJOR	This read only field returns the major version number for the module specification.
23-16 MINOR	Minor Version Number This read only field returns the minor version number for the module specification.
15-0 FEATURE	Feature Specification Number This read only field returns the feature set number. 0000_0000_0000_0000b - Standard features implemented. 0000_0000_0000_0001b - Supports state, logic and parallel modes. 0000_0000_0000_0010b - Supports pin control registers. 0000_0000_0000_0011b - Supports state, logic and parallel modes; plus pin control registers.

51.6.1.3 Parameter Register (PARAM)

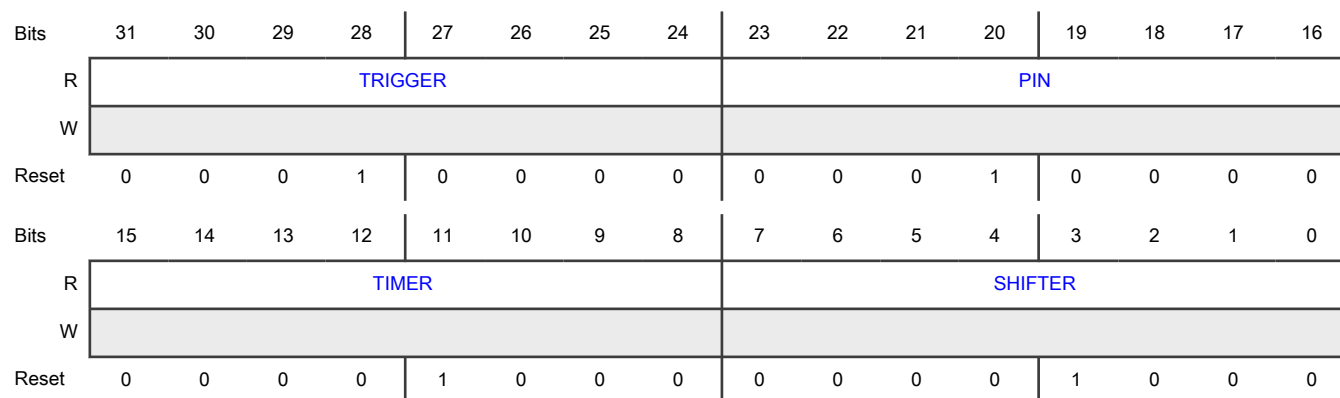
Offset

Register	Offset
PARAM	4h

Function

Contains the number of shifters, timers, pins, and triggers.

Diagram



Fields

Field	Function
31-24	Trigger Number

Table continues on the next page...

Table continued from the previous page...

Field	Function
TRIGGER	Number of external triggers implemented.
23-16 PIN	Pin Number Number of Pins implemented.
15-8 TIMER	Timer Number Number of Timers implemented.
7-0 SHIFTER	Shifter Number Number of Shifters implemented.

51.6.1.4 FlexIO Control Register (CTRL)

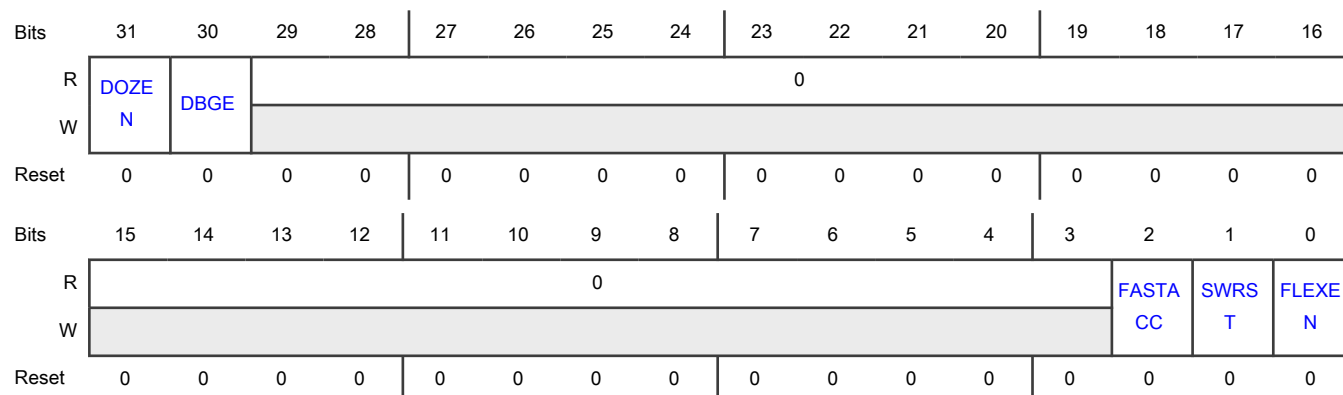
Offset

Register	Offset
CTRL	8h

Function

Control register.

Diagram



Fields

Field	Function
31 DOZEN	Doze Enable Disables FlexIO operation in Doze modes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - FlexIO enabled in Doze modes. 1b - FlexIO disabled in Doze modes.
30 DBGE	Debug Enable Enables FlexIO operation in Debug mode. 0b - FlexIO is disabled in debug modes. 1b - FlexIO is enabled in debug modes
29-3 —	Reserved
2 FASTACC	Fast Access Enables fast register accesses to FlexIO registers, but requires the FlexIO functional clock to be at least equal to the frequency of the bus clock. 0b - Configures for normal register accesses to FlexIO 1b - Configures for fast register accesses to FlexIO
1 SWRST	Software Reset The FlexIO Control Register is not affected by the software reset, all other logic in the FlexIO is affected by the software reset and all other register accesses are ignored until this bit is cleared. This register bit remains set until cleared by software, and the reset has cleared in the FlexIO clock domain. 0b - Software reset is disabled 1b - Software reset is enabled, all FlexIO registers except the Control Register are reset.
0 FLEXEN	FlexIO Enable Determines if the FlexIO is enabled or disabled. 0b - FlexIO module is disabled. 1b - FlexIO module is enabled.

51.6.1.5 Pin State Register (PIN)

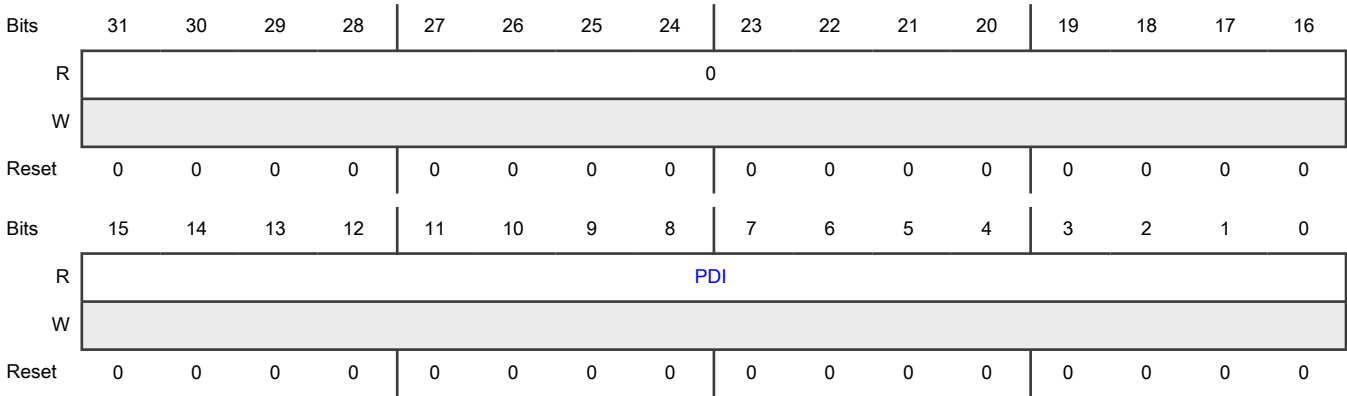
Offset

Register	Offset
PIN	Ch

Function

Reports status of the Pin Data Input.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 PDI	Pin Data Input Returns the input data on each of the FlexIO pins.

51.6.1.6 Shifter Status Register (SHIFTSTAT)

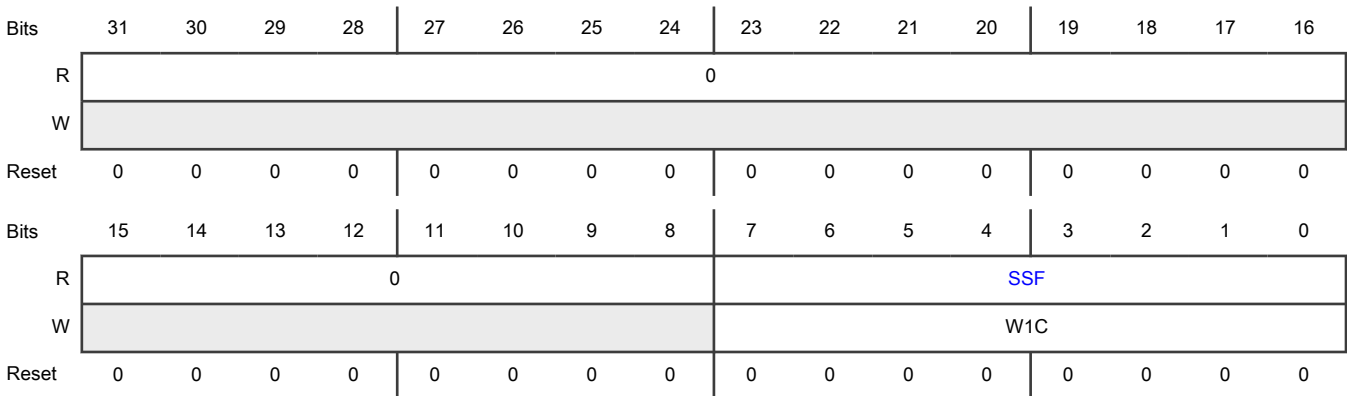
Offset

Register	Offset
SHIFTSTAT	10h

Function

Shifter status flags.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSF	Shifter Status Flag The shifter status flag is updated when one of the following events occurs: For SMOD=Receive, the status flag is set when SHIFTBUF has been loaded with data from Shifter (SHIFTBUF is full), and the status flag is cleared when SHIFTBUF register is read. For SMOD=Transmit, the status flag is set when SHIFTBUF data has been transferred to the Shifter (SHIFTBUF is empty) or when initially configured for SMOD=Transmit, and the status flag is cleared when the SHIFTBUF register is written. For SMOD=Match Store, the status flag is set when a match has occurred between SHIFTBUF and Shifter, and the status flag is cleared when the SHIFTBUF register is read. For SMOD=Match Continuous, returns the current match result between the SHIFTBUF and Shifter. The status flag cannot be cleared by reading SHIFTBUF. For SMOD=State, the status flag for a shifter sets when it is selected by the current state pointer. For SMOD=Logic, returns the current value of the programmable logic block output. The status flag can also be cleared by writing a logic one to the flag for all modes except Match Continuous/State /Logic. 0b - Status flag is clear. 1b - Status flag is set.

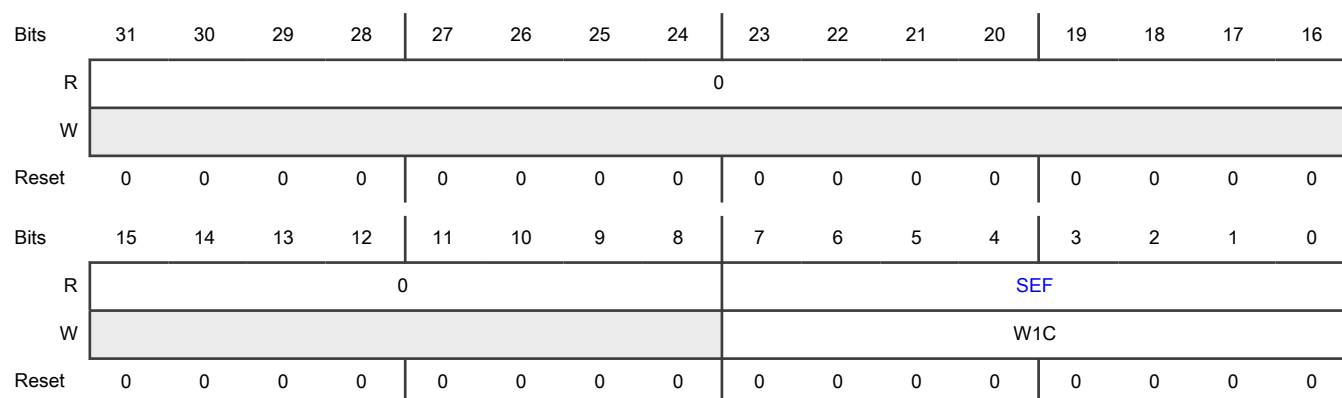
51.6.1.7 Shifter Error Register (SHIFTErr)

Offset

Register	Offset
SHIFTErr	14h

Function

This register reports shifter error flags. Write one to clear.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 SEF	Shifter Error Flags The shifter error flag is set when one of the following events occurs: For SMOD=Receive, indicates Shifter was ready to store new data into SHIFTBUF before the previous data was read from SHIFTBUF (SHIFTBUF Overrun), or indicates that the received start or stop bit does not match the expected value. For SMOD=Transmit, indicates Shifter was ready to load new data from SHIFTBUF before new data had been written into SHIFTBUF (SHIFTBUF Underrun). For SMOD=Match Store, indicates a match event occurred before the previous match data was read from SHIFTBUF (SHIFTBUF Overrun). For SMOD=Match Continuous, the error flag is set when a match has occurred between SHIFTBUF and Shifter. For SMOD=Logic, the error flag is set when the output of the programmable logic block has asserted. Can be cleared by writing logic one to the flag. For SMOD=Match Continuous, can also be cleared when the SHIFTBUF register is read. 0b - Shifter Error Flag is clear. 1b - Shifter Error Flag is set.

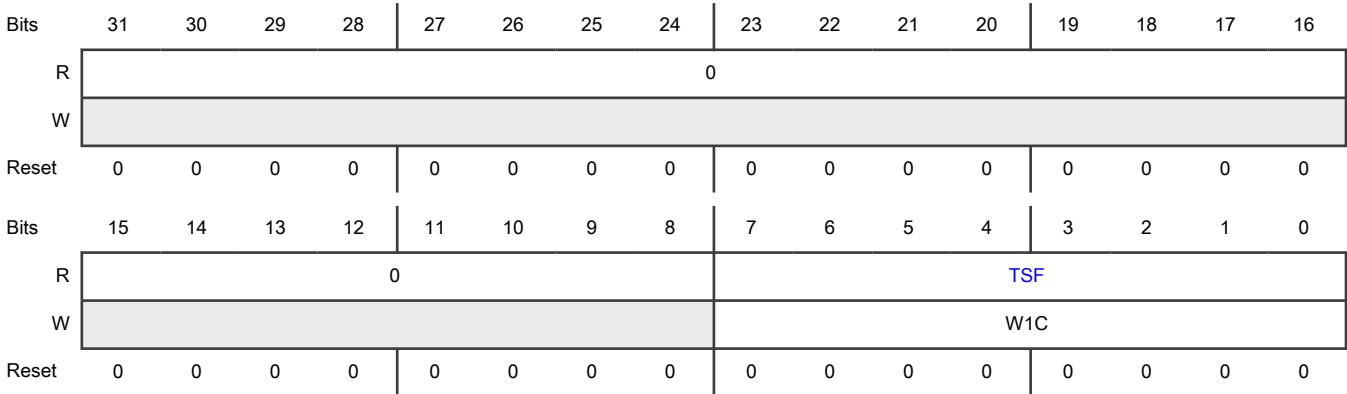
51.6.1.8 Timer Status Register (TIMSTAT)**Offset**

Register	Offset
TIMSTAT	18h

Function

This register reports timer status flags. Write one to clear.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 TSF	Timer Status Flags The timer status flag sets depending on the timer mode, and can be cleared by writing logic one to the flag. In 8-bit baud counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 8-bit high PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 16-bit counter mode, the timer status flag is set when the 16-bit counter equals zero and decrements. In 16-bit counter disable mode, the timer status flag is set when a timer disable event is detected. In 8-bit word counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 8-bit low PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 16-bit input capture mode, the timer status flag is set when a timer disable event is detected and the timer status flag is clear. In this mode, the timer compare register should only be read when the timer status flag is set. 0b - Timer Status Flag is clear. 1b - Timer Status Flag is set.

51.6.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

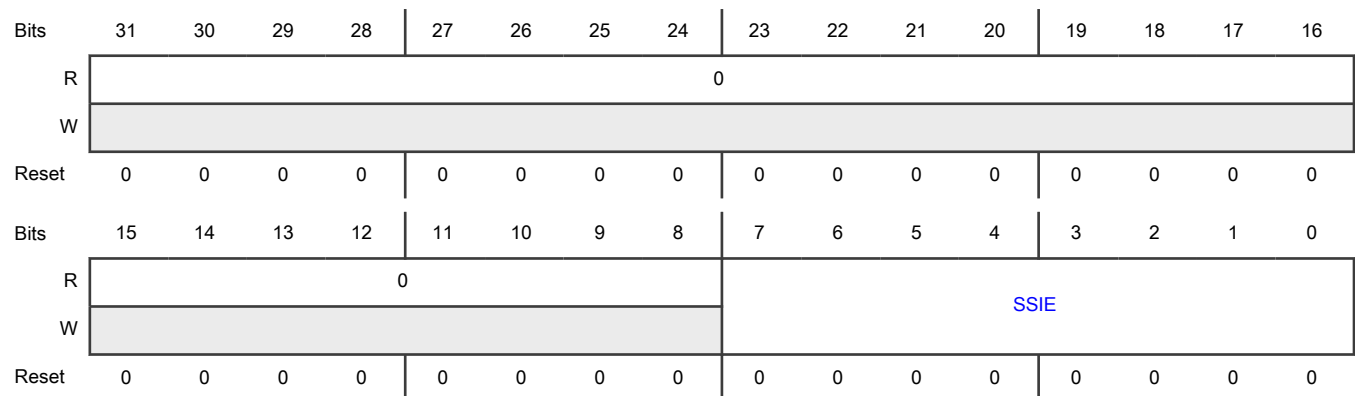
Offset

Register	Offset
SHIFTSIEN	20h

Function

Enables for Shifter Status Interrupts.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSIE	Shifter Status Interrupt Enable Enables interrupt generation when corresponding SSF is set. 0b - Shifter Status Flag interrupt disabled. 1b - Shifter Status Flag interrupt enabled.

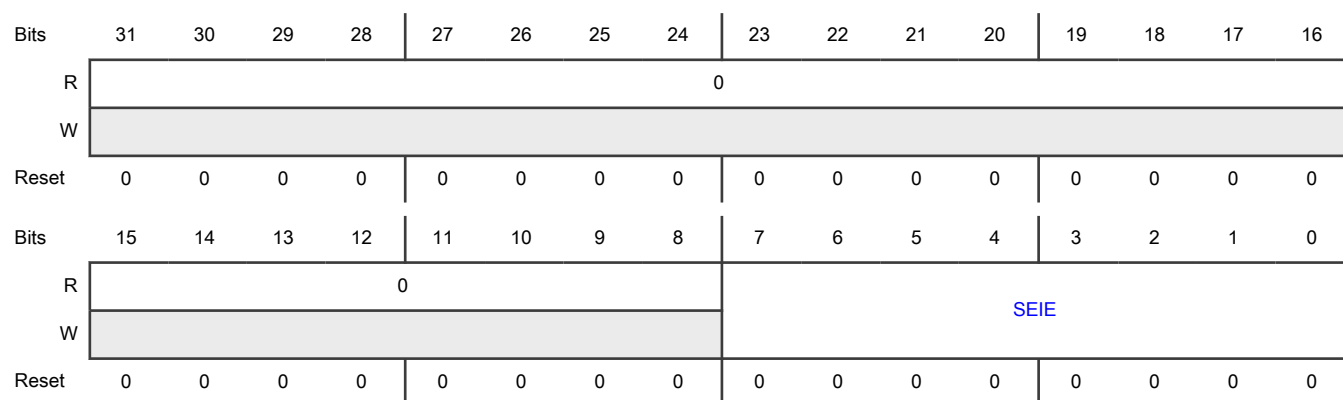
51.6.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

Offset

Register	Offset
SHIFTEIEN	24h

Function

Enables for Shifter Error Interrupts.

Diagram**Fields**

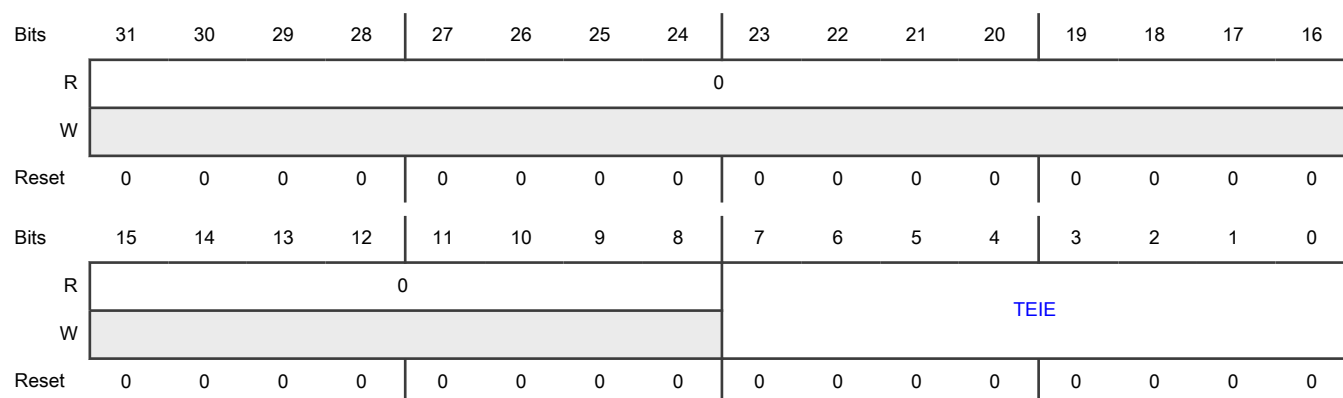
Field	Function
31-8 —	Reserved
7-0 SEIE	Shifter Error Interrupt Enable Enables interrupt generation when corresponding SEF is set. 0b - Shifter Error Flag interrupt disabled. 1b - Shifter Error Flag interrupt enabled.

51.6.1.11 Timer Interrupt Enable Register (TIMIEN)**Offset**

Register	Offset
TIMIEN	28h

Function

Enables for Timer Status Interrupts.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 TEIE	Timer Status Interrupt Enable Enables interrupt generation when corresponding TSF is set. 0b - Timer Status Flag interrupt is disabled. 1b - Timer Status Flag interrupt is enabled.

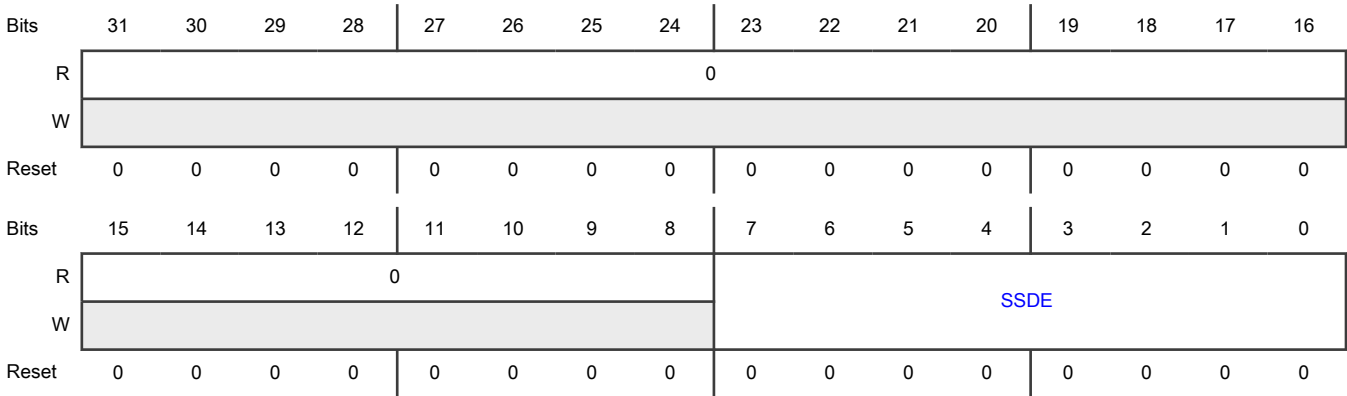
51.6.1.12 Shifter Status DMA Enable (SHIFTSDEN)**Offset**

Register	Offset
SHIFTSDEN	30h

Function

Enables for Shifter DMA requests.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 SSDE	Shifter Status DMA Enable Enables DMA request generation when corresponding SSF is set. 0b - Shifter Status Flag DMA request is disabled. 1b - Shifter Status Flag DMA request is enabled.

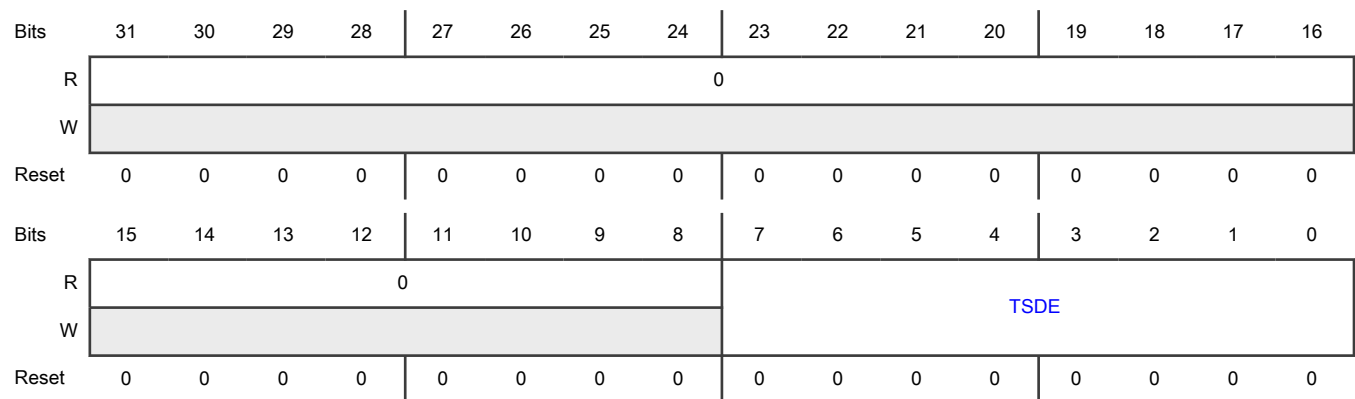
51.6.1.13 Timer Status DMA Enable (TIMERSDEN)

Offset

Register	Offset
TIMERSDEN	38h

Function

Enables for DMA requests when Timer Status Flag is set.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 TSDE	Timer Status DMA Enable Enables DMA request generation when corresponding TSF is set. When the Timer Status DMA request is enabled, reading or writing a Timer Compare Register clears the corresponding Timer Status Register. The DMA must therefore read or write the Timer Compare Register as part of the DMA transfer, otherwise the DMA request remains asserted. 0b - Timer Status Flag DMA request is disabled. 1b - Timer Status Flag DMA request is enabled.

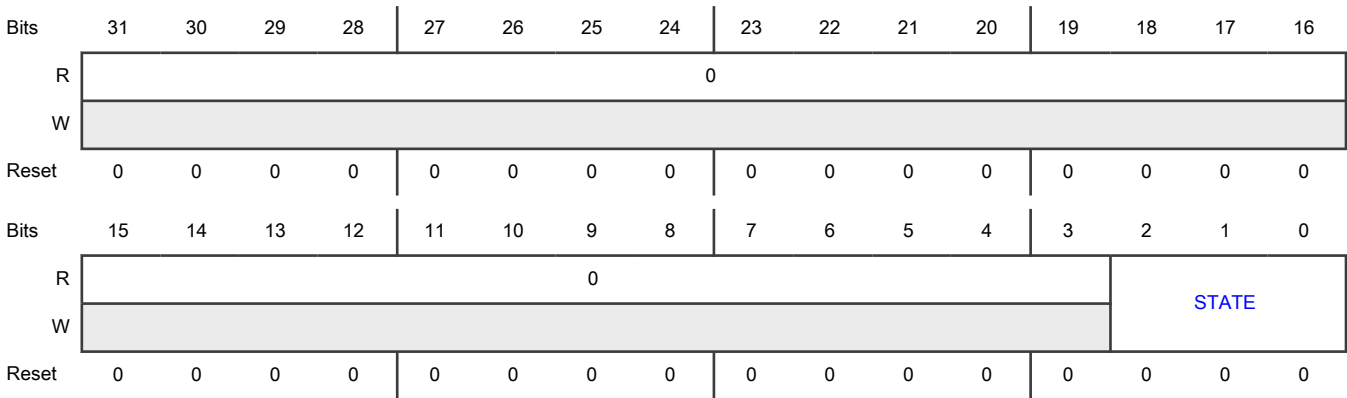
51.6.1.14 Shifter State Register (SHIFTSTATE)**Offset**

Register	Offset
SHIFTSTATE	40h

Function

Contains the pointer to keep track of the current shifter.

Diagram



Fields

Field	Function
31-3 —	Reserved
2-0 STATE	Current State Pointer The current state field maintains a pointer to keep track of the current Shifter (configured for State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the incorrect state returned. This bitfield is writable and the value written overrides the current state.

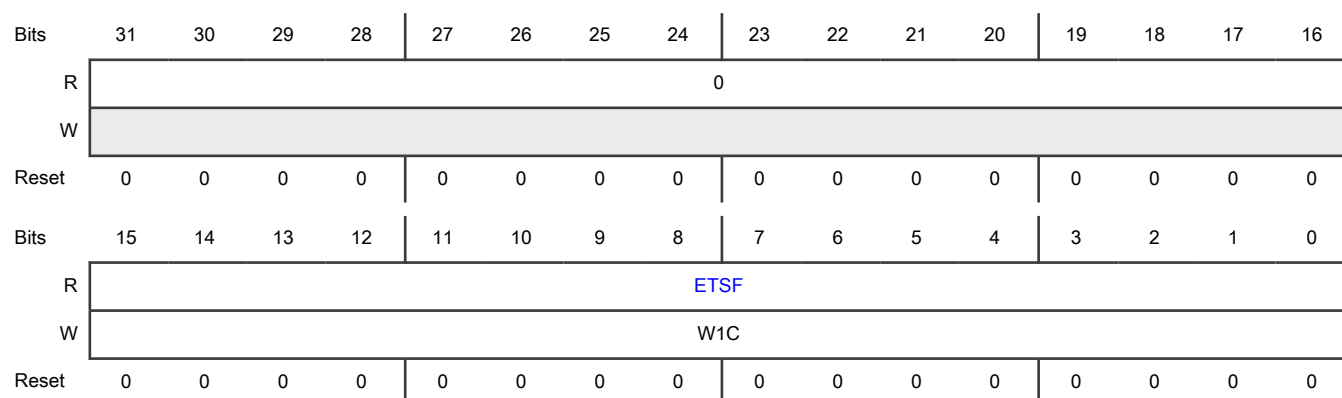
51.6.1.15 Trigger Status Register (TRGSTAT)

Offset

Register	Offset
TRGSTAT	48h

Function

External Trigger Status Flags. Write one to clear.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 ETSF	External Trigger Status Flags The external trigger status flag is set when a rising edge is detected on the corresponding external trigger input. 0b - External Trigger Status Flag is clear 1b - External Trigger Status Flag is set

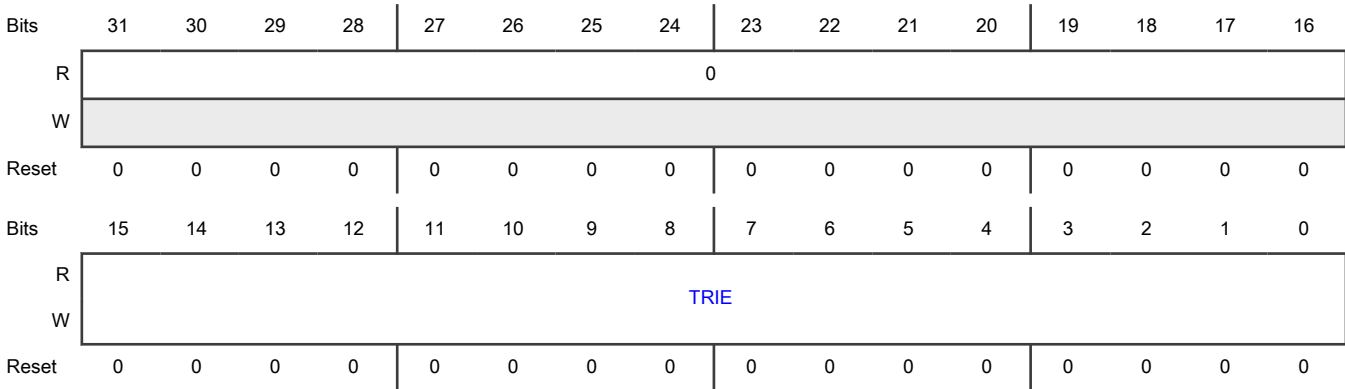
51.6.1.16 External Trigger Interrupt Enable Register (TRIGIEN)**Offset**

Register	Offset
TRIGIEN	4Ch

Function

Enables for external trigger interrupts.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 TRIE	External Trigger Interrupt Enable Enables interrupt generation when corresponding ETSF is set. 0b - External Trigger Status Flag interrupt is disabled. 1b - External Trigger Status Flag interrupt is enabled.

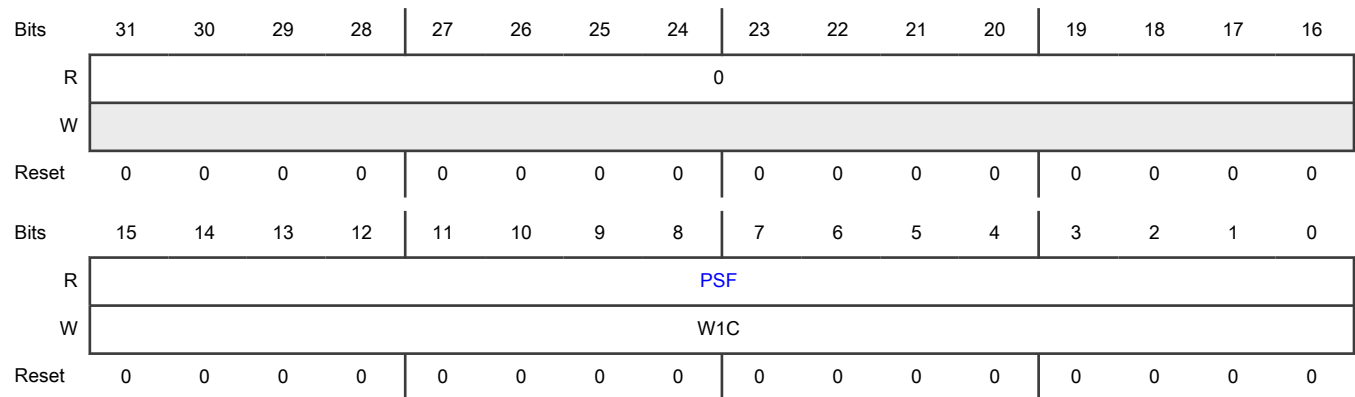
51.6.1.17 Pin Status Register (PINSTAT)

Offset

Register	Offset
PINSTAT	50h

Function

Pin Status Flags. Write one to clear.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 PSF	Pin Status Flags The pin status flag is set when a rising edge (if configured) or falling edge (if configured) is detected on the corresponding pin, as configured by pin. 0b - Pin Status Flag is clear 1b - Pin Status Flag is set

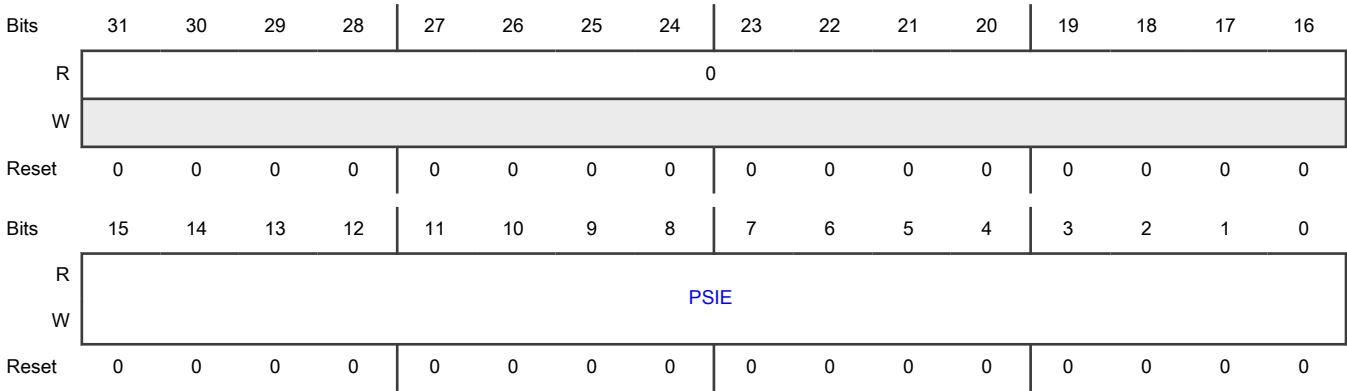
51.6.1.18 Pin Interrupt Enable Register (PINIEN)**Offset**

Register	Offset
PINIEN	54h

Function

Enables for pin status interrupts.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 PSIE	Pin Status Interrupt Enable Enables interrupt generation when corresponding PSF is set. 0b - Pin Status Flag interrupt is disabled. 1b - Pin Status Flag interrupt is enabled.

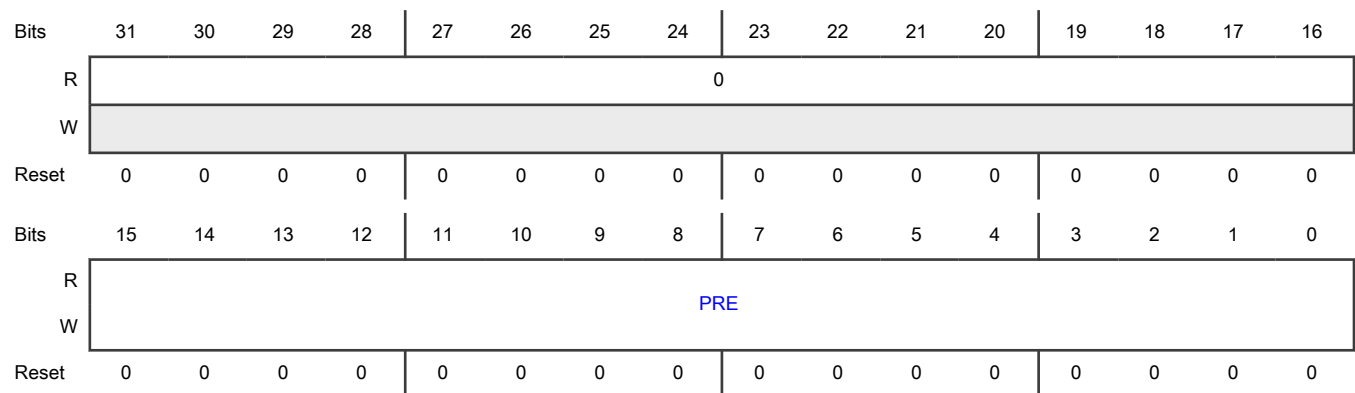
51.6.1.19 Pin Rising Edge Enable Register (PINREN)

Offset

Register	Offset
PINREN	58h

Function

Enables for pin status flag on rising edge.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 PRE	Pin Rising Edge When set, the pin status flag is set whenever a rising edge is detected on the pin. 0b - Pin Status Flag does not set when a pin rising edge is detected. 1b - Pin Status Flag does set when a pin rising edge is detected.

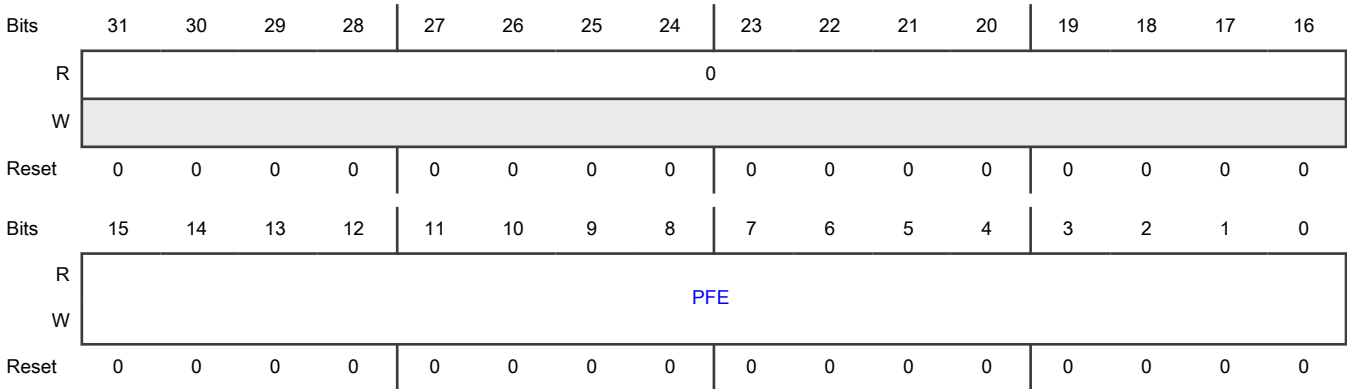
51.6.1.20 Pin Falling Edge Enable Register (PINFEN)**Offset**

Register	Offset
PINFEN	5Ch

Function

Enables for pin status flag on falling edge.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 PFE	Pin Falling Edge When set, the pin status flag is set whenever a falling edge is detected on the pin. 0b - Pin Status Flag does not set when a pin falling edge is detected. 1b - Pin Status Flag does set when a pin falling edge is detected.

51.6.1.21 Pin Output Data Register (PINOUTD)

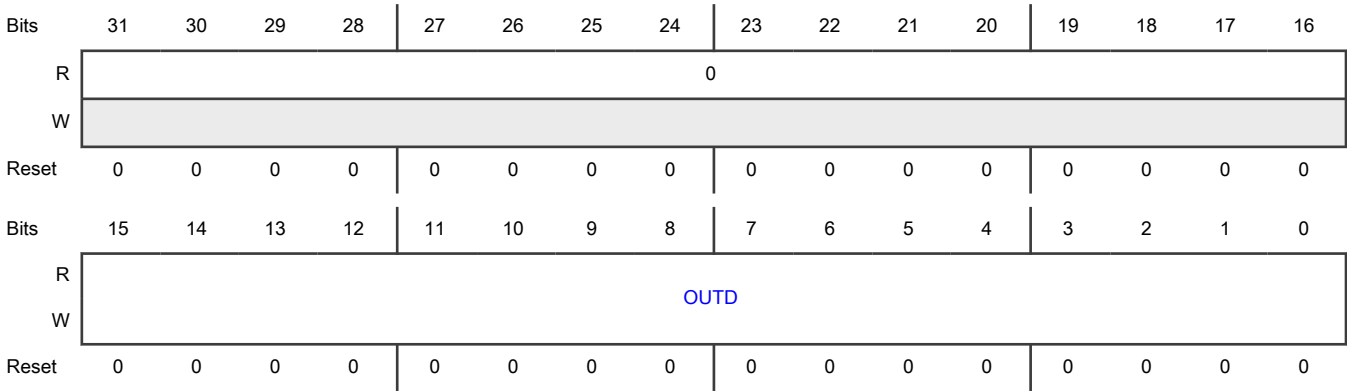
Offset

Register	Offset
PINOUTD	60h

Function

Data Out when direct pin output in enabled.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 OUTD	Output Data Configures the value driven on the corresponding pin when direct pin output is enabled. 0b - Logic zero is driven on the pin when direct pin output is enabled. 1b - Logic one is driven on the pin when direct pin output is enabled.

51.6.1.22 Pin Output Enable Register (PINOUTE)

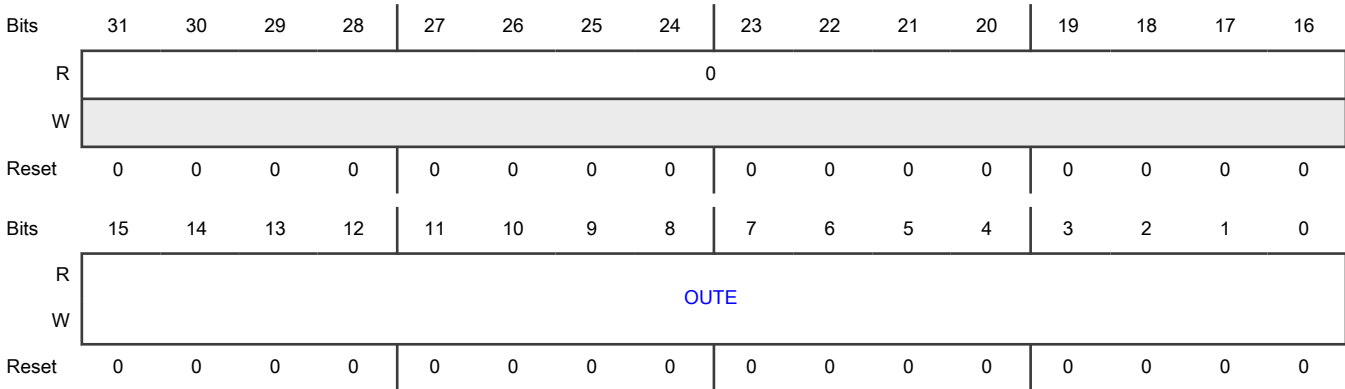
Offset

Register	Offset
PINOUTE	64h

Function

Enables for pin output.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 OUTE	Output Enable Enables direct output on the corresponding pin. 0b - Pin is controlled by timer/shifter configuration. 1b - Pin is an output and driven with value of PINOUTD register.

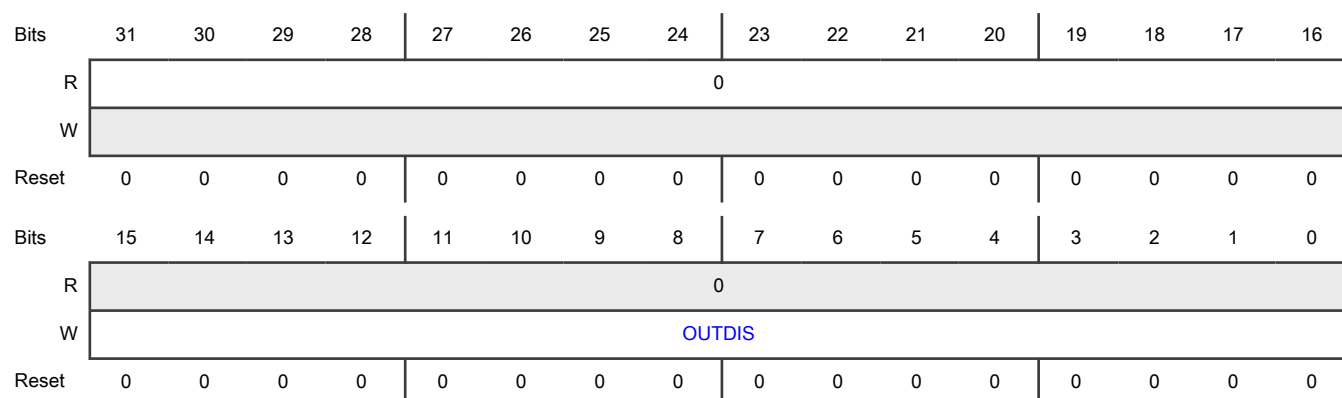
51.6.1.23 Pin Output Disable Register (PINOUTDIS)

Offset

Register	Offset
PINOUTDIS	68h

Function

Disable for pin output.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 OUTDIS	Output Disable Configures corresponding pins to disable direct output. 0b - No effect. 1b - Corresponding PINOUTD and PINOUTE bits are cleared.

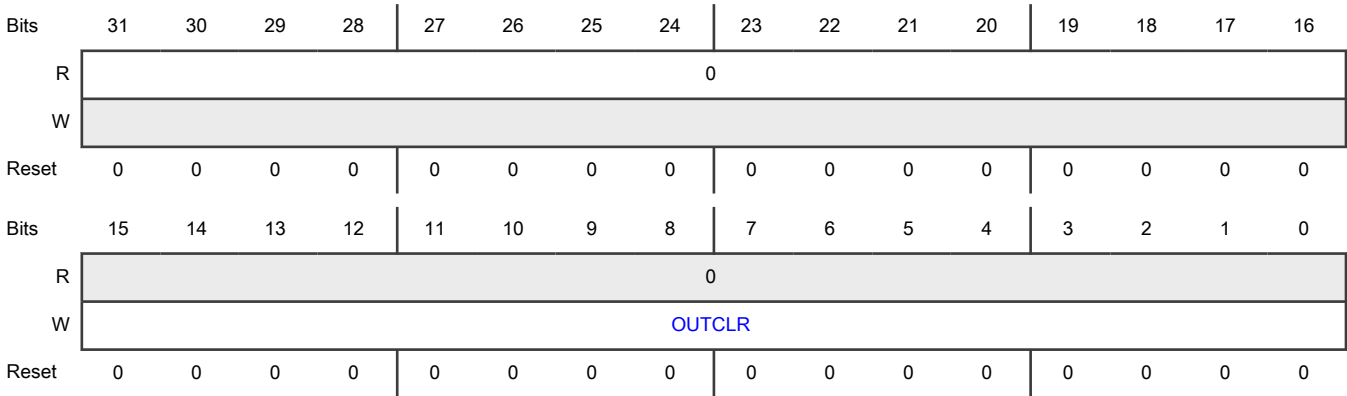
51.6.1.24 Pin Output Clear Register (PINOUTCLR)**Offset**

Register	Offset
PINOUTCLR	6Ch

Function

Clears pin output.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 OUTCLR	Output Clear Configures corresponding pins to output zero. 0b - No effect. 1b - Corresponding PINOUTD is cleared and PINOUTE is set.

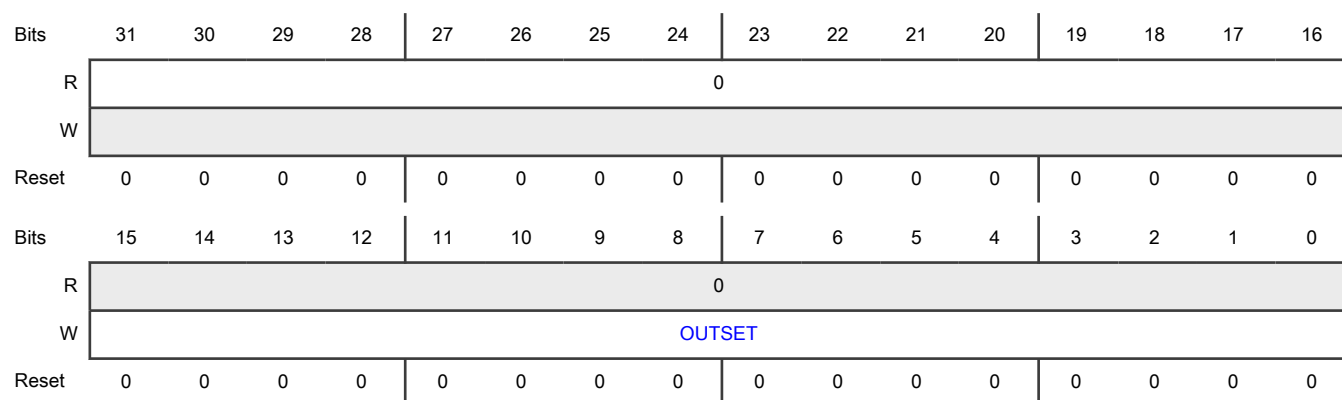
51.6.1.25 Pin Output Set Register (PINOUTSET)

Offset

Register	Offset
PINOUTSET	70h

Function

sets pin output.

Diagram**Fields**

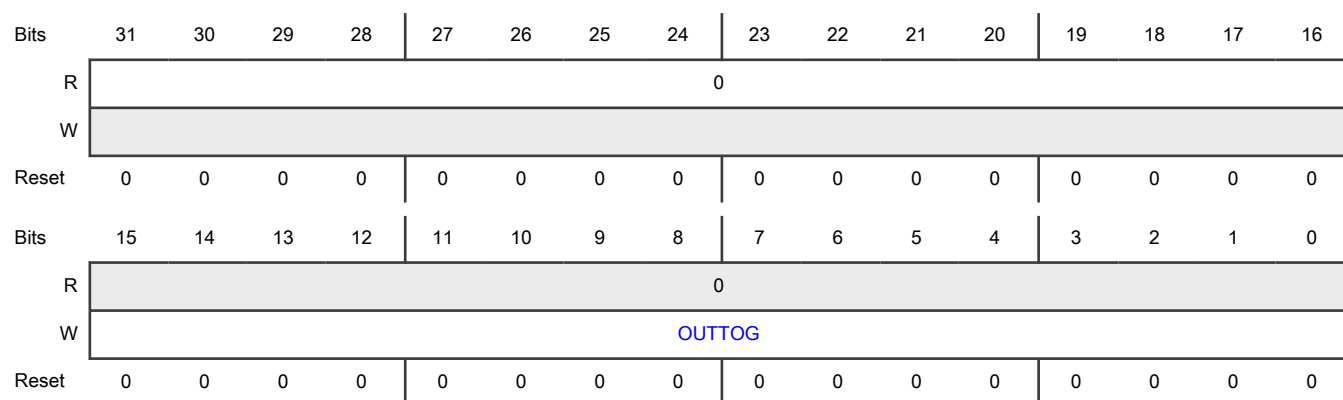
Field	Function
31-16 —	Reserved
15-0 OUTSET	Output Set Configures corresponding pins to output logic one. 0b - No effect. 1b - Corresponding PINOUTD is set and PINOUTE is set.

51.6.1.26 Pin Output Toggle Register (PINOUTTOG)**Offset**

Register	Offset
PINOUTTOG	74h

Function

Toggles pin output.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 OUTTOG	Output Toggle Configures corresponding pins to toggle. 0b - No effect. 1b - Corresponding PINOUTD is inverted and PINOUTE is set.

51.6.1.27 Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)**Offset**

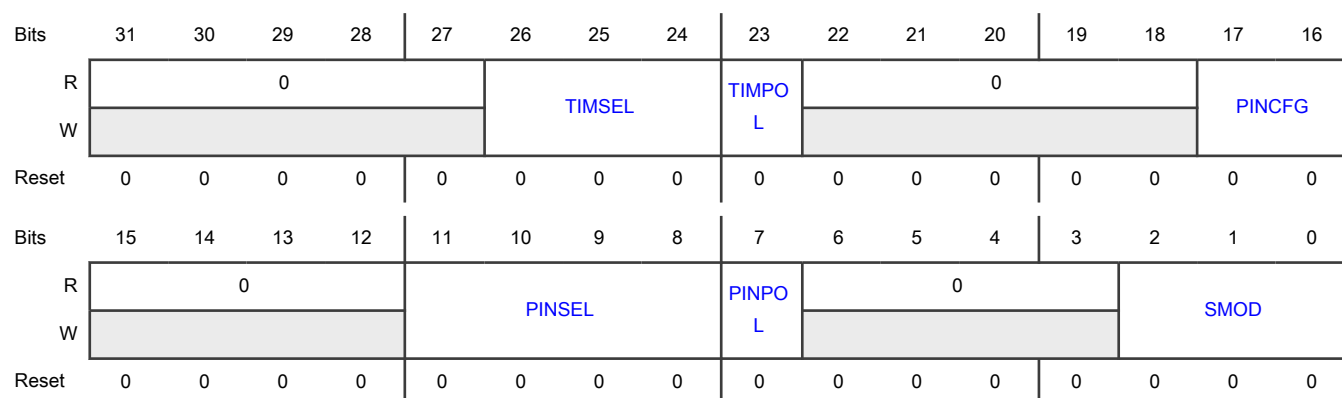
For a = 0 to 7:

Register	Offset
SHIFTCTL _a	80h + (a × 4h)

Function

Control for Shift Register N

Diagram



Fields

Field	Function
31-27 —	Reserved
26-24 TIMSEL	Timer Select Selects which Timer is used for controlling the logic/shift register and generating the Shift clock. TIMSEL=i selects TIMERi.
23 TIMPOL	Timer Polarity Determines the shift occurs on the positive edge or negative edge of the shift clock. 0b - Shift on posedge of Shift clock 1b - Shift on negedge of Shift clock
22-18 —	Reserved
17-16 PINCFG	Shifter Pin Configuration For pins configured as an output (PINCFG=11), this field takes effect when the register is written. NOTE When initially configuring PINCFG=0x11, FlexIO may briefly drive the pin low. To avoid this, software can configure PINCFG=0x10 along with the rest of the control register and then perform a subsequent write to set PINCFG=0x11. 00b - Shifter pin output disabled 01b - Shifter pin open drain or bidirectional output enable 10b - Shifter pin bidirectional output data 11b - Shifter pin output
15-12	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
11-8 PINSEL	Shifter Pin Select Selects which pin is used by the Shifter input or output. PINSEL=i selects the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written.
7 PINPOL	Shifter Pin Polarity For pins configured as an output (PINCFG=11), this field takes effect when the register is written. 0b - Pin is active high 1b - Pin is active low
6-3 —	Reserved
2-0 SMOD	Shifter Mode Configures the mode of the Shifter. 000b - Disabled. 001b - Receive mode. Captures the current Shifter content into the SHIFTBUF on expiration of the Timer. 010b - Transmit mode. Load SHIFTBUF contents into the Shifter on expiration of the Timer. 011b - Reserved. 100b - Match Store mode. Shifter data is compared to SHIFTBUF content on expiration of the Timer. 101b - Match Continuous mode. Shifter data is continuously compared to SHIFTBUF contents. 110b - State mode. SHIFTBUF contents are used for storing programmable state attributes. 111b - Logic mode. SHIFTBUF contents are used for implementing programmable logic look up table.

51.6.1.28 Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)

Offset

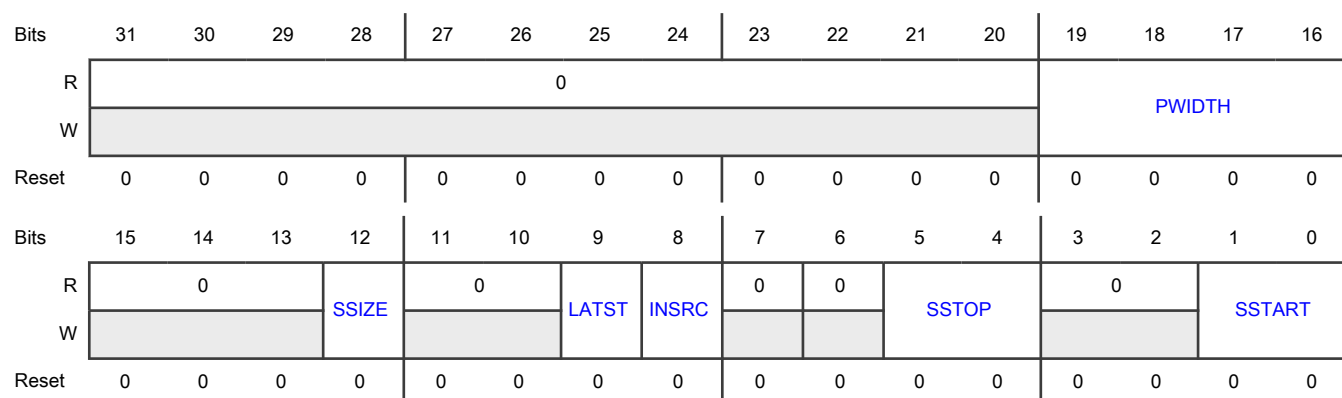
For a = 0 to 7:

Register	Offset
SHIFTCFGa	100h + (a × 4h)

Function

Contains configuration bit fields for Shifter Register N.

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 PWIDTH	Parallel Width For all Shifters, this register field configures the number of bits to be shifted on each Shift clock as follows: 1-bit shift for PWIDTH=0 2-bit shift for PWIDTH=1 4-bit shift for PWIDTH=2...3 8-bit shift for PWIDTH=4...7 16-bit shift for PWIDTH=8...15 For Shifters which support parallel transmit (SHIFTER0, SHIFTER4, ...) or parallel receive (SHIFTER3, SHIFTER7, ...), this register field, together with SHIFTCTL[PINSEL], also selects the pins to be driven or sampled on each Shift clock as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL] Shifters which do not support parallel transmit or parallel receive only support parallel shift when INSRC=1. For SMOD=State, this field is used to disable state outputs. See State Mode section for more detail.
15-13 —	Reserved
12 SSIZE	Shifter Size Configures the size of the Shift Registers. A 24-bit shift register shifts data into bits [23:0] only and does not update bits [31:24] during shift operations. When shift register is configured for 24-bit, configuring PWIDTH=8..15 performs a 12-bit shift and PWIDTH=16..31 performs a 24-bit shift. 0b - Shift register is 32-bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Shift register is 24-bit.
11-10 —	Reserved
9 LATST	Late Store Configures what happens when a receive or match shift register is configured to both shift and store on the same cycle. 0b - Shift register stores the pre-shift register state. 1b - Shift register stores the post-shift register state.
8 INSRC	Input Source Selects the input source for the shifter. Configuring INSRC=1 is not supported for the last shifter. 0b - Pin 1b - Shifter N+1 Output
7 —	Reserved
6 —	Reserved
5-4 SSTOP	Shifter Stop bit For SMOD =Transmit, this field allows automatic stop bit insertion if the selected timer has also enabled a stop bit. For SMOD =Receive or Match Store, this field allows automatic stop bit checking if the selected timer has also enabled a stop bit. For SMOD =State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD =Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00b - Stop bit disabled for transmitter/receiver/match store 01b - Stop bit disabled for transmitter/receiver/match store, receiver/match store will store receive data on the configured shift edge when timer in stop condition 10b - Transmitter outputs stop bit value 0 on store, receiver/match store sets error flag if stop bit is not 0, receiver/match store will also store receive data on the configured shift edge when timer in stop condition 11b - Transmitter outputs stop bit value 1 on store, receiver/match store sets error flag if stop bit is not 1, receiver/match store will also store receive data on the configured shift edge when timer in stop condition
3-2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 SSTART	Shifter Start bit For SMOD=Transmit, this field allows automatic start bit insertion if the selected timer has also enabled a start bit. For SMOD=Receive or Match Store, this field allows automatic start bit checking if the selected timer has also enabled a start bit. For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD=Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on enable 01b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on first shift 10b - Transmitter outputs start bit value 0 before loading data on first shift, receiver/match store sets error flag if start bit is not 0 11b - Transmitter outputs start bit value 1 before loading data on first shift, receiver/match store sets error flag if start bit is not 1

51.6.1.29 Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)

Offset

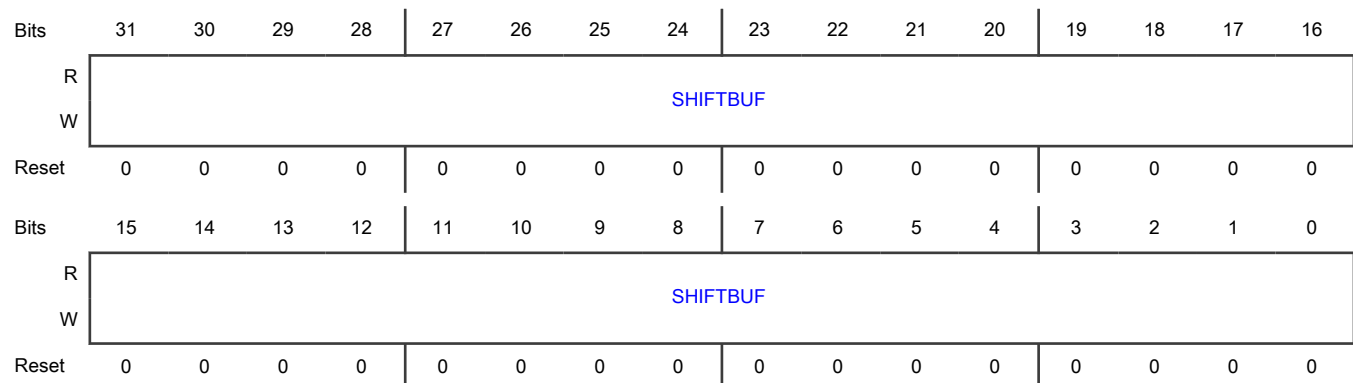
For a = 0 to 7:

Register	Offset
SHIFTBUFa	200h + (a × 4h)

Function

Contains Shift buffer data.

Diagram



Fields

Field	Function
31-0 SHIFTBUF	Shift Buffer Shift buffer data is used for a variety of functions depending on the SMOD setting: For SMOD=Receive, Shifter data is transferred into SHIFTBUF at the expiration of Timer. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available. For SMOD=Transmit, SHIFTBUF data is transferred into the Shifter before the Timer begins. For SMOD=Match Store, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). The Match is checked when the Timer expires and Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available. For SMOD=Match Continuous, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). For SMOD=Logic, SHIFTBUF[31:0] is used to implement a 5-input, 32-bit programmable logic look-up table. See Logic Mode section for more detail. For SMOD=State, SHIFTBUF[31:24] is used to drive the output value when this shifter is selected by the current state pointer and SHIFTBUF[23:0] is used to configure the value of the next state transition. See State Mode section for more detail.

51.6.1.30 Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)

Offset

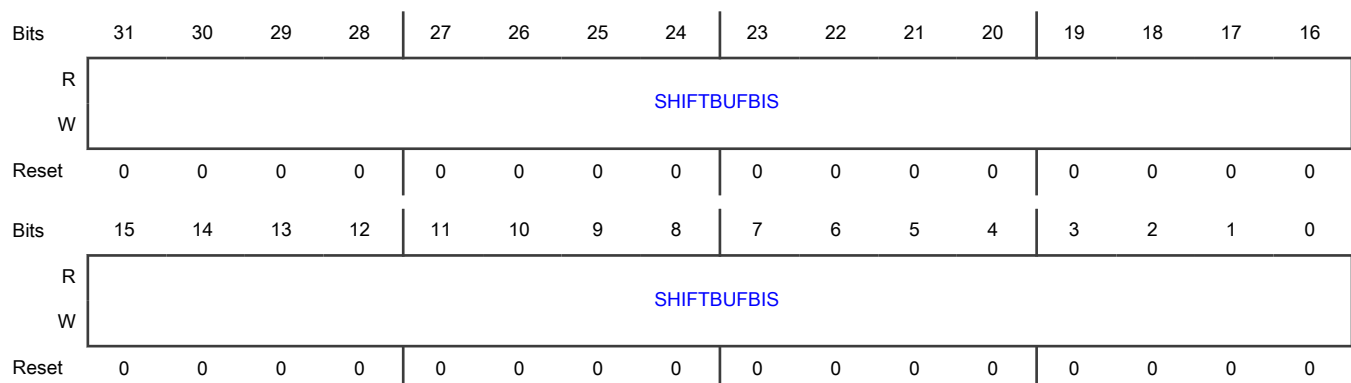
For a = 0 to 7:

Register	Offset
SHIFTBUFBISa	280h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register are bit swapped.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFBIS	Alias to SHIFTBUF register, except reads/writes to this register are bit swapped. Reads return SHIFTBUF[0:31].

51.6.1.31 Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)

Offset

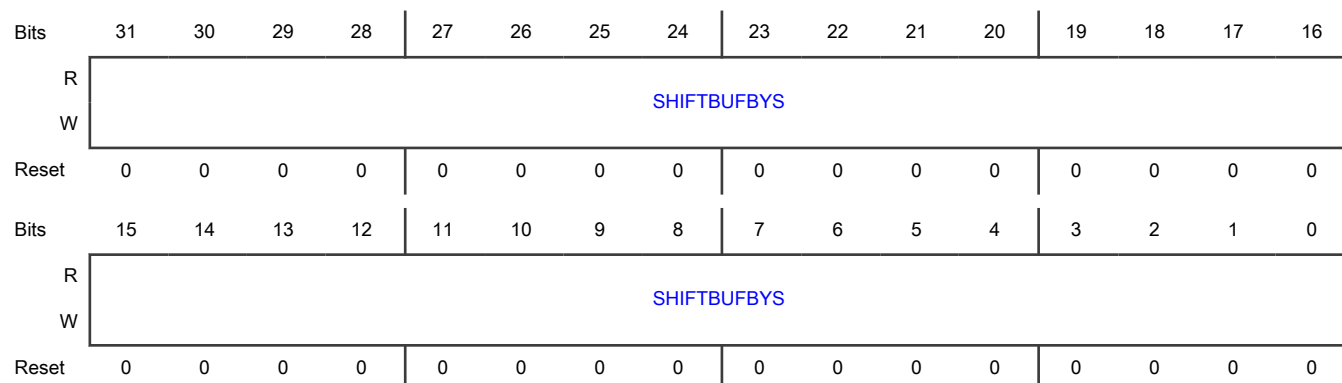
For a = 0 to 7:

Register	Offset
SHIFTBUFBYSa	300h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register are byte swapped.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFBYS	Alias to SHIFTBUF register, except reads/writes to this register are byte swapped. Reads return { SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24] }.

51.6.1.32 Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)

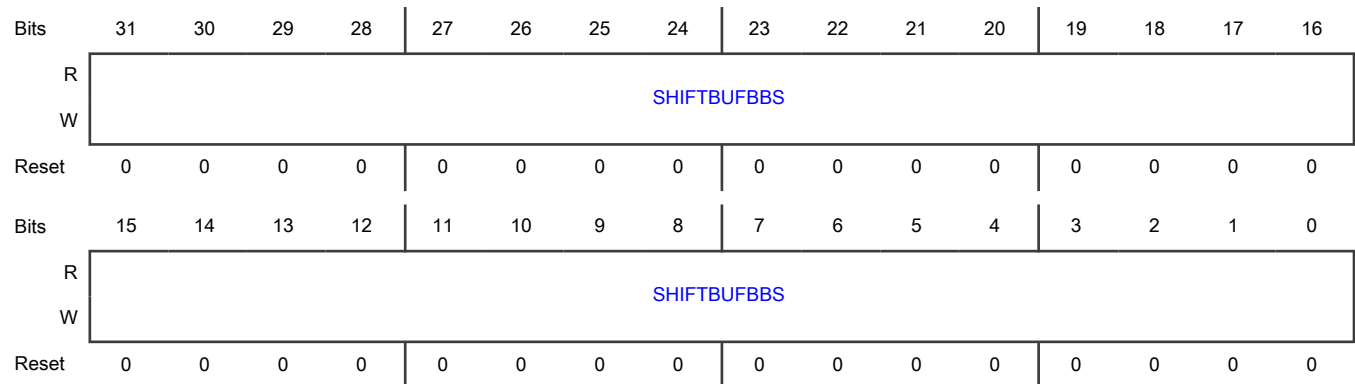
Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBBSa	380h + (a × 4h)

Function

Contains SHIFTBUF register data, but is bit swapped within each byte.

Diagram**Fields**

Field	Function
31-0	Shift Buffer
SHIFTBUFBBBS	Alias to SHIFTBUF register, except reads/writes to this register are bit swapped within each byte. Reads return { SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7] }.

51.6.1.33 Timer Control N Register (TIMCTL0 - TIMCTL7)**Offset**

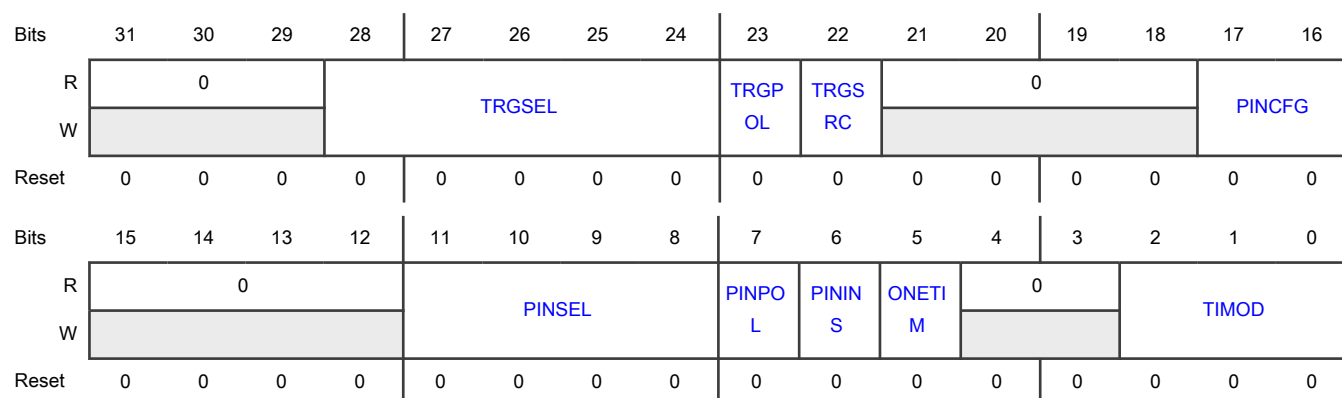
For a = 0 to 7:

Register	Offset
TIMCTLa	400h + (a × 4h)

Function

Timer Control for Timer N.

Diagram



Fields

Field	Function
31-29 —	Reserved
28-24 TRGSEL	Trigger Select The valid values for TRGSEL depends on the FLEXIO_PARAM register. - When TRGSRC = 1, the valid values for N depends on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register. - When TRGSRC = 0, the valid values for N depends on TRIGGER field in FLEXIO_PARAM register. Refer to the chip-specific FlexIO information for external trigger selection. NOTE For a pin, N=0 to 15. For a Shifter, N=0 to 7. For a Timer, N=0 to 7. When TRGSRC = 0 the trigger selection is configured as follows: - N - External trigger N input When TRGSRC = 1 the internal trigger can be configured to select an input pin as follows: 2*N - Pin N input When TRGSRC = 1 the internal trigger can be configured to select a shifter/timer signal as follows: 4*N+1 - Shifter N status flag 4*N+3 - Timer N trigger output In other words, the expanded internal trigger selection (TRGSRC = 1) is as follows: 0000 = Pin 0 0001 = Shifter 0 Flag 0010 = Pin 1 0011 = Timer 0 Trigger

Table continues on the next page...

Table continued from the previous page...

Field	Function
	• 0100 = Pin 2 • 0101 = Shifter 1 Flag • 0110 = Pin 3 • 0111 = Timer 1 Trigger • ... • This continues up to the Pin 15, Shifter 7 and Timer 7.
23 TRGPOL	Trigger Polarity Determines if the trigger is active high or active low. 0b - Trigger active high 1b - Trigger active low
22 TRGSRC	Trigger Source Determines if the trigger source is external or internal. 0b - External trigger selected 1b - Internal trigger selected
21-18 —	Reserved
17-16 PINCFG	Timer Pin Configuration Configures the direction of the Timer pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written. NOTE When initially configuring PINCFG=0x11, FlexIO may briefly drive the pin low. To avoid this, software can configure PINCFG=0x10 along with the rest of the control register and then perform a subsequent write to set PINCFG=0x11. 00b - Timer pin output disabled 01b - Timer pin open drain or bidirectional output enable 10b - Timer pin bidirectional output data 11b - Timer pin output
15-12 —	Reserved
11-8 PINSEL	Timer Pin Select Selects which pin is used by the Timer input or output. PINSEL=i selects the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 PINPOL	Timer Pin Polarity For pins configured as an output (PINCFG=11), this field takes effect when the register is written. 0b - Pin is active high 1b - Pin is active low
6 PININS	Timer Pin Input Select When set, the timer input pin is a different pin from the timer output pin. PINSEL must select an even numbered pin when this bit is set, so the output pin is even numbered and input pin is odd numbered. 0b - Timer pin input and output are selected by PINSEL. 1b - Timer pin input is selected by PINSEL+1, timer pin output remains selected by PINSEL.
5 ONETIM	Timer One Time Operation When set, configures the timer to perform a single enable/disable iteration, after which software must clear the timer status flag for the timer to be enabled again. 0b - The timer enable event is generated as normal. 1b - The timer enable event is blocked unless timer status flag is clear.
4-3 —	Reserved
2-0 TIMOD	Timer Mode In 8-bit baud counter mode, the lower 8-bits of the counter and compare register are used to configure the baud rate of the timer shift clock, and the upper 8-bits are used to configure the shifter bit count. In 8-bit PWM high mode, the lower 8-bits of the counter and compare register are used to configure the high period of the timer shift clock, and the upper 8-bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal. In 16-bit counter mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. In 16-bit counter disable mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. In 8-bit word counter mode, the lower 8-bits of the counter and compare register are used to configure the shifter bit count, and the upper 8-bits are used to configure the shifter word count. In 8-bit PWM low mode, the lower 8-bits of the counter and compare register are used to configure the low period of the timer shift clock and the upper 8-bits are used to configure the high period of the timer shift clock. The shifter bit count is configured using another timer or external signal. In 16-bit input capture mode, the inverted value of 16-bit counter is latched into the compare register when a timer counter disable condition is detected (as configured by TIMDIS). This also sets the timer status flag. The timer counter is immediately restarted from 0xFFFF 000b - Timer Disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - Dual 8-bit counters baud mode. 010b - Dual 8-bit counters PWM high mode. 011b - Single 16-bit counter mode. 100b - Single 16-bit counter disable mode. 101b - Dual 8-bit counters word mode. 110b - Dual 8-bit counters PWM low mode. 111b - Single 16-bit input capture mode.

51.6.1.34 Timer Configuration N Register (TIMCFG0 - TIMCFG7)

Offset

For a = 0 to 7:

Register	Offset
TIMCFGa	480h + (a × 4h)

Function

The options to enable or disable the timer using the Timer N-1 enable or disable are reserved when N is evenly divisible by 4 (eg: Timer 0).

NOTE

The pin and trigger level/edges specified below refer to the signal state after being modified by the PINPOL or TRGPOL setting in the TIMCTL register. For example, "Trigger low" means a trigger is actually at logic level 1 if the TRGPOL is set to 1 (active low).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TIMOUT				0	TIMDEC				0	TIMRST	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TIMDIS				0	TIMENA				0	TSTOP				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	Reserved
25-24 TIMOUT	Timer Output Configures the initial state of the Timer Output and whether it is affected by the Timer reset. 00b - Timer output is logic one when enabled and is not affected by timer reset 01b - Timer output is logic zero when enabled and is not affected by timer reset 10b - Timer output is logic one when enabled and on timer reset 11b - Timer output is logic zero when enabled and on timer reset
23 —	Reserved
22-20 TIMDEC	Timer Decrement Configures the source of the Timer decrement and the source of the Shift clock. 000b - Decrement counter on FlexIO clock, Shift clock equals Timer output. 001b - Decrement counter on Trigger input (both edges), Shift clock equals Timer output. 010b - Decrement counter on Pin input (both edges), Shift clock equals Pin input. 011b - Decrement counter on Trigger input (both edges), Shift clock equals Trigger input. 100b - Decrement counter on FlexIO clock divided by 16, Shift clock equals Timer output. 101b - Decrement counter on FlexIO clock divided by 256, Shift clock equals Timer output. 110b - Decrement counter on Pin input (rising edge), Shift clock equals Pin input. 111b - Decrement counter on Trigger input (rising edge), Shift clock equals Trigger input.
19 —	Reserved
18-16 TIMRST	Timer Reset Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset only resets the lower 8-bits that configure the baud rate. In all other modes, the timer reset resets the full 16-bits of the counter. 000b - Timer never reset 001b - Timer reset on Timer Output high. 010b - Timer reset on Timer Pin equal to Timer Output 011b - Timer reset on Timer Trigger equal to Timer Output 100b - Timer reset on Timer Pin rising edge 101b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
	110b - Timer reset on Trigger rising edge 111b - Timer reset on Trigger rising or falling edge
15 —	Reserved
14-12 TIMDIS	Timer Disable Configures the condition that causes the Timer to be disabled and stop decrementing. 000b - Timer never disabled 001b - Timer disabled on Timer N-1 disable 010b - Timer disabled on Timer compare (upper 8-bits match and decrement) 011b - Timer disabled on Timer compare (upper 8-bits match and decrement) and Trigger Low 100b - Timer disabled on Pin rising or falling edge 101b - Timer disabled on Pin rising or falling edge provided Trigger is high 110b - Timer disabled on Trigger falling edge 111b - Reserved
11 —	Reserved
10-8 TIMENA	Timer Enable Configures the condition that causes the Timer to be enabled and start decrementing. 000b - Timer always enabled 001b - Timer enabled on Timer N-1 enable 010b - Timer enabled on Trigger high 011b - Timer enabled on Trigger high and Pin high 100b - Timer enabled on Pin rising edge 101b - Timer enabled on Pin rising edge and Trigger high 110b - Timer enabled on Trigger rising edge 111b - Timer enabled on Trigger rising or falling edge
7-6 —	Reserved
5-4 TSTOP	Timer Stop Bit The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Stop bit disabled 01b - Stop bit is enabled on timer compare 10b - Stop bit is enabled on timer disable 11b - Stop bit is enabled on timer compare and timer disable
3-2 —	Reserved
1 TSTART	Timer Start Bit When start bit is enabled, configured shifters outputs the contents of the start bit when the timer is enabled and the timer counter reloads from the compare register on the first rising edge of the shift clock. 0b - Start bit disabled 1b - Start bit enabled
0 —	Reserved

51.6.1.35 Timer Compare N Register (TIMCMP0 - TIMCMP7)

Offset

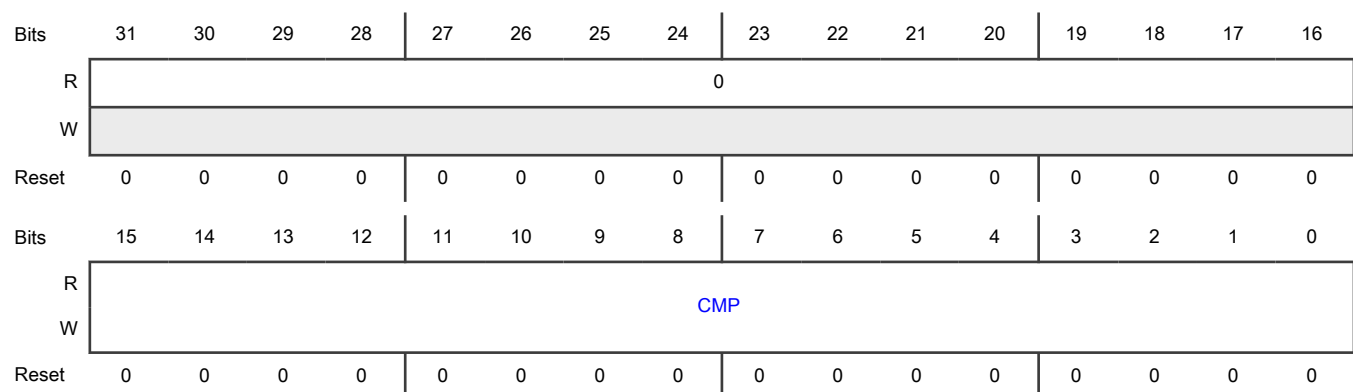
For a = 0 to 7:

Register	Offset
TIMCMPa	500h + (a × 4h)

Function

Contains the timer compare value.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-0 CMP	Timer Compare Value The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset and when the timer decrements down to zero. In 8-bit baud counter mode, the lower 8-bits configure the baud rate divider equal to $(CMP[7:0] + 1) * 2$. The upper 8-bits configure the number of bits in each word equal to $(CMP[15:8] + 1) / 2$. In 8-bit PWM high mode, the lower 8-bits configure the high period of the output to $(CMP[7:0] + 1)$ and the upper 8-bits configure the low period of the output to $(CMP[15:8] + 1)$. In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal $(CMP[15:0] + 1) * 2$. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word equal to $(CMP[15:0] + 1) / 2$. In 16-bit counter disable mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal $(CMP[15:0] + 1) * 2$. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word equal to $(CMP[15:0] + 1) / 2$. In 8-bit word counter mode, the lower 8-bits configure the number of bits in each word equal to $(CMP[7:0] + 1) / 2$. The upper 8-bits configure the number of words to transfer equal to $(CMP[15:8] + 1) / 2$. In 8-bit PWM low mode, the lower 8-bits configure the low period of the output to $(CMP[7:0] + 1)$ and the upper 8-bits configure the high period of the output to $(CMP[15:8] + 1)$. In 16-bit input capture mode, the compare register is updated with the inverse of the timer counter value whenever the timer status flag is set. The timer compare register should only be read when the timer status flag is set.

51.6.1.36 Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)

Offset

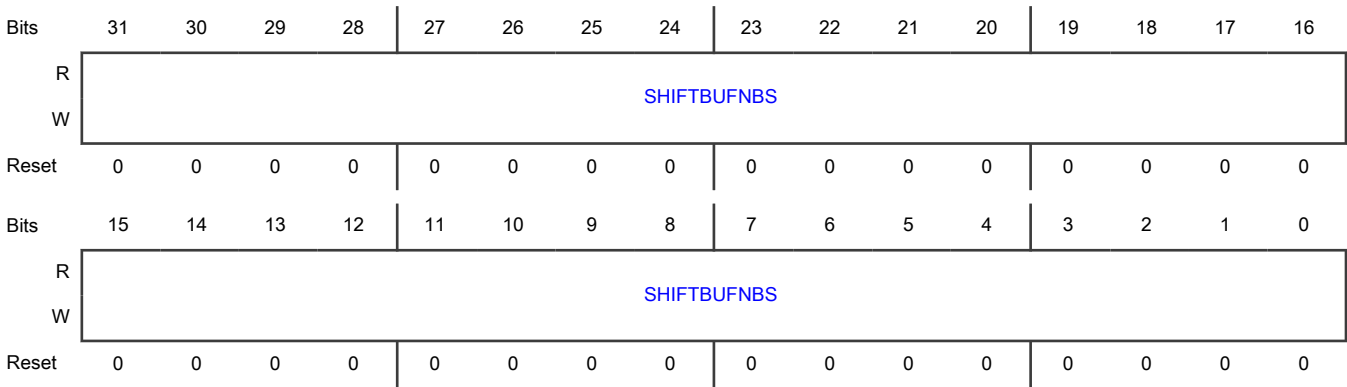
For a = 0 to 7:

Register	Offset
SHIFTBUFNBSa	680h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register are nibble swapped within each byte.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFNBS	Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped within each byte. Reads return { SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4] }.

51.6.1.37 Shifter Buffer N Half Word Swapped Register (SHIFTBUFHWS0 - SHIFTBUFHWS7)

Offset

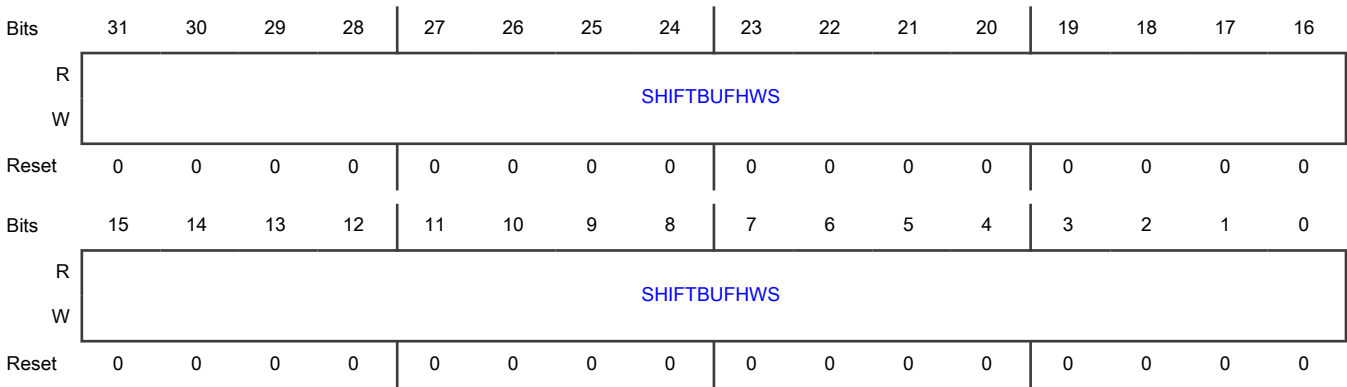
For a = 0 to 7:

Register	Offset
SHIFTBUFHWSa	700h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register are half word swapped.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFHWS	Alias to SHIFTBUF register, except reads/writes to this register are half word swapped. Reads return { SHIFTBUF[15:0], SHIFTBUF[31:16] }.

51.6.1.38 Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)

Offset

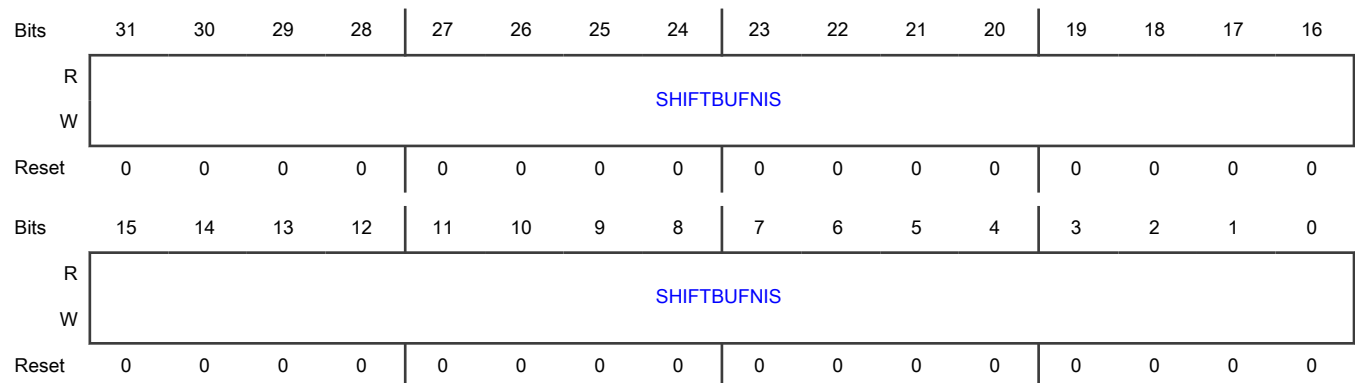
For a = 0 to 7:

Register	Offset
SHIFTBUFNISa	780h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register are nibble swapped.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFNIS	Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped. Reads return { SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28] }.

51.6.1.39 Shifter Buffer N Odd Even Swapped Register (SHIFTBUFOES0 - SHIFTBUFOES7)

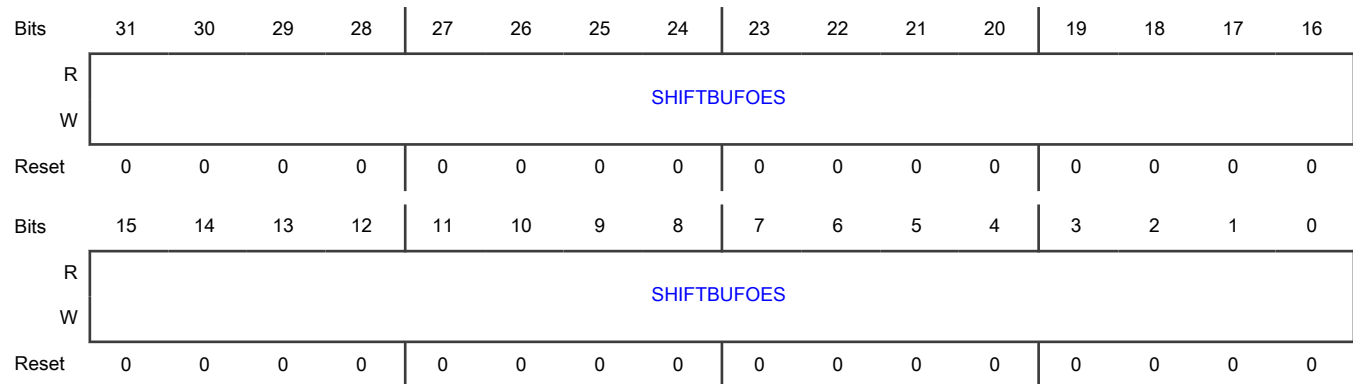
Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFOESa	800h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register have odd and even bits partitioned separately.

Diagram**Fields**

Field	Function
31-0 SHIFTBUFOES	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register have the odd and even bits partitioned separately. Only 32-bit accesses are supported to this register. Reads return { SHIFTBUF[31], SHIFTBUF[29], SHIFTBUF[27], SHIFTBUF[25], SHIFTBUF[23], SHIFTBUF[21], SHIFTBUF[19], SHIFTBUF[17], SHIFTBUF[15], SHIFTBUF[13], SHIFTBUF[11], SHIFTBUF[9], SHIFTBUF[7], SHIFTBUF[5], SHIFTBUF[3], SHIFTBUF[1], SHIFTBUF[30], SHIFTBUF[28], SHIFTBUF[26], SHIFTBUF[24], SHIFTBUF[22], SHIFTBUF[20], SHIFTBUF[18], SHIFTBUF[16], SHIFTBUF[14], SHIFTBUF[12], SHIFTBUF[10], SHIFTBUF[8], SHIFTBUF[6], SHIFTBUF[4], SHIFTBUF[2], SHIFTBUF[0] }.

51.6.1.40 Shifter Buffer N Even Odd Swapped Register (SHIFTBUFEOS0 - SHIFTBUFEOS7)**Offset**

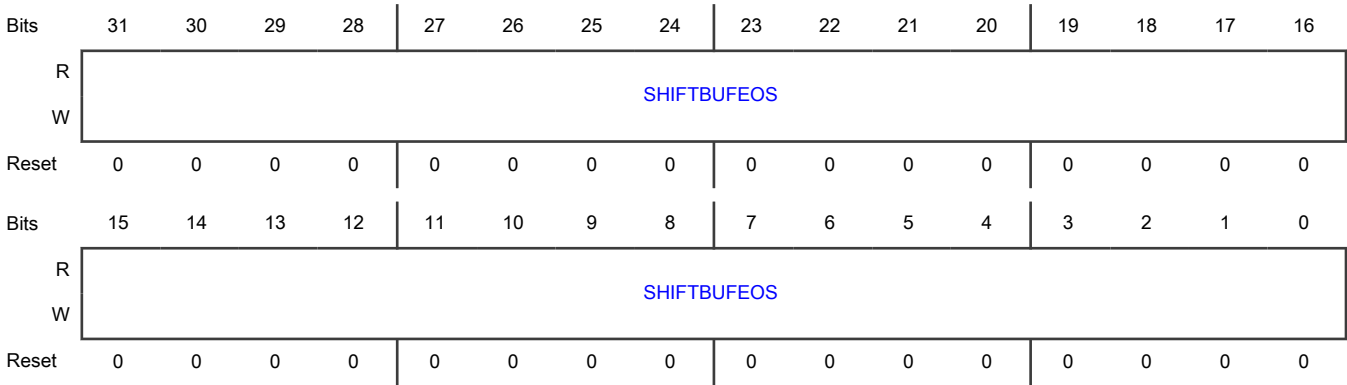
For a = 0 to 7:

Register	Offset
SHIFTBUFEOSa	880h + (a × 4h)

Function

SHIFTBUF register contents, but contents of this register have even and odd bits partitioned separately.

Diagram



Fields

Field	Function
31-0 SHIFTBUEOS	Shift Buffer Alias to SHIFTBUEOS register, except reads/writes to this register have the even and odd bits partitioned separately. Only 32-bit accesses are supported to this register. Reads return { SHIFTBUEOS[30], SHIFTBUEOS[28], SHIFTBUEOS[26], SHIFTBUEOS[24], SHIFTBUEOS[22], SHIFTBUEOS[20], SHIFTBUEOS[18], SHIFTBUEOS[16], SHIFTBUEOS[14], SHIFTBUEOS[12], SHIFTBUEOS[10], SHIFTBUEOS[8], SHIFTBUEOS[6], SHIFTBUEOS[4], SHIFTBUEOS[2], SHIFTBUEOS[0], SHIFTBUEOS[31], SHIFTBUEOS[29], SHIFTBUEOS[27], SHIFTBUEOS[25], SHIFTBUEOS[23], SHIFTBUEOS[21], SHIFTBUEOS[19], SHIFTBUEOS[17], SHIFTBUEOS[15], SHIFTBUEOS[13], SHIFTBUEOS[11], SHIFTBUEOS[9], SHIFTBUEOS[7], SHIFTBUEOS[5], SHIFTBUEOS[3], SHIFTBUEOS[1] }.

51.6.1.41 Shifter Buffer N Halfword Byte Swapped Register (SHIFTBUEHBS0 - SHIFTBUEHBS7)

Offset

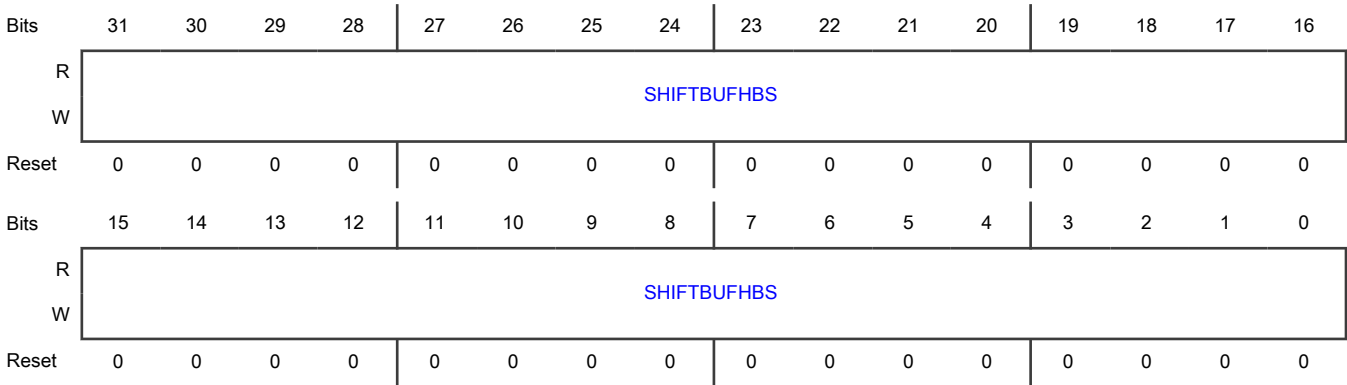
For a = 0 to 7:

Register	Offset
SHIFTBUEHBSa	900h + (a × 4h)

Function

SHIFTBUE register contents, but contents of this register are halfword byte swapped.

Diagram



Fields

Field	Function
31-0	Shift Buffer
SHIFTBUFHBS	Alias to SHIFTBUF register, except reads/writes to this register are halfword byte swapped. Reads return { SHIFTBUF[23:16], SHIFTBUF[31:24], SHIFTBUF[7:0], SHIFTBUF[15:8] }.

51.7 Glossary

Baud rate	Number of bits transmitted or received per second by the LPUART
Break character	Break character is generated when the transmitter is holding the data line at the space level for at least 1 character time
Idle character	Idle character is generated when the transmitter is holding the data line at logic 1 for at least 1 character time
PWM	Pulse Width Modulation

Chapter 52

Improved Inter-Integrated Circuit (I3C)

52.1 Chip-specific I3C information

Table 542. Reference links to related information

Topic	Related module	Reference
Full description	I3C	I3C
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

This chip supports 2 I3C master/slave interfaces that are accessible by both CPUs and both DMA controllers.

Supported High-Keeper methods

I3C's MCONFIG[HKEEP] identifies High-Keeper methods corresponding to the field's four possible values. This chip:

- Supports only the methods corresponding to when HKEEP is 00b, 10b, or 11b
- Does not support the method corresponding to when HKEEP is 01b

NOTE

Maximum I3C FCLK (clock after I3C01FCLKDIV register) is 25 MHz.

52.1.1 Module instances

This device has two instances of the I3C module, I3C0 and I3C1.

52.2 Overview

The MIPI Alliance Improved Inter-Integrated Circuit (MIPI I3C) brings major improvements in use and power over I2C, and provides an alternative to SPI for mid-speed applications. The I3C bus is designed to support future sensor interface architectures, widely expected in Internet-of-Things applications.

The I3C bus is intended to be used by microcontrollers (MCU) and application processors (AP) to connect to sensors, actuators, and other MCUs (as slaves). Connecting an MCU to other MCUs and connecting an AP to an MCU are considered to be the major use cases.

The I3C bus protocol supports:

- In-band interrupts: interrupts can go from Slave to Master without extra wires, such that the Master knows which Slave sent the interrupt.
- Common Command Codes (CCC)
- Dynamic addressing
- Multi-master / multi-drop
- Hot-Join
- I2C compatibility

52.2.1 Features

I3C module:

- 2-wire multi-drop bus capable of 12 MHz clock speeds, with up to 11 devices.
 - Uses standard pads (I2C uses special pads) with 4 mA drive
 - Slave addresses are dynamically assigned, and slaves do not require a static address. However, slaves may have an I2C static address assigned at start-up, so that the slave can operate naturally on an I2C bus.
 - Slaves may use the inbound SCL clock as the peripheral clock, which means that devices can have slow or inaccurate clocks internally.
 - Simple slaves can have no internal clock; for example, like a temperature sensor.
 - For reads from a slave, the slave normally ends the read, but the I3C master can also terminate the read. For I2C and SPI, the master must know the length of the data that is being read, but for I3C the master does not have to know the length of the data being read.
- In-Band interrupts (IBI) are allowed, which allow slaves to notify a master.
 - Can be both equivalent to a separate GPIO, but can also be directly data-bearing.
 - In-Band interrupts are prioritized, so that if multiple Slaves want to interrupt a master at the same time, then the order is resolved. dynamic addresses are used to establish the priority of the slaves, which means that the master controls the priority of the slaves.
 - Interrupts can be started even when the master is not active on the bus, and yet no free-running clock is needed (for that situation).
 - A time-stamping option allows the resolution of an initial event, not just when an interrupt gets through
- Built-in commands are in separate space, so that these commands do not collide with normal Master to Slave messages.
 - Controls bus behavior, modes and states, low-power state, inquiries, and more
 - Has additional room for new built-in commands to be used by other groups
- Organized forms of multi-master modes:
 - Secondary masters that can use clean handoffs between different Masters
- Hot-join onto I3C bus allows devices to connect to the bus later than when the bus starts.
 - Enables a device or module to get onto the I3C bus, because it woke up after power-up, or was physically inserted onto the I3C bus.
 - Also provides a clean method for notification when new devices or modules get onto the I3C bus.
- Capable of using both I2C and I3C buses:
 - I3C supports specific legacy I2C devices on the bus
 - I3C Slave devices can operate on I2C buses
 - Supports bridging to I2C, SPI, UART, and other buses
- Higher data rate modes are optionally available
 - Only the master and the specific slave have to support the higher data rate; the other slaves are able to ignore the higher data rate.
 - Has an HDR-DDR mode (High Data Rate - Double Data Rate), which is about 2x the data rate of SDR (so about 20 Mbps)
- Slaves can be as small as 2 K gates (or less), which allows the I3C to be fully state-machine driven, as well as using a processor (to control the I3C).
- Masters can be as small as 2.5 K gates, but rely on the processor to handle the I3C.

The I3C peripheral supports the full I3C feature set, except for the ternary data rates (HDR-TSP, HDR-TSL).

52.3 Functional description

The following sections describe functional details of the I3C module.

52.3.1 Operating modes

This section describes all functional operation modes of the I3C module.

52.3.1.1 Master vs. slave roles for I3C

The I3C protocol defines 5 roles for devices on the I3C bus:

- **Main Master:** controls the I3C bus
- **Secondary Master:** controls the I3C with permission from the Main Master, and returns control of the bus back to the Main Master when it (Secondary Master) has finished with its tasks
- **Slave:** responds to commands from any I3C master
- **Peer-to-Peer Slave:** can read or write to other peer-to-peer slaves without any help from the I3C master
- **I2C Slave:** responds to commands from any I3C master

The I3C peripheral contains both a master and slave component, and can be parameterized to be either master or slave, or both master and slave. However, if the I3C is chosen to be a master, then the I3C peripheral should generally support slave mode, to make handoffs to multiple masters easier to do.

In general, any I3C master should be able to be a master or a slave, because a master becomes a slave when giving over control to another master. The only exception to this rule would be when using a point-to-point master or when using a master that never shares mastership.

Master requirements

Master mode uses software that supports the various requirements of the master (including as a Secondary Master):

- Special handling of Enter Dynamic Address Assignments (ENTDAA): assigning dynamic addresses to each slave. This is a process that is supported by the I3C peripheral, but requires the software to make choices.
- Building up a table of slaves and their capabilities, which is used to control what kinds of actions and commands may be sent to the slaves.
- Handling requests such as In-band interrupts, peer-to-peer slave, and master request (handoff).
- Adjusting clock speed or write-to-read timing to match the slave's limitations. This can be done in hardware using dividers and uneven duty cycles, but the software must make the decisions.
- Adjusting the maximum data length.
- Being a slave after a master handoff to another master

Additionally, a master needs 3 pins normally, unless a very precise, high strength pullup is included in the SDA pad. The 3-pin model allows 1 pad/pin to enable a system-provided pullup, which is sized to system requirements.

The master needs a reasonably accurate clock, normally capable of a frequency that is a multiple of a frequency between 11 MHz and 12.5 MHz. For example:

- 24 MHz (multiple of 12 MHz)
- 48 MHz (multiple of 12 MHz)
- 25 MHz (multiple of 12.5 MHz)
- 50 MHz (multiple of 12.5 MHz)
- 33 MHz (multiple of 11 MHz)

- 66 MHz (multiple of 11 MHz)

Higher ratios can also be used.

Slave requirements

The I3C slave mode is intended to allow a much simpler module to be used, allowing the device implementation to use fewer logic circuits when space on the device is limited. For example, when the slave minimum requirements are very small (just I3C SDR mode and a few Common Command Code (CCC) commands). Conversely, slave functionality in the module logic can be increased, increasing the module's area on the device, but saving software development costs. What is important is that the Slave can be scaled for system requirements, by doing more in software or by doing more in hardware. This decision is made when the module is integrated into the device.

I3C slave acting as a normal I2C slave on I3C buses

The I3C slave, if assigned an I2C static address, acts like a normal I2C device when the I3C slave first comes up. If the I3C slave is placed on an I2C bus with an I2C master, then the I3C slave will simply stay in I2C mode and operate normally. The software knows that the I3C slave is in I2C mode, because:

- There has not been a DACHG indicating that a dynamic address was assigned
- The SDYNADDR register contains 0x0

If full I2C support for Fast Mode (FM) and Fast Mode Plus (FM+) is desired, then the pads should also support a 50ns spike filter, which must be turned off when the I3C 7'h7E broadcast address is seen (indicating an I3C master). Turning off the spike filters can be done by hardware (using the raw net indicating that the address was seen) or can be done by software. A 50ns spike filter is not needed for the I3C to operate on an I2C bus; therefore the spike filter is not a requirement for the I3C peripheral.

How a slave re-joins the I3C bus

When a slave powers up or after a hard reset, when the slave tries to rejoin the I3C bus, the slave needs a new Dynamic Address (DA). The slave provides ways to rejoin the I3C bus:

- If the dynamic address is lost, then Hot-Join is used.
- If the dynamic address is retained in the peripheral (for example, with SRFFs), then the SCONFIG.OFFLINE bit is used (see below).
- If the dynamic address is retained in separate always-on memory, then the SDYNADDR register (as a writable mechanism) can be used with the SCONFIG.OFFLINE bit.

When SLVENA is set to 1, the OFFLINE bit of the SCONFIG register may be set. This causes the peripheral to safely rejoin the bus, by ensuring that the I3C bus is not in High Data Rate (HDR) mode, using the same approach as S0/S1 exit does: waiting for the first of the HDR Exit Pattern, or waiting for 60 μ s of SCL and SDA lines not changing.

After using OFFLINE, the I3C peripheral still cannot safely use IBI until it sees a STOP (SSTATUS register), which ensures that the next START is safe to use for IBI).

If the application needs to do an IBI right away, then the application should wait for the first STOP (SSTATUS register), or should wait for 200 μ s of SCL and SDA lines being High. This can be done using the SSTATUS and SINTSET controls.

- If the SSTATUS indicates that the bus is not busy and the peripheral interrupts on START or STOP, then use a timer to measure 200 μ s. If the timer goes off with no START or STOP, then it is safe to use IBI.
- If a START causes an interrupt, then the timer should be turned off and the application should wait for a STOP.
- If a STOP causes an interrupt, then it is safe to use IBI.

52.3.1.2 Using the I3C Master for I2C and I3C

The I3C Master operates with a set of built-in capabilities mixed with the application flow:

- Built-in Enter Dynamic Address Assignment (ENTDAA) mechanism to simplify assignment of DAs to slaves (when Set Dynamic Address from Static Address (SETDASA) is not available).

- Request for START+Address with IBI support, including both I2C and I3C modes
- SDR Write flow via FIFO, with automatic parity in I3C and ACK/NACK detect in I2C
- SDR Read flow via the FIFO, with an automatic NACK generator for I2C, and optional use of a terminate generator for I3C
- Request for repeated START+Address or STOP when finished with previous, including both I2C and I3C
- Auto-IBI mode which responds immediately to a Slave-initiated IBI; can be used when in sleep or deep sleep
- Special message mode for SDR and DDR to simplify for DMA use
- All SCL, SDA, Pull-up, and High-Keeper controls are automated
- I2C and I3C Frequency and duty cycle configurations from functional clock

Note the MCONFIG register and the MIBIRULES register should be configured before using the Master.

52.3.2 Operations

This section describes the I3C module's operations.

52.3.2.1 Reading and writing I2C messages using normal method

The normal method:

1. Set up interrupts. Normally:
 - a. MCTRLDONE: optional
 - b. COMPLETE: to fire when end of data
 - c. RXPEND if read; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to read out bytes.
 - d. TXNOTFULL if write; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to supply bytes.
 - e. IBIWON: so the software knows if an IBI has occurred
 - f. MERRWARN: in case of an error
2. Write the MCTRL register with:
 - a. REQUEST set to 1 (EmitStartAddr)
 - b. TYPE set to 1 (I2C)
 - c. IBIRESP set depending on how you want IBIs handled
 - d. DIR set to 1 if Read, or set to 0 if Write
 - e. ADDR set to the static address of the I2C slave
 - f. If read, then RDTERM may be set for the length, or may be set as the data is read out (for example, set to 1 (with REQUEST=0) when you want it to stop next).
3. If Write: Write the MWDATAB register for each byte before the last byte, then write the MWDATAB register for the last byte.
 - a. This can be done or started before step 1 **Set up interrupts**
 - b. If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level, to allow the application to provide more, or use DMA.
4. If Read: wait for RXPEND, and then read out data via the MRDATAB register. DMA may also be used
5. On COMPLETE, the message may be ended with a STOP, or a new message started with a repeated START.
 - a. Write MCTRL with REQUEST of 2 (EmitStop to STOP)
 - b. Write MCTRL with REQUEST of 1 (EmitStartAddr to start another message).

Note that IBI is not possible in this case.

52.3.2.2 Reading/writing I3C messages using normal methods (SDR and HDR-DDR)

The normal method for I3C is the same as for I2C with a few differences. Note that **Option 2: From stopped and not in HDR-DDR mode** below is preferred from stopped (bus free condition) when in Single Data Rate mode (SDR) [and not in High Data Rate - Double Data Rate mode (HDR-DDR)], because it allows any slave to issue an IBI, it avoids collisions with IBI addresses, and it is faster (when MSB of dynamic addresses are always 0).

1. Set up interrupts. Normally:
 - MCTRLDONE: optional
 - COMPLETE: to fire when end of data
 - RXPEND if read; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to read out bytes.
 - TXNOTFULL if Write; this can be used with the MDATACTRL register to set the FIFO trigger. The alternative is for DMA to supply bytes.
 - IBIWON: so the software knows if an IBI has occurred
 - MERRWARN: in case of an error
2. Write the MCTRL register:
 - **Option 1:** Write the MCTRL register with:
 - a. REQUEST set to 1 (EmitStartAddr) (cannot have pre-written TX data in the FIFO)
 - b. TYPE set to 0 (if I3C) or set to 2 (if DDR)
 - c. IBIRESP set depending on how you want IBIs handled
 - d. DIR set to 1 (if Read) or set to 0 (if Write)
 - e. ADDR set to the dynamic address of the I3C slave
 - f. If read, then optionally RDTERM is set for the maximum length to auto-terminate.
 - g. If write, then pre-writing the data (MWDATAB or MWDATAW) is preferred, to ensure that there are no time delays waiting on the data.
 - **Option 2:** From stopped and not in HDR-DDR mode, write MCTRL register with;
 - a. REQUEST set to 1 (EmitStartAddr)
 - b. TYPE set to 0 (I3C)
 - c. IBIRESP set depending on how you want IBIs to be handled
 - d. DIR set to 0
 - e. ADDR set to 0x7E
 - f. Wait for MCTRLDONE (for example, by interrupt), then proceed with **Option 1**: This Option 2 method has advantages for IBIs when 7E is emitted on START (but not on repeated STARTs).

NOTE

The Option 2 method would not allow an immediate switch to read; the normal I3C process is to write index, then read data. So, if doing a blind read (no index is written), do not use Option 2 method.

3. If Write: then write the MWDATAB register for each byte before the last byte, then write the MWDATABE register for the last byte.
 - This can be done or started before step 1 [REQUEST set to 1 (EmitStartAddr)] of **Option 1: Write the MCTRL register with**. It should not be before **Option 2: From stopped and not in HDR-DDR mode**.

- If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level, to allow the application to provide more; or use DMA.
- 4. If Read: wait for RXPEND, and then read out data using the MRDATAB register. DMA may also be used.
- 5. On COMPLETE, the message may be ended with STOP, or a new message started with a repeated START.
 - Write MCTRL with REQUEST of 2 (EmitStop to STOP)
 - Write MCTRL with REQUEST of 1 (EmitStartAddr to start another message).

Note that IBI is not possible in this case.

52.3.2.3 Sending a CCC to one or all I3C slaves

The normal Common Command Code (CCC) method is to use an I3C Write with an address of 0x7E. The 1st byte will be the CCC.

- If Broadcast, then any remaining bytes are the bytes that go with the Common Command Code (CCC). This would end with a STOP or a repeated START and 0x7E normally.
- If Direct, then only the CCC byte is sent, followed by a repeated START and the address of the I3C Slave (for SETDASA, this is its I2C Static address). This may be repeated with more repeated START and addresses until done. Then, it may end in STOP or a repeated START and 0x7E.

NOTE

Each repeated START is just a new EmitStart request. This can be chained by interrupt or pushed by message model using DMA.

52.3.2.4 In-band interrupt (IBI) handling

An in-band interrupt occurs when a slave sends its address after a START, and the slave's address is numerically lower than the address that the master sent (and any other slaves trying to send their addresses). When the master sends a 0x7E, the master will always lose, which is the intention.

The in-band interrupt (IBI) can happen unexpectedly when any new START (vs. repeated START) is emitted, but the in-band interrupt can also be in response to a slave pulling SDA low, which can happen in one of two ways:

- The slave has set the request to AutoIBI mode, and so it happens automatically (the engine emits 0x7E to allow the slave to win).
- The application gets the SLVSTART (slave START request) interrupt and so emits 0x7E (normally).

In either of these cases, the application can decide whether to:

- Always accept (ACK) the IBI
- Always reject (NACK) the IBI
- Allow the decision to be made by the application on a case-by-case basis (manual).

How the IBI is handled is configured by the MCTRL.IBIRESP field, which can be set to ACK, NACK, or manual.

- If manual is selected, then the application rewrites it when stopped, pending on an IBI. The application can ACK or NACK based on the IBI address (which is in the MSTATUS register). Note that manual mode chooses if there is an IBI byte or not when doing ACK.
- If always ACK was selected in the IBIRESP field, then the MIBIRULES register must be configured so that the engine knows if bytes will follow.
- If IBI bytes will follow (also known as IBI Mandatory byte), then the COMPLETE bit is not set when IBIWON is set, and RXPEND is set for 1 or more bytes. When the last byte is received, then the COMPLETE status is set.

52.3.2.5 Assigning dynamic addresses to I3C devices

If Dynamic Addresses (DAs) are all assigned below 0x40 (7-bit values from 0x3F down to 0x03, except where this is not allowed), then the MSB0 bit can be set in the Master In-Band Interrupt Registry and Rules register (MIBIRULES). This optimizes the START timing. When dynamic addresses are assigned, the MIBIRULES register should be programmed based on which I3C slave uses IBI bytes (known from Bus Capabilities Register (BCR)).

Any SETDASA (Set Dynamic Address from Static Address) assignments can be done using the normal Common Command Code (CCC) model (except when using the static address for the directed part).

There is a built-in mechanism to process the Dynamic Address Assignment (DAA) mode:

1. Set up interrupts. Normally:
 - MCTRLDONE: need to know when a slave has sent its ID and BCR/DCR and a new DA is needed.
 - COMPLETE: to fire when DAA done (NACKed by all slaves).
 - RXPEND to read the IDs of the slaves. The alternative is for DMA to read out bytes.
 - IBIWON: so the software knows if an IBI happened, which should not be possible normally.
 - MERRWARN: in case of an error
2. Write the MCTRL register with:
 - REQUEST set to 4 - ProcessDAA
 - IBIRESP set to IBI response, if that is possible (if not the 1st Slave assignment)
3. Wait for the MCTRLDONE interrupt, while reading in ID using the RXPEND interrupt. If MSTATUS indicates in DAA mode (STATE=5) and BETWEEN (bit 4=1), then it is waiting for a dynamic address for the slave whose ID was read in.
 - Write the dynamic address into MWDTAB using bits 6:0, such as 0x14 for dynamic address=7'h14.
 - Write MCTRL REQUEST to 4 (ProcessDAA again). This writes the dynamic address and then move on to the next dynamic address.
 - If MSTATUS indicates all is done via the COMPLETE bit being set and STATE=0, then all slaves have been assigned.
 - If MSTATUS indicates NACK after having written a new dynamic address, then the dynamic address was not accepted by the slave. The normal next step is REQUEST set to EmitStop and start over. However, it is also OK to set REQUEST to 4 (ProcessDAA again).

52.3.2.6 Using Master Message mode

Master Message mode is really intended for use with DMA, although message mode can also be used by the processor. The message mode concept is that all writes are to the same location, including the control and then the data; the read back is from an associated location.

NOTE

The data writes for message mode work exactly in the same way as the half-word access registers MWDTAB and MRDTAB.

To send a message via Single Data Rate (SDR), the following steps are used (from DMA or from processor):

1. Write to the MWMSG_SDR register with the control request, which includes the address, I2C vs. I3C, read or write, the count of bytes to process, and how to end (on STOP or ready for repeated START).
2. Process the Data:
 - If Write: then write the rest of the data to MWMSG, using the DMA trigger (or interrupt) to keep the TX FIFO full. Will stop asking when count is down to 0.
 - If Read: then a DMA trigger (or interrupt) indicates when the RX FIFO is to be read out via the MRMSG_SDR.
3. In-band interrupt (IBI) behavior is selected in the MCTRL register.

4. Use END type STOP to exit MSG mode. This can be done with the original message, with a 0 length message, or by using the MCTRL register.

To send a message via Double Data Rate (DDR), the following steps are used (from DMA or from processor):

1. Write to the MWMSG_DDR register with the control request, which includes the count of byte-pairs and how to end (on High Data Rate (HDR) Exit or ready for HDR Restart).
2. Write the 2nd control data to MWMSG_DDR, which includes the address, read or write, and the 7-bit command value.
3. Process the Data:
 - If Write: then write the rest of the data to same register, using the DMA trigger (or interrupt) to keep the TX FIFO full. Will stop asking when count is down to 0.
 - If Read: then a DMA trigger (or interrupt) indicates when the RX FIFO is to be read out.
4. In-band interrupt (IBI) behavior is selected in the MCTRL register.
5. Use END type Exit to exit DDR MSG mode. This can be done with the original message, with a 0 length message, or by using the MCTRL register.

52.3.2.7 Handing off mastership to another slave and getting it back

To hand off mastership, the master either:

- Waits for an Master Request (MR) (an IBIWON interrupt with MSTATUS indicating that an MR is using the IBITYPE field)
- Or pushes the request manually

In either case, the master sends a GETACCMST request. This is a directed GET that tells the slave that the slave is being assigned mastership (if the slave wants it). If the slave accepts (and sends back its own dynamic address in bits 7:1 and the negative parity of the dynamic address in bit 0), then a STOP should be issued and the MCONFIG.MSTENA field should be set to 2 (switching to slave mode).

To gain mastership, the slave sends an MR using the SCTRL register.

- If the MSTENA field is 2, then the GETMSTACC CCC (Common Command Code) will accept
- If the MSTENA field is not 2, then the GETMSTACC CCC will refuse

After mastership has granted, the MSTATUS.NOWMASTER bit is set. The application must enable the NOWMASTER bit in the MINTSET register, so the application will be interrupted when a mastership transfer happens.

52.4 External signals

This table describes the I3C module signals.

Signal	I/O	Description
SCL	I/O	Serial clock
SDA	I/O	Serial data
PUR	O	Pull up resistance. There is internal pull up resistance on SDA, which is controlled by the I3C master. PUR can be used to actively control an external pull up resistance on SDA, if the internal pull up is not enough.

52.5 Initialization

52.5.1 Configuration initialization

The configuration is handled once (normally) when initializing the I3C module. Configuration is done using the MCONFIG register, which controls the frequencies, duty cycle, optimizations for performance, and other parameters.

Table 543. Configuration parameters used to initialize the I3C module

Configuration Parameter		Description
MSTENA	Master Enable	Determines whether the I3C peripheral starts in Master mode (ie. the “Main Master” in I3C terms) or starts in slave mode (and will switch to Master later).
HKEEP	High Keeper	Determines how the high-keeper (weak pullup) is to be implemented, depending on the device capabilities.
PPBAUD	Push-Pull BAUD	Sets the push-pull frequency as a divider from the FCLK (alternative clock fed to the peripheral). This sets the half-clock period baseline (used at least for the high time of SCL). • If FCLK = 24 MHz, then a PPBAUD = 0 yields 12 MHz (42.67 ns per half period) • If FCLK = 50 MHz, then a PPBAUD = 1 yields 12.5 MHz (20+20= 40 ns per half period)
PPLOW	Push-Pull Low	Changes the duty cycle for push-pull. It indicates how many more FCLKs to use for low. For example, a 50 MHz FCLK, a PPBAUD of 1, and a PPLOW of 1, yields 20+20= 40 ns high and 20+20+20= 60 ns low. This would be equivalent of 10 MHz SCL timing, but maintaining the 40 ns high needed, so that I2C devices do not see the high periods.
ODBAUD	Open Drain BAUD	The number of PPBAUD periods to make up one I3C open-drain half-clock baseline. For example, if PPBAUD yields 12 MHz (40 ns per PPBAUD period), then 5 PPBAUD can be used to get 200 ns. See also ODHPP for details about short high and long low.
ODHPP	Open Drain High Push-Pull	Optional field that allows for the I3C open drain to be long low and short high. The high period of SCL will be the PPBAUD period. This leaves enough time for the pull-up resistor to pull the SDA high when SCL is low, but is quick when SCL is high and there are no changes happening.
I2CBAUD	I2C Baud	Indicates how many ODBAUD periods are needed to communicate with I2C devices. For example, if ODBAUD gives 200 ns, and the goal is FM+ (Fast Mode, 1 MHz), then the sum needs to be 1 us. If odd vs. even, then I2CBAUD acts differently. For example, if I2CBAUD= 3, it will give 3 ODBAUD periods low and 2 ODBAUD periods high. • If I2CBAUD= 4, then it will give 3 ODBAUD periods for low and 3 ODBAUD periods for high. • Also, if I2CBAUD= 3, you get 200*3= 600 ns and 200*2= 400 ns, with the sum 600 + 400 = 1000ns = 1us.
SKEW	Skew	The normal SDA skew from SCL is handled by using the time for the SCL to reach its pad and come back to the design. This is normally 2 ns to 5 ns (or sometimes worse). If the SKEW is too fast, then to add more delay, the SKEW allows specifying the number of FCLKs to insert.

Additional optimizations:

- Use MIBIRULES.MSB0 to get much faster START header times. If the master application assigns all I3C Dynamic Addresses to be less than 0x40 (it does not have MSB set), then this bit (MSB0) can be set. This means that when the master emits 0x7E (broadcast) and the 1st bit is not driven low by a slave, the rest of the header can be at push-pull speeds. This is about 2x faster (or more, depending on optimizations above).
- Auto-emit 0x7E speeds up the frame when used in conjunction with MSB0, because it allows the processor to be sleeping when the frame starts automatically (in response to a slave).

52.6 Application information

This section describes applications supported by the I3C module.

52.6.1 Using registers when the System clock is much faster than the Functional clock

The Master block has 2 basic clocks:

- System clock: which controls the access into the memory-mapped registers.
- Functional clock (FCLK): which is used to generate the SCL clock rate on the I2C/I3C bus.

Because there are two clocks, requests made to MCTRL and MWMSG_xxx registers have to cross over the clock domains. This is also true of data reads and writes, but there are certain aspects of MCTRL that require consideration.

The normal use is to write MCTRL and wait for an interrupt indicating DONE and/or COMPLETE. Once that interrupt arrives, the interrupt handler likely writes a new request (like a new message or a STOP).

- **When the System clock is less than or equal to 8x faster than the FCLK**, there are no special actions needed. The servicing of the interrupt takes long enough to ensure that all clock-crossing details are settled. For example, if the FCLK is 25 MHz and the system clock is 150 MHz, there are no problems.
- **When the System clock is more than 8x faster than FCLK (or if the code is using a polling spin loop on MSTATUS)**, then special considerations may apply:
 - The MSTATUS sticky bits for DONE and COMPLETE are set, and optionally an interrupt is triggered.
 - To complete the clock crossing, the internal state goes through a handshake process.
 - If a new request posts before 2 FCLKs have completed, then the new request may be lost.
 - For example, if the FCLK is 25 MHz and the System clock is 400 MHz, then 2 FCLKs is 16 beats of the System clock, or really 17 beats (since the clocks are not aligned). If the interrupt handler is entered in under 16 clocks and the interrupt handler tries to write the MCTRL register immediately, then the new request is lost.
 - In reality, most processors need around 12 clocks to enter an interrupt handler, and then the handler code needs to read MSTATUS and make decisions, so this will not be an issue; but if the processor is very fast, then an extra delay is needed.
 - Likewise, if using a polling spin loop, then the timing from detecting MSTATUS DONE or COMPLETE to writing the next MCTRL would need to consider that time required (2 FCLKs).

52.6.2 Master registers summary

The Master registers are integrated with the Slave registers. Other than the configuration registers, the Master registers are above all of the Slave registers, starting at 0x80. The mapping is designed to work for both types of registers, and many Master registers are just aliases of Slave registers, but with different meanings.

- When Master operation is enabled (MCONFIG.MSTENA = 1), the Master registers should be used.
- When Slave operation is enabled (MCONFIG.MSTENA = 0), the Slave registers should be used.

Offset	Type	Register		Description
0x000	RW	MCONFIG	Master Configuration	Configuration for the master
0x084	RW	MCTRL	Master Control	Master main control. Starts actions on the bus (see Using Master Message mode).
0x088	RO and R/W1C	MSTATUS	Master Status	Status for the master, including interrupt causes.

Table continues on the next page...

Table continued from the previous page...

Offset	Type	Register		Description
0x08C	RW	MIBIRULES	Master In-Band Interrupts Register and Rules	Rules for IBI use, and registry of Slaves which use an IBI byte.
0x09C	RW	MERRWARN	Master Errors and Warnings	Error and Warning state from the misuse of registers and detected errors on the bus. Related to MSTATUS.ERRWARN bit.
0x0A0	RW	MDMACTRL (alias)	Master DMA Control	DMA control register, if DMA is enabled in the I3C module. Alias of Slave DMACTRL (SDMACTRL).
0x0AC	RW	MDATACTRL (alias)	Master Data Control	Controls data buffering and indicates the current buffer state. Alias of Slave DATACTRL (SDATACTRL). This is also controls the FIFO.
0x0B0	WO	MWDATAB (alias)	Master Write Data Byte	Write a byte of data, possibly the last byte. Alias of Slave WDATAB (SWDATAB).
0x0B4	WO	MWDATABE (alias)	Master Write Last Data Byte	Write the last byte of data. Alias of Slave WDATABE (SWDATABE).
0x0B8	WO	MWDATAH (alias)	Master Write Data Half-Word	Write a half-word of data, possibly the last half-word of data. Alias of Slave WDATAH (SWDATAH).
0x0BC	WO	MWDATAHE (alias)	Master Write Data Half-Word as End	Write the last half-word of data. Alias of Slave WDATAHE (SWDATAHE).
0x0C0	RO	MRDATAB (alias)	Master Read Data Byte	Read a byte of data. Alias of Slave RDATAB (SRDATAB).
0x0C4	RO	MRDATAH (alias)	Master Read Data Half-Word	Read a half-word of data. Alias of Slave RDATAH (SRDATAH).
0xD0	WO	MWMSG_SDR (semi-alias)	Master Write SDR Message	Set up and push through a message in Single Data Rate (SDR) mode. Written with 16b words.
0xD4	RO	MRMSG_SDR (alias)	Master Read SDR Message	Read content from Slave in message mode. Read as 16b words.
0xD8	WO	MWMSG_DDR	Master Write DDR Message	Set up and push through a message in DDR mode. Written with 16b words.

Table continues on the next page...

Table continued from the previous page...

Offset	Type	Register		Description
0xDC	RO	MRMSG_DDR (alias)	Master Read DDR Message	Read content from a slave in Double Data Rate (DDR) message mode. Read as 16b words.
0xE4	RW	MDYNADDR	Master Writable Dynamic Address	Allows writing its own Dynamic Address when switching to slave mode

52.6.3 Slave registers summary

Offset	Type	Register		Description
0x000	RW	Reserved		Reserved for the master
0x004	RW	SCONFIG	Configuration	Configuration fields to set up before a block is activated
0x008	RO	SSTATUS	Status	Status for master and slave. Not all bits used if only slave mode is used.
0x00C	RW	SCTRL	Control	Control for active use, usually event generation (interrupts to the master)
0x010	R/W1S	SINTSET	Interrupts Set	Interrupt enables
0x014	W1C	SINTCLR	Interrupts Clear	Interrupt disables
0x018	RO	SINTMASKED	Interrupts Masked	Mask of sourced interrupts (SSTATUS ANDed with SINTSET)
0x01C	RW	SERRWARN	Errors and Warnings	Error and Warning state from protocol errors and issues. Related to SERRWARN status and interrupt.
0x020	RW	SDMACTRL	DMA Control	DMA control register
0x02C	RW	SDATACTRL	Data Control	Controls data buffering and indicates the current buffer state. This register also controls the FIFO.
0x030	WO	SWDATAB	Write Data Byte	Write a byte of data, including using the 9th bit to mark the end (of the last byte)
0x034	WO	SWDATABE	Write Last Data Byte	Write a byte of data, which is the end (of the last byte).
0x038	WO	SWDATAH	Write Data Half-Word	Write a half-word of data, using the 16th bit to mark as the end (the last byte of a half-word is the end).
0x03C	WO	SWDATAHE	Write Data Half-Word as End	Write a half-word of data, which is the end (the last byte of a half-word is the end).
0x040	RO	SRDATAB	Read Data Byte	Read a byte of data
0x048	RO	SRDATAH	Read Data Half-Word	Read a half-word of data

Table continues on the next page...

Table continued from the previous page...

Offset	Type	Register		Description
0x060	RO	SCAPABILITIES	Capabilities	Indicates what is available/supported in this module, including master and/or slave, HDR modes, and others.
0x064	RO	SDYNADDR	Writable Dynamic Address	The dynamic address once assigned; otherwise this is 0.
0x068	RO	SMAXLIMITS	Maximum Limits	Indicates the maximum read and write limits set by the master (or the original requested limits).
0x06C	RW	SIDPARTNO	ID Part Number	Enables the application to write the ID part-number.
0x070	RW	SIDEXT	ID Extensions	Enables the application to write the ID extension of DCR and/or BCR and/or instance.
0x074	RW	SVENDORID	Vendor ID	Enables the application to write the Vendor ID.
0x078	RW	STCCLOCK	Time Control Clock	Enables the application to dynamically set the time control clock and accuracy information.
0x07C	RO	SMSGMAPADDR	Message Mapped Address	Returns the index of the matching Dynamic address or Static address.
0xFFC	RO	SID	Block ID	Shows the block ID (module ID) and revision.

52.7 I3C register descriptions

Table 544. Register Types

Type		
R	Read	
RO	Read-Only	
RW	Read/Write	
W1C	Write 1 to Clear	W1S and W1C registers are readable (they can be read).
W1S	Write 1 to Set	
WO	Write-Only	

NOTE

Writing a Read-Only (RO) register or reading a Write-Only (WO) register can cause bus errors. This module will not check if programmed values in the registers are correct; the application software must ensure that valid programmed values are being written.

However, the I3C peripheral will ignore writes to RO registers, and the I3C peripheral will return 0 for reads from WO or non-existent registers. It requires all 32-bit regs to be read and written as 32-bit and aligned to 32 bits; the peripheral uses the Advanced Peripheral Bus (APB), so the peripheral would not know if partial read or write operations (less than 32 bits) are used.

52.7.1 I3C memory map

I3C0 base address: 4003_6000h

I3C1 base address: 4003_7000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Master Configuration Register (MCONFIG)	32	RW	See section
4h	Slave Configuration Register (SCONFIG)	32	RW	See section
8h	Slave Status Register (SSTATUS)	32	RW	See section
Ch	Slave Control Register (SCTRL)	32	RW	See section
10h	Slave Interrupt Set Register (SINTSET)	32	RW	See section
14h	Slave Interrupt Clear Register (SINTCLR)	32	RW	See section
18h	Slave Interrupt Mask Register (SINTMASKED)	32	R	See section
1Ch	Slave Errors and Warnings Register (SERRWARN)	32	RW	See section
20h	Slave DMA Control Register (SDMACTRL)	32	RW	See section
2Ch	Slave Data Control Register (SDATACTRL)	32	RW	See section
30h	Slave Write Data Byte Register (SWDATAB)	32	RW	See section
34h	Slave Write Data Byte End (SWDATABE)	32	RW	See section
38h	Slave Write Data Half-word Register (SWDATAH)	32	RW	See section
3Ch	Slave Write Data Half-word End Register (SWDATAHE)	32	RW	See section
40h	Slave Read Data Byte Register (SRDATAB)	32	R	See section
48h	Slave Read Data Half-word Register (SRDATAH)	32	R	See section
60h	Slave Capabilities Register (SCAPABILITIES)	32	RW	E83F_FE78h
64h	Slave Dynamic Address Register (SDYNADDR)	32	RW	See section
68h	Slave Maximum Limits Register (SMAXLIMITS)	32	RW	See section
6Ch	Slave ID Part Number Register (SIDPARTNO)	32	RW	0000_0000h
70h	Slave ID Extension Register (SIDEXT)	32	RW	See section
74h	Slave Vendor ID Register (SVENDORID)	32	RW	See section
78h	Slave Time Control Clock Register (STCCLOCK)	32	RW	See section
7Ch	Slave Message-Mapped Address Register (SMSGMAPADDR)	32	R	See section
84h	Master Main Control Register (MCTRL)	32	RW	See section
88h	Master Status Register (MSTATUS)	32	RW	See section
8Ch	Master In-band Interrupt Registry and Rules Register (MIBIRULES)	32	RW	0000_0000h
90h	Master Interrupt Set Register (MINTSET)	32	RW	See section
94h	Master Interrupt Clear Register (MINTCLR)	32	RW	See section
98h	Master Interrupt Mask Register (MINTMASKED)	32	R	See section
9Ch	Master Errors and Warnings Register (MERRWARN)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
A0h	Master DMA Control Register (MDMACTRL)	32	RW	See section
ACh	Master Data Control Register (MDATACTRL)	32	RW	See section
B0h	Master Write Data Byte Register (MWDATAB)	32	RW	See section
B4h	Master Write Data Byte End Register (MWDATABE)	32	RW	See section
B8h	Master Write Data Half-word Register (MWDATAH)	32	RW	See section
BCh	Master Write Data Byte End Register (MWDATAHE)	32	RW	See section
C0h	Master Read Data Byte Register (MRDATAB)	32	R	See section
C8h	Master Read Data Half-word Register (MRDATAH)	32	R	See section
CCh	Write Byte Data 1 (to bus) (MWDATAB1)	32	RW	0000_0000h
D0h	Master Write Message in SDR mode (MWMSG_SDR_CONTROL)	32	RW	See section
D0h	Master Write Message Data in SDR mode (MWMSG_SDR_DATA)	32	RW	0000_0000h
D4h	Master Read Message in SDR mode (MRMSG_SDR)	32	R	See section
D8h	Master Write Message in DDR mode (MWMSG_DDR_CONTROL)	32	RW	See section
D8h	Master Write Message Data in DDR mode (MWMSG_DDR_DATA)	32	RW	0000_0000h
DCh	Master Read Message in DDR mode (MRMSG_DDR)	32	RW	See section
E4h	Master Dynamic Address Register (MDYNADDR)	32	RW	See section
FFCh	Slave Module ID (SID)	32	R	See section

52.7.2 Master Configuration Register (MCONFIG)

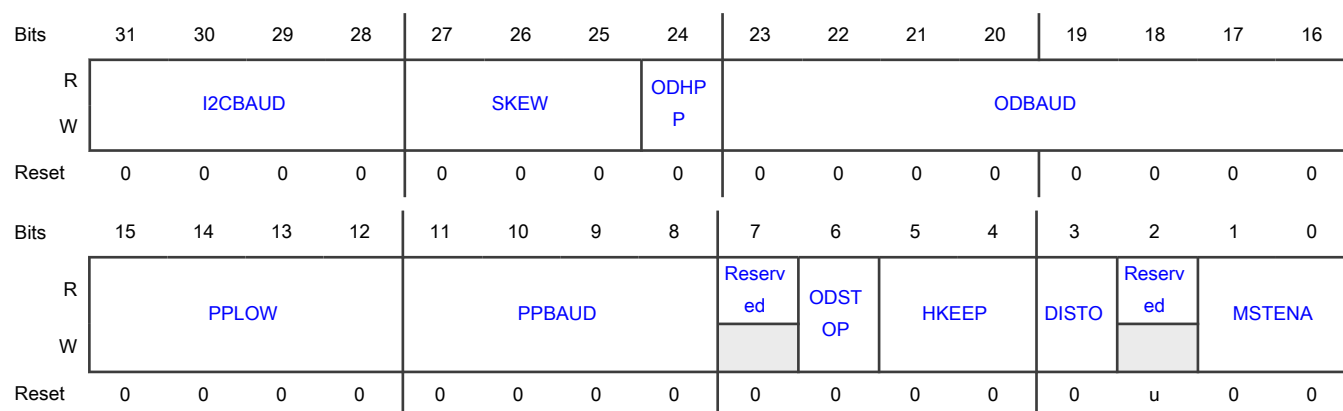
Offset

Register	Offset
MCONFIG	0h

Function

The MCONFIG register controls all Master states when Master operation is enabled. The MCONFIG register should not be changed when an active transaction is happening.

Diagram



Fields

Field	Function
31-28 I2CBAUD	I2C baud rate The I2C low and high times in ODBAUD counts. - I2CBAUD > 1 is the main count load, and it is <i>count-1</i>. Therefore, for I2CBAUD > 1: I2CBAUD==0 is 1 ODBAUD beat, and I2CBAUD==1 is 2 ODBAUD beats. - If I2CBAUD[28] is 1, then low will be 1 extra ODBAUD beat. This means that the I2CBAUD field is normally 3, where ODBAUD gives 200 ns; therefore, this means that for I2CBAUD > 1 I2CBAUD==1, which means 2 ODBAUD. Therefore, (2+1)*200=600ns low, with 2*200=400ns high for 1 us period, to match Fast Mode Plus (FM+). For Fast Mode (FM), I2CBAUD is normally 11 (giving 2.6 us) or is 6 (giving 2.4 us).
27-25 SKEW	Skew Number of FCLK counts for an SDA change after SCL for I3C push-pull; this is in addition to the roundtrip of the SCL line from the pad back to the module (3 ns to 4 ns or more). - SKEW is normally not needed, so assign SKEW=0. - SKEW is only used if SDA is not naturally skewing from an SCL change. - I2C automatically skews SDA (but not PUR) to match I2C rules.
24 ODHPP	Open drain high push-pull If ODHPP=1, then Open-Drain High should be 1 PPBAUD count for I3C messages; otherwise ODHPP should be the same value as ODBAUD. Using an ODHPP that is the same value as ODBAUD is commonly used so that I2C devices do not see high.
23-16 ODBAUD	Open drain baud rate ODBAUD= (number of PPBAUD counts) - 1. ODBAUD should not be 0 (not the same as push-pull). This would normally go for 200 ns (see I2CBAUD for I2C counts). When used with ODHPP, this gives 250 ns per clock in I3C. Therefore, if PPBAUD gives 12 MHz, then 1 PPBAUD is ½ of 12MHz or 41.67 ns, and then to get around 200 ns, use 5-1=4 for ODBAUD.
15-12	Push-Pull low

Table continues on the next page...

Table continued from the previous page...

Field	Function
PPLOW	Adder for push-pull low, to create a duty-cycle with a longer low period, with up to 15 more FCLKs low than high. PPLOW=0 means a 50:50 duty cycle.
11-8 PPBAUD	Push-pull baud rate The FCLK Counter for each push-pull low and normally high period. So, PPBAUD=0 if run at $\frac{1}{2}$ input FCLK speed (for example, a 24 MHz FCLK yields a 12 MHz SCL, because each FCLK is SCL Low or SCL High). Note the use of duty-cycle via Push-Pull Low (PPLOW). For example, 24 MHz with 50:50 duty cycle is 12 MHz, but with PPLOW adding 3 more low beats, the push-pull baud rate becomes 4.8 MHz (from 24 MHz/5 beats).
7 —	Reserved
6 ODSTOP	Open drain stop If ODSTOP=1, then STOP will be emitted at open-drain speeds even for I3C messages. This can be useful for legacy devices, to ensure that the legacy devices see the STOP.
5-4 HKEEP	High-Keeper Indicates how High-Keeper is to be supported. NOTE Your specific device may not support any or all of these High-Keeper methods. Refer to the chip-specific section for this module, at the beginning of this chapter. 00b - NONE. Use PUR (Pull-Up Resistor). Hold SCL High. 01b - WIRED_IN. Wired-in High Keeper controls; use pin_HK (High Keeper) controls. 10b - PASSIVE_SDA. Can Hi-Z (high impedance) for Bus Free (IDLE) and hold. 11b - PASSIVE_ON_SDA_SCL. Can Hi-Z (high impedance) both for Bus Free (IDLE), and can Hi-Z SDA for hold.
3 DISTO	Disable Timeout This timeout detects application errors. • If DISTO=1, then it disables the timeout (if the timeout is configured). • If the master has been left in a state other than STOPped for more than 100 us (because 10 KHz is the slowest allowed I3C speed), then the timeout will send an MERRWARN (interrupt). • To prevent the MERRWARN interrupt when doing development or testing, write 1 to DISTO to disable the timeout. • In systems that support timeouts, timeout will be disabled automatically during debug.
2 —	Reserved
1-0 MSTENA	Master enable Indicates if the master is enabled and what “states” are allowed for it (the master).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - MASTER_OFF. Master is off (is not enabled). If MASTER_OFF is enabled, then the I3C module can only use slave mode. 01b - MASTER_ON. Master is on (is enabled). When used from start-up, this I3C module is master by default (the main master). The module will control the bus unless the master is handed off. If the master is handed off, then MSTENA must move to 2 after that happens. The handoff means emitting GETACCMST and if accepted, the module will emit a STOP and set the MSTENA bit to 2 (or 0). 10b - MASTER_CAPABLE. The I3C module is master-capable; however the module is operating as a slave now. When used from the start, the I3C module will start as a slave, but will be prepared to switch to master mode. To switch to master mode, the slave emits an Master Request (MR), or gets a GETACCMST CCC command and accepts it (to switch on the STOP). 11b - Reserved

52.7.3 Slave Configuration Register (SCONFIG)

Offset

Register	Offset
SCONFIG	4h

Function

Contains fields that must be configured before the module is activated.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SADDR								Reserv	BAMATCH						
W									ed							
Reset	0	0	0	0	0	0	0	u	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							OFFLI	Reserv	Reserv	Reserv	DDRO	S0IGN	MATC	NACK	SLVE
W								NE	ed	ed	ed	K	ORE	HSS		NA
Reset	u	u	u	u	u	u	0	0	u	0	0	0	0	0	0	0

Fields

Field	Function
31-25	Static address

Table continues on the next page...

Table continued from the previous page...

Field	Function
SADDR	SADDR sets the I2C 7-bit static address.
24 —	Reserved
23-16 BAMATCH	Bus available match This is the Bus Available condition match value for the current slow clock. BAMATCH provides the count of the slow clock to count out 1 us (or more), to allow an In-Band Interrupt (IBI) to drive SDA low when the master is not doing so. The maximum width and maximum values are controlled by the I3C module.
15-10 —	Reserved
9 OFFLINE	Offline If OFFLINE=1 when the slave enable (SCONFIG.SLVENA) is set to 1, then the I3C module will wait for either 60 us of bus quiet or a High Data Rate (HDR) Exit Pattern. This ensures that the bus is not in High Data Rate (HDR) mode, and so can safely track Single Data Rate (SDR) mode.
8 IDRAND	ID random • If IDRAND=1, then SIDPARTNO.PARTNO is a random value. • If IDRAND=0, then SIDPARTNO.PARTNO is a part number and an instance.
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 DDROK	Double Data Rate OK If DDROK=1, then HDR-DDR messaging is allowed; set the corresponding SIDEXT.BCR bit (Bus Characteristics Register bit in Slave ID Extension Register) to say that High Data Rate (HDR) is available, and configure the corresponding HDRCAP HDR-DDR bit to allow using Double Data Rate (DDR) mode. NOTE The DDROK bit must be set before the Slave can connect to the I3C bus; the Slave peripheral will indicate to the Master whether it (the slave) can support the feature during dynamic address assignment.
3 S0IGNORE	S0/S1 errors ignore

Table continues on the next page...

Table continued from the previous page...

Field	Function
	If S0IGNORE=1, then the slave will not detect S0 or S1 errors, and therefore the slave will not lock up, and will wait for an exit pattern. S0IGNORE=1 should only be used when the I3C bus will not use a High Data Rate (HDR) mode.
2 MATCHSS	Match START or STOP If MATCHSS=1, then the START and STOP sticky STATUS bits will only be set if SSTATUS.MATCHED is set. This allows START and STOP to be used to detect the end of a message to/from this slave.
1 NACK	Not acknowledge If NACK=1, then the slave will NACK all requests to it, except a Common Command Code (CCC) broadcast. NACK=1 should be used with caution, because the master may decide that the slave is missing, if NACK is overused.
0 SLVENA	Slave enable - If SLVENA=1, then a slave can operate on the I2C or I3C bus. - If SLVENA=0, then a slave will ignore the I2C or I3C bus. SLVENA should not be set before registers like SCONFIG (SIDPARTNO, SIDEXT, and others) are set, because these registers affect the data to/from the master. Slave enable is normally only configured just one time before the I3C bus comes up. If slave enable is used at other times, see Hot-Join. In the case of Hot-Join, the Hot-Join bit (SCAPABILITIES.IBI_MR_HJ) should be set before setting the slave enable bit (SLVENA), so that the device does not see a START or STOP incorrectly.

52.7.4 Slave Status Register (SSTATUS)

Offset

Register	Offset
SSTATUS	8h

Function

Not all bits are used if the module only acts as a slave. The Slave Status register is used to indicate both sticky status for interrupts, as well as “states” and “modes” related to the I3C bus. The fields are divided into current activity, interrupt maskable actions, then states and modes on the bus.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMECTRL		ACTSTATE		HJDIS	Reserv ed	MRDIS	IBIDIS	Reserved		EVDET		Reserv ed	EVEN T	CHAN DLED	HDRM ATCH
W																
Reset	0	0	0	0	0	u	0	0	0	0	u	u	u	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRW ARN	CCC	DACH G	TXNO TFU...	RX_ PEND	STOP	MATC HED	STAR T	Reserv ed	STHD R	STDA A	STRE QWR	STRE QRD	STCC CH	STMS G	STNO TST...
W																
Reset	0	0	0	1	0	0	0	0	u	0	0	0	0	0	0	0

Fields

Field	Function
31-30 TIMECTRL	Time control Indicates if time control is currently enabled. 00b - NO_TIME_CONTROL. No time control is enabled 01b - Reserved 10b - ASYNC_MODE. Asynchronous standard mode (0) is enabled 11b - Reserved
29-28 ACTSTATE	Activity state from Common Command Codes (CCC) 00b - NO_LATENCY. Normal bus operations 01b - LATENCY_1MS. 1 ms of latency 10b - LATENCY_100MS. 100 ms of latency 11b - LATENCY_10S. 10 seconds of latency
27 HJDIS	Hot-Join is disabled HJDIS=1 if Hot-Join is disabled at this time. While Hot-Join is disabled, CTRL requests will be held off (not responded to).
26 —	Reserved
25 MRDIS	Master requests are disabled MRDIS=1 if Master Requests are disabled at this time. While Master Requests are disabled, CTRL requests will be held off (not responded to).
24 IBIDIS	In-Band Interrupts are disabled IBIDIS=1 if In-Band Interrupts are disabled at this time. While In-Band Interrupts are disabled, CTRL requests will be held off (not responded to).

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-22 —	Reserved
21-20 EVDET	Event details Current details of the last (EVENT=1) or pending event. 00b - NONE. No event or no pending event 01b - NO_REQUEST. Request not sent yet. Either there was no START yet, or is waiting for Bus-Available or Bus-Idle (HJ). 10b - NACKED. Not acknowledged(Request sent and NACKed); the module will try again. 11b - ACKED. Acknowledged (Request sent and ACKed), so Done (unless the time control data is still being sent).
19 —	Reserved
18 EVENT	Event For a Slave, a pending In-Band Interrupt (IBI), P2P (peer-to-peer), MR (Master Request), or Hot-Join (HJ) has been sent as requested. See the upper status register fields for details. Note that for an in-band interrupt, this occurs on the ACK if no IBI byte is present, and after the 1 IBIDATA byte has been sent. So, if time control is used, those time control bytes (in IBI) will go out after this EVENT is signaled.
17 CHANDLED	Common-Command-Code handled A Common Command Code (CCC) is being handled by the module. This is a notification only, but the result may be an updated SSTATUS register.
16 HDRMATCH	High Data Rate command match An HDR command matched this device's I3C Dynamic Address. The HDR command will be available as the 1st byte and RXPEND will be set (whether the command is read or write, the MSb of that command byte indicates if it is a read or a write command). If the HDR command is a read, and there are to-bus bytes waiting, then the command will be ACKed and the data sent back; otherwise the HDR command will be NACKed. Note that when HDRMATCH is set, the ERRWARN bit should be checked, because the HPAR error may have been encountered after signaling this HDR command (the parity is after the destination address and command).
15 ERRWARN	Error warning An error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the Slave Errors and Warnings Register (SERRWARN).
14 CCC	Common Command Code A Common-Command-Code (CCC) has been received, and is not handled by the I3C module. There are 2 types of Common Command Codes: • Broadcasted CCC, which will then also correspond with RXPEND and the 1st byte will be the CCC (command).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Direct CCC, which may never be directed to this device. If Direct CCC are directed to this device, then the TXSEND or RXPEND will be triggered, and the RXPEND will contain the command.
13 DACHG	DACHG The Slave Dynamic Address has been assigned, re-assigned, or reset (lost) and is now in that state of being valid or none.
12 TXNOTFULL	Transmit buffer is not full TXNOTFULL=1 when the To-bus buffer/FIFO can accept more data to go out. For all but External FIFO, this uses SDATACTRL.TXTRIG, which defaults to not-full. If DMA is enabled for TX, then it will also be signaled to provide more data.
11 RX_PEND	Received message pending Receiving a message from master, which is not being handled by the I3C module (not a Common Command Code (CCC), so is internally processed by the module). For all but External FIFO, this uses SDATACTRL.RXTRIG, which defaults to not-empty. If DMA is enabled for RX, then DMA will be signaled as well. RX_PEND will self-clear if data is read (from FIFO and non-FIFO sources).
10 STOP	Stop The STOP bit is from a stopped state being detected, meaning that a STOP state was present on the bus since the bus was last cleared. The STNOTSTOP state will also indicate if the I3C module is in stop mode. A fast STOP/START combination may not trigger the STOP status; for that case, START will always be set.
9 MATCHED	Matched An incoming header matched this device's I3C Dynamic or I2C Static address (if any) since the bus was last cleared.
8 START	Start START=1 if a START or repeated START was seen after the START bit was last cleared. This is not usually needed, but can be used for wake events.
7 —	Reserved
6 STHDR	Status High Data Rate STHDR=1 if the I3C bus is in HDR-DDR, HDR-TSP, or HDR-TSL modes, regardless of whether HDR mode is supported by this module or not, and regardless of whether the message is to this module or to some other module.
5 STDAA	Status Dynamic Address Assignment STDAA=1 if the I3C bus is in Enter Dynamic Address Assignment (ENTDAA) mode, regardless of whether this bus slave has or does not have a Dynamic Address.
4	Status request write

Table continues on the next page...

Table continued from the previous page...

Field	Function
STREQWR	STREQWR=1 if the request (REQ) in process is SDR write data from the master to this bus slave (or all I3C slaves), but not in Enter Dynamic Address Assignment (ENTDAA) mode. See Status High Data Rate (STHDR) for Double Data Rate (DDR) handling.
3 STREQRD	Status required STREQRD=1 • if the REQ in process is an SDR read from this slave • or if an In-Band Interrupt (IBI) is being pushed out Also see Status High Data Rate (STHDR) for Double Data Rate (DDR) handling.
2 STCCCH	Status Common Command Code Handler STCCCH=1 if a Common Command Code (CCC) message is being handled automatically.
1 STMSG	Status message STMSG=1 if this bus slave is listening to the bus traffic or responding. If STNOSTOP=1, then STMSG will be 0 when a non-matching address is seen, until the next repeated START or STOP occurs.
0 STNOTSTOP	Status not stop • when the bus is busy (has activity), STNOTSTOP=1 • when the I3C module is in a STOP condition, STNOTSTOP=0 Other SSTATUS bits may also be set when busy. Note that STNOTSTOP can also be true (=1) after an S0 or S1 error, where the I3C module is waiting for an Exit Pattern.

52.7.5 Slave Control Register (SCTRL)

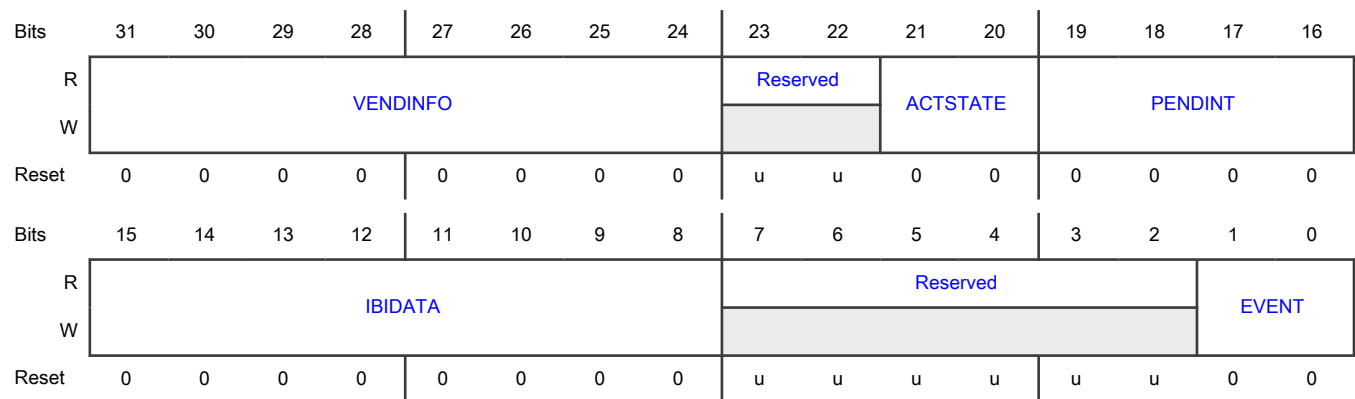
Offset

Register	Offset
SCTRL	Ch

Function

Contains controls for the active use of the I3C bus, like event generation (for example, interrupts to the master). The Slave Control register is used to activate various special operations for the Slave, but only if the module is configured to support those operations. This includes events such as IBI and GETSTATUS fields (except the Protocol error, which is automatically set).

Diagram



Fields

Field	Function
31-24 VENDINFO	Vendor information Should be set to the Vendor Reserved field that the GETSTATUS CCC will return. The vendor information should be maintained by the application, because the master will read this field. If VENDINFO is not configured, then the GETSTATUS field will always return 0.
23-22 —	Reserved
21-20 ACTSTATE	Activity state (of slave) Should be set to the slave's activity state that the GETSTATUS CCC (command code) will return as activity mode. The activity state should be maintained by the application, because the master will read this field. If the activity state is not configured, then the GETSTATUS command will always return 0.
19-16 PENDINT	Pending interrupt Should be set to the pending interrupt that the GETSTATUS CCC (command code) will return. The pending interrupt should be maintained by the application, because the master will read this field. If PENDINT=0 and - if an IBI interrupt is pending, then the GETSTATUS command will return 1. - if an IBI interrupt is not pending, then the GETSTATUS field will return 0.
15-8 IBIDATA	In-Band Interrupt Data Data byte to go with an In-Band Interrupt (IBI), if the module is enabled for in-band interrupts. If SCTRL.IBIDATA is enabled, then in-band interrupts are required.
7-2 —	Reserved
1-0 EVENT	EVENT If EVENT is set to non-0, it will request an event.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- After being requested, SSTATUS.EVENT and SSTATUS.EVDET will show the status as it progresses. - After completion, the EVENT field will automatically return to 0. - After EVENT is non-0, only 0 can be written to EVENT (to cancel), until the event processing is done (finished). 00b - NORMAL_MODE. If EVENT is set to 0 after was a non-0 value, event processing will cancel if the event processing has not yet started; if event processing has already been started, then event processing will not be cancelled. 01b - IBI. Start an In-Band Interrupt. This will try to push an IBI interrupt onto the I3C bus. If data is associated with the IBI, then the data will be read from the SCTRL.IBIDATA field. If time control is enabled, then this data will also include any time control-related bytes; additionally, the IBIDATA byte will have bit 7 set to 1 automatically (as is required for time control). The IBI interrupt will occur after the 1st (mandatory) IBIDATA, if any. 10b - MASTER_REQUEST. Start a Master-Request. 11b - HOT_JOIN_REQUEST. Start a Hot-Join request. A Hot-Join Request should only be used when the device is powered on after the I3C bus is already powered up, or when the device is connected using hot insertion methods (the device is powered up when it is physically inserted onto the powered-up I3C bus). The hot join will wait for Bus Idle, and SCTRL.EVENT=HOT_JOIN_REQUEST must be set before the slave enable (SCONFIG.SLVENA).

52.7.6 Slave Interrupt Set Register (SINTSET)

Offset

Register	Offset
SINTSET	10h

Function

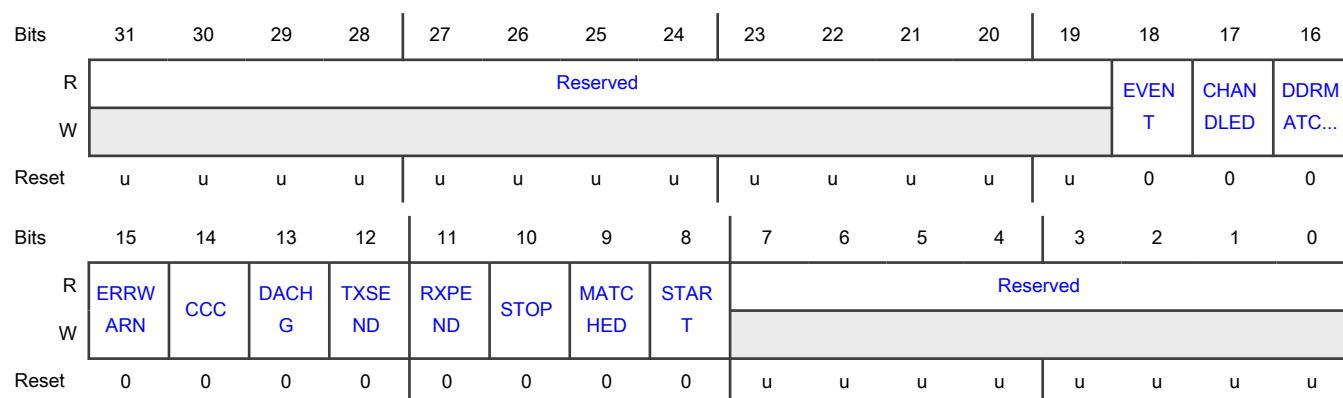
Sets interrupt enables for select STATUS bits. Reading this register (SINTSET) returns the status of the interrupt enables.

- To activate an interrupt enable, write 1 to its field in this register (SINTSET).
- To disable an interrupt enable, write 1 to the appropriate bit in the Interrupt Clear Register (SINTCLR). Writing 0 to the interrupt enable in this register (SINTSET) does not disable the interrupt.

The Interrupt registers allow masking interrupt sources, as well as checking which interrupts have activated. The normal method is to enable an interrupt, and then once the interrupt fires, the interrupt is either cleared by writing the SSTATUS register or cleared by action on the corresponding data register. The Interrupt is level held, meaning the interrupt stays set until the cause is cleared by some method. The module prevents races, so that if a new event comes in, that new event will not be lost.

- SINTSET sets interrupt enables for STATUS bits. Reading the SINTSET register returns the status of the interrupt enables.
- SINTCLR clears interrupt enables for STATUS bits.
- SINTMASKED returns the value of the STATUS bits ANDed with their interrupt enables.

Diagram



Fields

Field	Function
31-19 —	Reserved
18 EVENT	Event interrupt enable
17 CHANDLED	Common Command Code (CCC) (that was handled by I3C module) interrupt enable
16 DDRMATCHED	Double Data Rate (DDR) interrupt enable
15 ERRWARN	Error/warning interrupt enable
14 CCC	Common Command Code (CCC) (that was not handled by I3C module) interrupt enable
13 DACHG	Dynamic address change interrupt enable
12 TXSEND	Transmit interrupt enable
11 RXPEND	Receive interrupt enable
10 STOP	Stop interrupt enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 MATCHED	Match interrupt enable
8 START	Start interrupt enable
7-0 —	Reserved

52.7.7 Slave Interrupt Clear Register (SINTCLR)

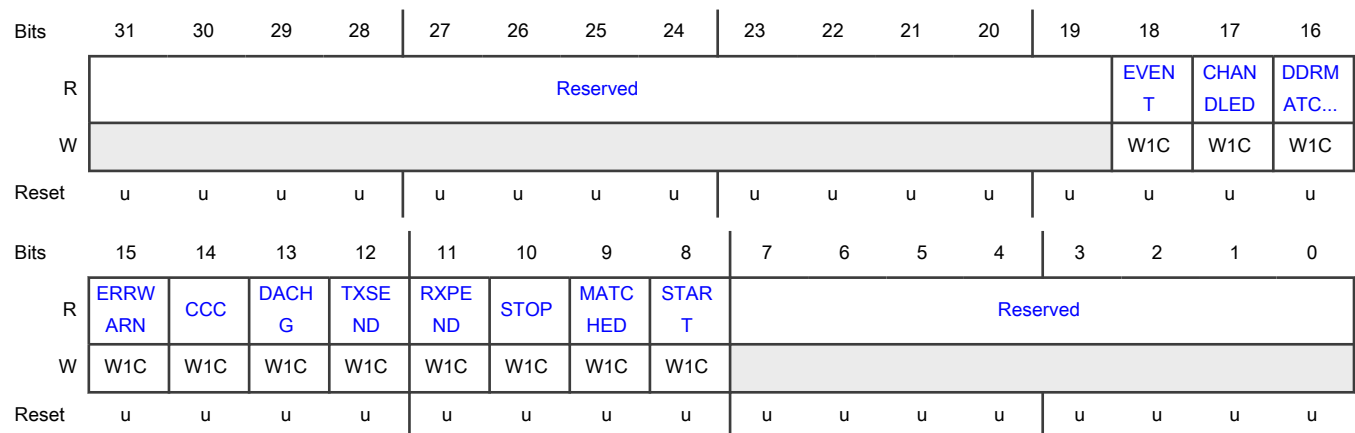
Offset

Register	Offset
SINTCLR	14h

Function

Clear interrupt enables for select STATUS bits. To clear an interrupt enable, write 1 to the corresponding bit in this register (SINTCLR); writing 0 has no effect.

Diagram



Fields

Field	Function
31-19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 EVENT	EVENT interrupt enable clear
17 CHANDLED	CHANDLED interrupt enable clear
16 DDRMATCHED	DDRMATCHED interrupt enable clear
15 ERRWARN	ERRWARN interrupt enable clear
14 CCC	CCC interrupt enable clear
13 DACHG	DACHG interrupt enable clear
12 TXSEND	TXSEND interrupt enable clear
11 RXPEND	RXPEND interrupt enable clear
10 STOP	STOP interrupt enable clear
9 MATCHED	MATCHED interrupt enable clear
8 START	START interrupt enable clear
7-0 —	Reserved

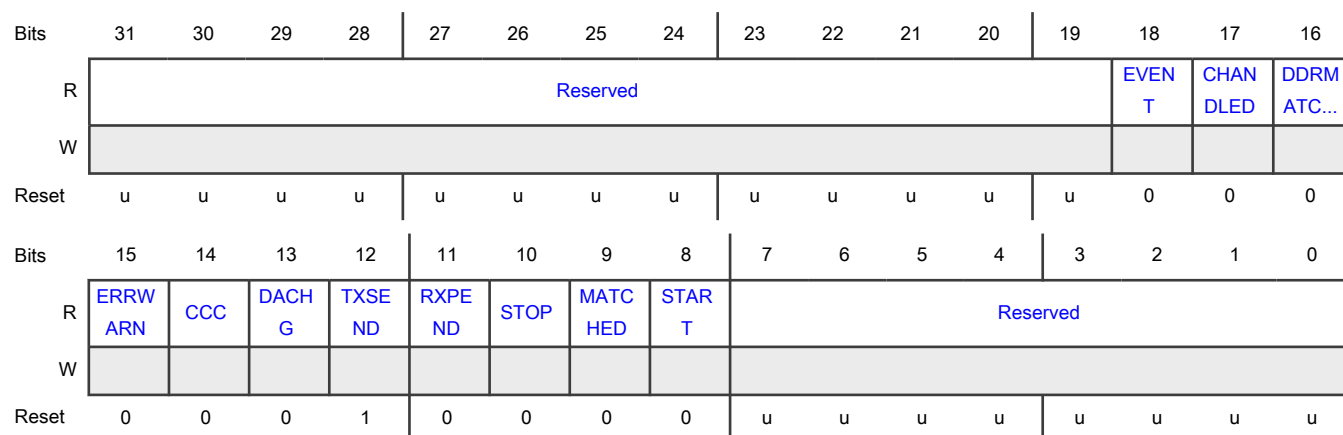
52.7.8 Slave Interrupt Mask Register (SINTMASKED)

Offset

Register	Offset
SINTMASKED	18h

Function

Returns the status of enabled interrupts (the value of STATUS ANDed with INTSET).

Diagram**Fields**

Field	Function
31-19 —	Reserved
18 EVENT	EVENT interrupt mask
17 CHANDLED	CHANDLED interrupt mask
16 DDRMATCHED	DDRMATCHED interrupt mask
15 ERRWARN	ERRWARN interrupt mask
14 CCC	CCC interrupt mask
13 DACHG	DACHG interrupt mask
12 TXSEND	TXSEND interrupt mask
11	RXPEND interrupt mask

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXPEND	
10 STOP	STOP interrupt mask
9 MATCHED	MATCHED interrupt mask
8 START	START interrupt mask
7-0 —	Reserved

52.7.9 Slave Errors and Warnings Register (SERRWARN)

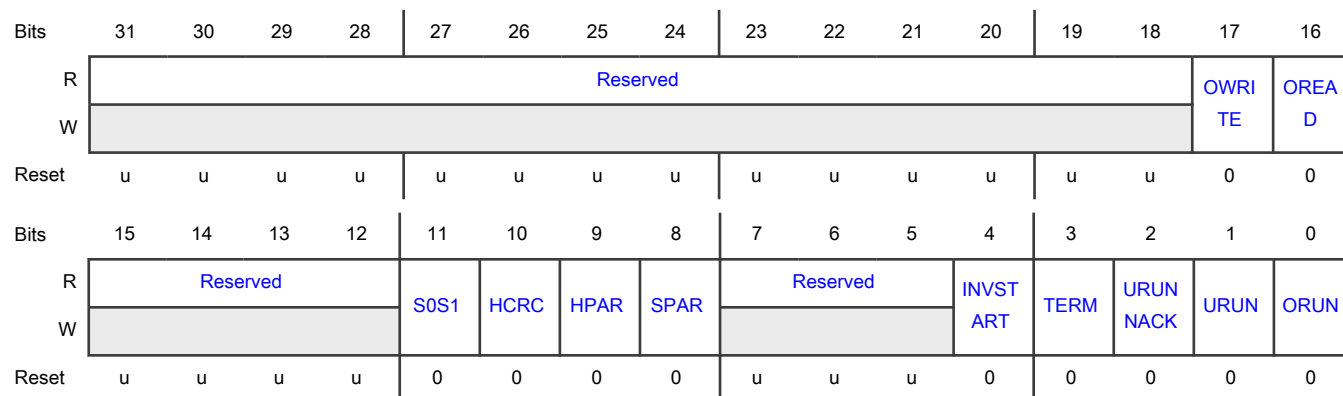
Offset

Register	Offset
SERRWARN	1Ch

Function

Contains errors and warnings from I3C/I2C protocol, which includes internal issues such as overrun and underrun, detected errors and conditions like parity errors, CRC errors, and read terminations by the Master. Related to SSTATUS.ERRWARN bit and SINTSET.ERRWARN interrupt.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 OWRITE	Over-write error The SWDATAB/BE register was written when FULL.
16 OREAD	Over-read error The SRDATAB register was read for more bytes than were available by the application.
15-12 —	Reserved
11 S0S1	S0 or S1 error A S0 or S1 error has occurred and the slave is locked and waiting for an HDR Exit Pattern. Writing 1 to S0S1 will cause the module to release the lock, but that should be used with great care. Normally, S0S1 will clear automatically when an Exit Pattern is detected. Therefore, writing 1 to S0S1 should only be used under controlled circumstances, to avoid problems. Before starting to operate normally, the module will then wait for a START (or repeated START) or STOP.
10 HCRC	HDR-DDR CRC error HDR-DDR Cyclic Redundancy Check (CRC) error on a message from the master. This calls into question the data from the whole DDR command frame. Note that this includes an HDR Restart or Exit being issued before an HDR-DDR message from master has finished.
9 HPAR	HDR parity error HDR Parity error or framing error on a message from the master. Note that the corresponding command or data that had the error will normally be in the RX buffer (which can be read using the Slave Read Data Byte Register (SRDATAB)).
8 SPAR	SDR parity error SDR Parity error on a message from the master. This will also set the GETSTATUS Protocol Error sticky bit (which is cleared after a GETSTATUS read).
7-5 —	Reserved
4 INVSTART	Invalid start error Invalid start with SCL falling while SDA=1 is in a STOP condition.
3 TERM	Terminated error The master terminated a read from a slave when an END was not set (on the same or previous read).
2 URUNNACK	Underrun and Not Acknowledged (NACKed) error The internal to-bus buffer/FIFO was underrun in the read header and so the module NACKed the header.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 URUN	Underrun error The internal to-bus buffer/FIFO was underrun during data read (the application did not provide the data fast enough). The END bit or register should be used if that was the last one.
0 ORUN	Overrun error The internal from-bus buffer/FIFO was overrun (too many characters are coming in and cannot be processed by the the user application fast enough).

52.7.10 Slave DMA Control Register (SDMACTRL)

Offset

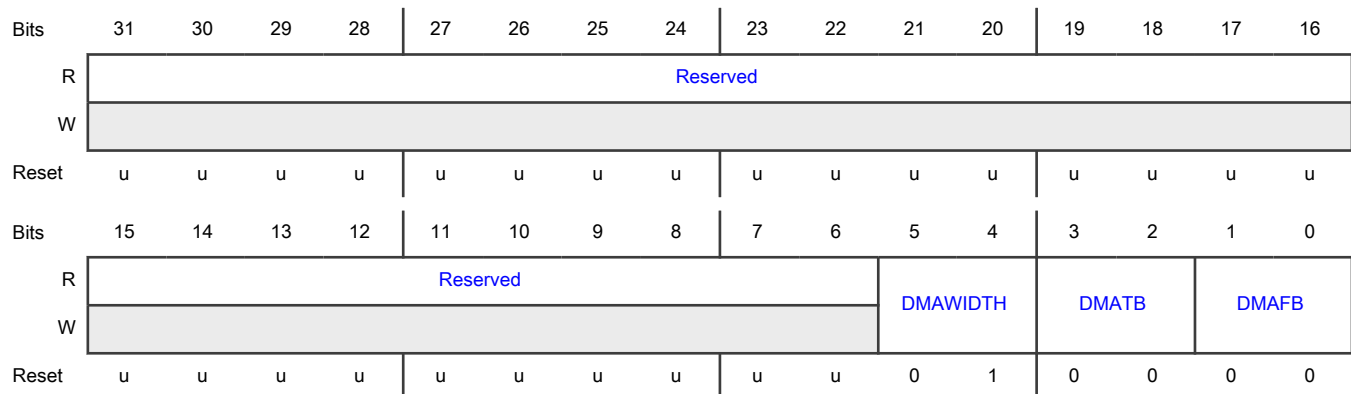
Register	Offset
SDMACTRL	20h

Function

The DMA Control register allows for DMA to be used for inbound messages and outbound messages. The DMA Control register is limited in value for Slave use, because the Slave has to be reactive to what happens. Two common use models:

- From-bus collection to avoid being overrun: The SCONFIG.MATCHSS bit is set, and then the processor enables the interrupts for START, and STOP, as well as enabling the DMA to collect the data. The START or STOP interrupt will only occur after a message is directed to the Slave (MATCHED bit set), and the DMA copied data can then be examined.
- To-bus for larger reads: Because I3C and I2C reads are preceded by a write that indicates what will be read (or in response to an IBI from the Slave), the DMA can be used to push through the data.
 - For I3C, the last value needs to be handled by the processor, unless the DMA moves wider words and is able to set the END bit (i.e., 16-bit values when in byte mode or 32-bit values when in half-word mode).
 - For I2C, the last value is determined by the Master, so the DMA may end early or may run out when the Master still wants more.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-4 DMAWIDTH	Width of DMA operations The width of DMA operations, if configured to allow half-word data access. 00b - BYTE 01b - BYTE_AGAIN 10b - HALF_WORD: Half word (16 bits). This will make sure that 2 bytes are free/available in the FIFO. 11b - Reserved
3-2 DMATB	DMA Write (To-bus) trigger If enabled with 1 or 2, DMATB will start a request DMA on a TX trigger; see the Slave Data Control Register (SDACTRL). DMATB will request until full unless the DMA is set up as a trigger. DMATB will cancel on MSTATUS.ERRWARN. 00b - NOT_USED. DMA is not used 01b - ENABLE_ONE_FRAME. DMA is enabled for 1 Frame (ended by DMA or terminated). DMATB auto-clears on a STOP or START (see the Match START or STOP bit (SCONFIG.MATCHSS)). 10b - ENABLE. DMA is enabled until turned off. Normally, ENABLE should only be used with Master Message mode.
1-0 DMAFB	DMA Read (From-bus) trigger If enabled with 1 or 2, DMAFB will request DMA on RX trigger (see Slave Data Control Register (SDACTRL)). It will request until empty unless the DMA is set up as a trigger. DMAFB will cancel on SSTATUS.ERRWARN. 00b - DMA not used 01b - DMA is enabled for 1 frame. Auto-clears on STOP or repeated START. See the Match START or STOP bit (SCONFIG.MATCHSS). 10b - DMA enable. DMA is enabled until it is turned off.

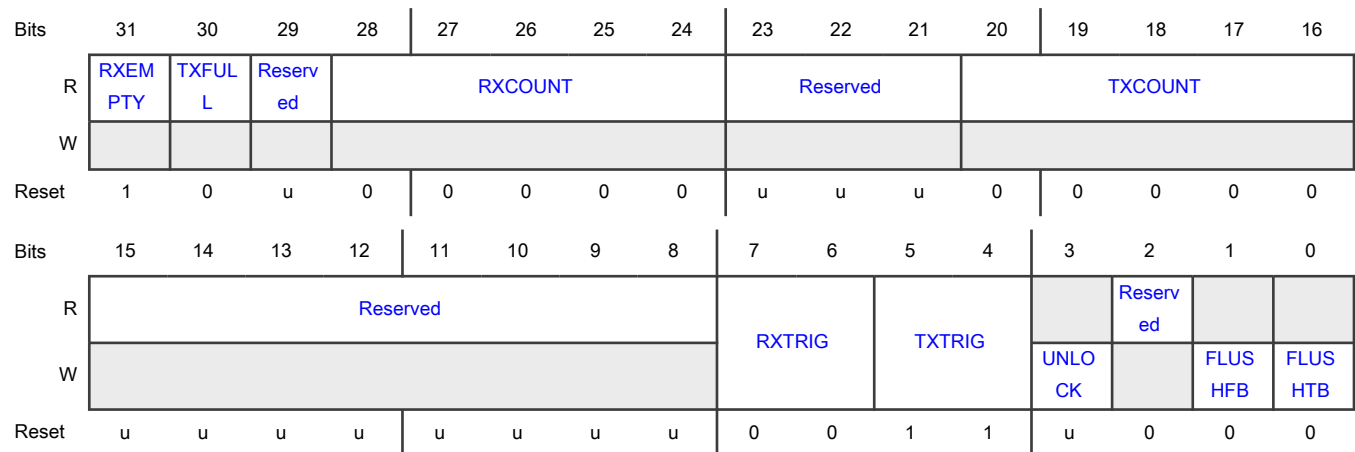
52.7.11 Slave Data Control Register (SDACTRL)**Offset**

Register	Offset
SDACTRL	2Ch

Function

Controls data buffering and FIFO behavior.

Diagram



Fields

Field	Function
31 RXEMPTY	RX is empty 0b - RX is not empty 1b - RX is empty
30 TXFULL	TX is full 0b - TX is not full 1b - TX is full
29 —	Reserved
28-24 RXCOUNT	Count of bytes in RX
23-21 —	Reserved
20-16 TXCOUNT	Count of bytes in TX
15-8 —	Reserved
7-6 RXTRIG	Trigger level for RX FIFO fullness Affects interrupts and DMA (if enabled). 00b - Trigger on not empty

Table continues on the next page...

Table continued from the previous page...

Field	Function
	01b - Trigger on $\frac{1}{4}$ or more full 10b - Trigger on $\frac{1}{2}$ or more full 11b - Trigger on $\frac{3}{4}$ or more full
5-4 TXTRIG	Trigger level for TX FIFO emptiness Affects interrupts and DMA (if enabled). The default is Trigger on 1 less than full or less (=3). 00b - Trigger on empty 01b - Trigger on $\frac{1}{4}$ full or less 10b - Trigger on $\frac{1}{2}$ full or less 11b - Trigger on 1 less than full or less (Default)
3 UNLOCK	Unlock - If 0 is written to UNLOCK, then the RXTRIG and TXTRIG fields cannot be changed on a write. - If 1 is written to UNLOCK, then the RXTRIG and TXTRIG fields can be changed on a write.
2 —	Reserved
1 FLUSHFB	Flushes the from-bus buffer/FIFO Not normally used.
0 FLUSHTB	Flush the to-bus buffer/FIFO Used when the master terminates a to-bus (read) message prematurely.

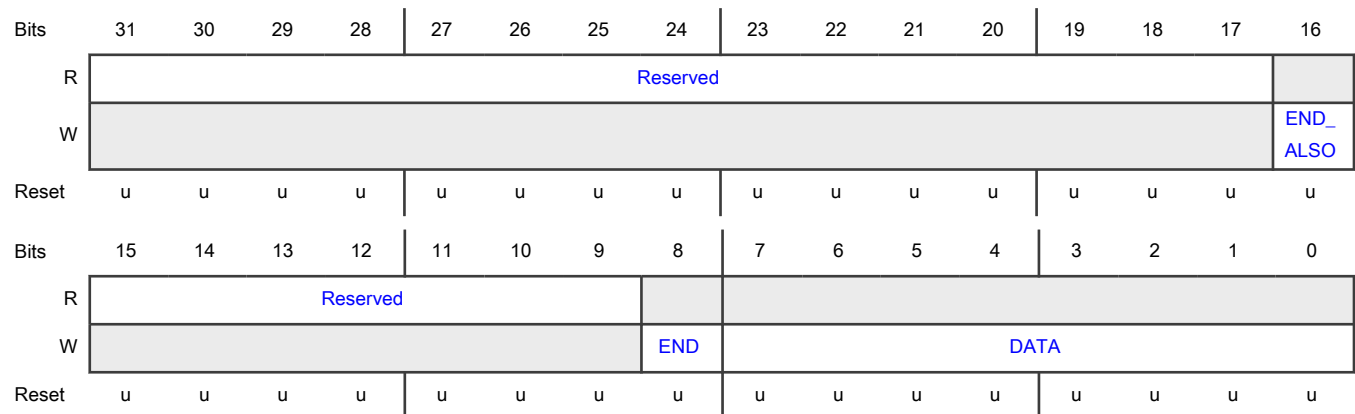
52.7.12 Slave Write Data Byte Register (SWDATAB)

Offset

Register	Offset
SWDATAB	30h

Function

Write a byte of data, using the 9th bit to mark as the end (the last byte of the message). This register allows writing a byte to the bus (to Master) unless an external FIFO is used. This takes a byte and an end-of-data (last) marker bit. A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the SSTATUS register.

Diagram**Fields**

Field	Function
31-17 —	Reserved
16 END_ ALSO	End also • If END_ALSO=1, then this bit marks the last byte of the message • If END_ALSO=0, then there are more bytes in the message END_ALSO is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END_ALSO must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.
15-9 —	Reserved
8 END	End • If END=1, then this bit marks the last byte of the message • If END=0, then there are more bytes in the message END is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.
7-0 DATA	The data byte to send to the master

52.7.13 Slave Write Data Byte End (SWDATABE)

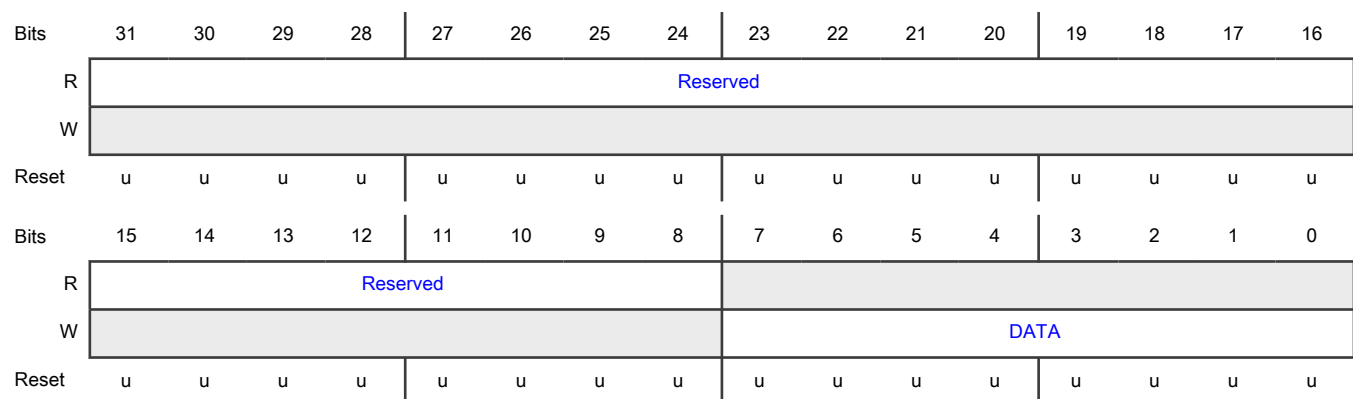
Offset

Register	Offset
SWDATABE	34h

Function

Write a byte of data, which is the end (the last byte of the message). Write a byte just like SWDATAB, but mark as end-of-data (last byte). For HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the SSTATUS register.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-0 DATA	The data byte to send to the master

52.7.14 Slave Write Data Half-word Register (SWDATAH)

Offset

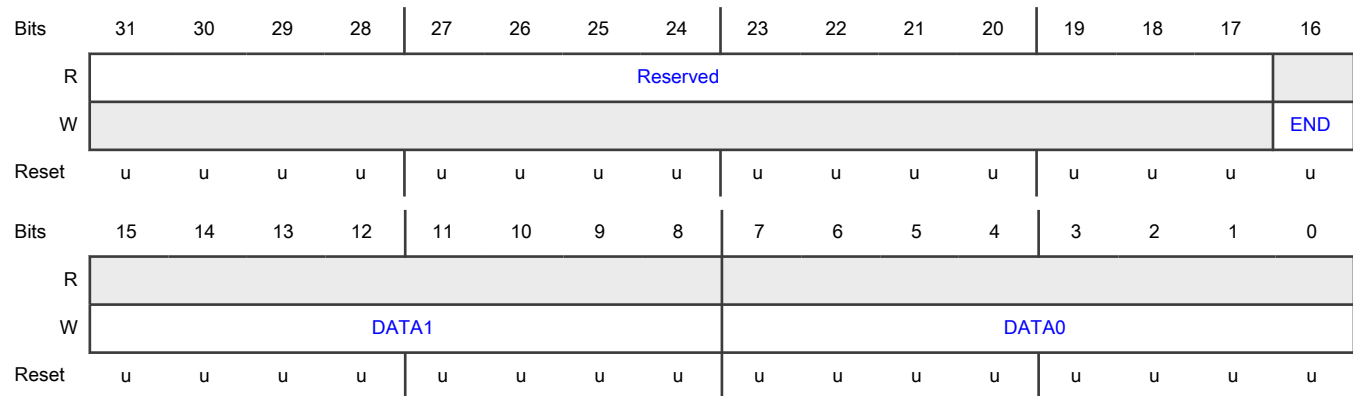
Register	Offset
SWDATAH	38h

Function

Write a half-word of data, using the 16th bit to mark as the end (the last byte of the half-word is the end). This register allows writing a half-word (pair of bytes) to the bus unless an external FIFO is used. This takes a half-word, which will send out the Low byte and then the High byte. An end-of-data (last) marker bit is allowed (or must be 0). A half-word should not be written

unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the SDATACTRL register.

Diagram



Fields

Field	Function
31-17 —	Reserved
16 END	End of message - If END=1, then END bit marks the last byte of the message - If END=0, then there are more bytes in the message For this register, this always marks DATA1 as the end. END is required to be used in I3C, but is optional in I2C. For HDR-DDR (Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.
15-8 DATA1	The 2nd byte to send to the master
7-0 DATA0	The 1st byte to send to the master

52.7.15 Slave Write Data Half-word End Register (SWDATAHE)

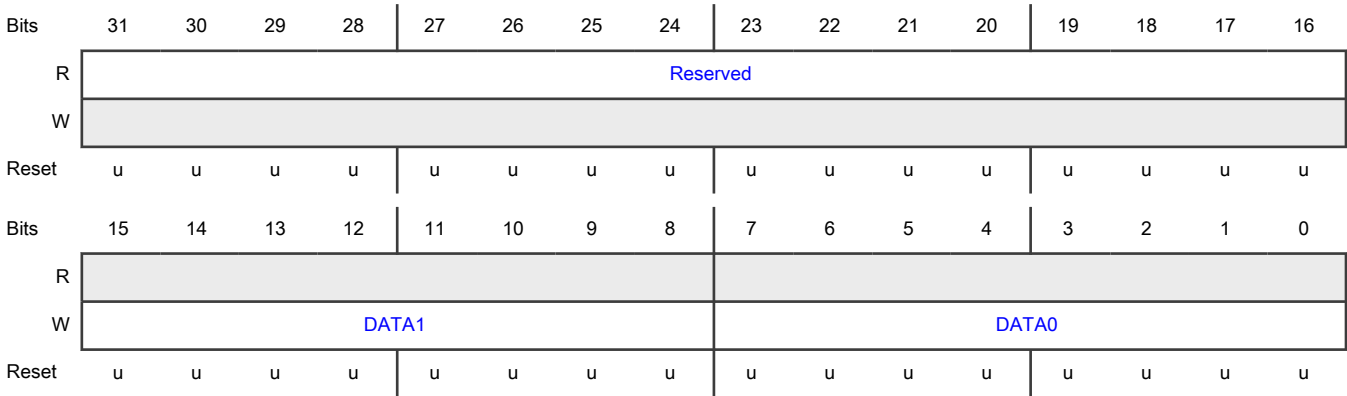
Offset

Register	Offset
SWDATAHE	3Ch

Function

Write a half-word of data, which is the end (the last byte of the half-word is the end). Write a half-word (byte pair) just like MWDATAH, but mark the 2nd bytes as end-of-data (last byte). For HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the SDATACTRL register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 DATA1	The 2nd byte to send to the master
7-0 DATA0	The 1st byte to send to the master

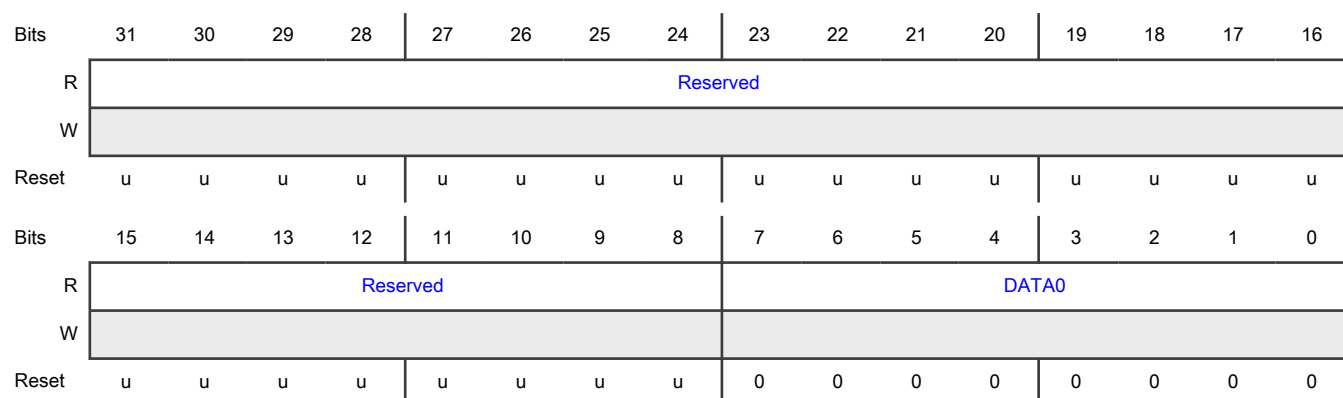
52.7.16 Slave Read Data Byte Register (SRDATAB)

Offset

Register	Offset
SRDATAB	40h

Function

Read a byte of data. This register allows reading a byte from the bus (Master). A byte should not be read unless there is data waiting, as indicated by the RXPEND bit being set in the SSTATUS register.

Diagram**Fields**

Field	Function
31-8 —	Reserved
7-0 DATA0	Byte read from the master

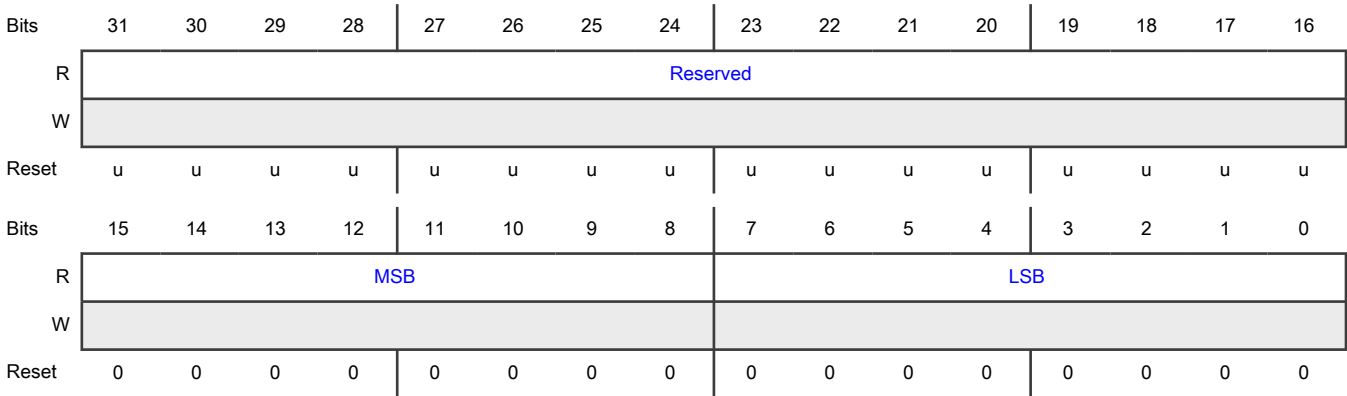
52.7.17 Slave Read Data Half-word Register (SRDATAH)**Offset**

Register	Offset
SRDATAH	48h

Function

Read a half-word of data. This register allows reading a half-word (byte pair) written by the Slave on an SDR Read or DAA or DDR. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. A half-word should not be read unless there is at least 2 bytes of data waiting, as indicated the RX FIFO level trigger or RXCOUNT available space in the SDATACTRL register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 MSB	The 2nd byte read from the slave
7-0 LSB	The 1st byte read from the slave

52.7.18 Slave Capabilities Register (SCAPABILITIES)

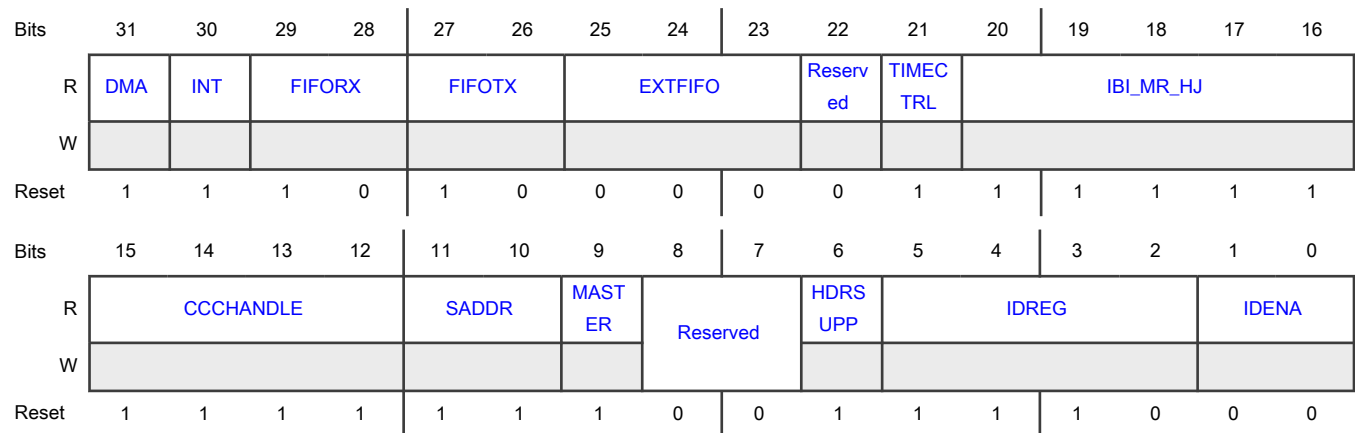
Offset

Register	Offset
SCAPABILITIES	60h

Function

Indicates which features are available and supported in this I3C module, including master and/or slave capabilities, HDR modes, and others.

Diagram



Fields

Field	Function
31 DMA	DMA 0b - DMA is not supported 1b - DMA is supported
30 INT	Interrupt 0b - Interrupts are not supported 1b - Interrupts are supported.
29-28 FIFORX	FIFO receive Indicates if RX (from-bus) is enabled and what size it is. 00b - FIFO_2BYTE. 2 (or 3)-byte RX FIFO, the default FIFO receive value (FIFORX) 01b - FIFO_4BYTE. 4-byte RX FIFO 10b - FIFO_8BYTE. 8-byte RX FIFO 11b - FIFO_16BYTE. 16-byte RX FIFO
27-26 FIFOTX	FIFO transmit Indicates if TX (to-bus) is enabled and what size it is. 00b - FIFO_2BYTE. 2-byte TX FIFO, the default FIFO transmit value (FIFOTX) 01b - FIFO_4BYTE. 4-byte TX FIFO 10b - FIFO_8BYTE. 8-byte TX FIFO 11b - FIFO_16BYTE. 16-byte TX FIFO
25-23 EXTFIFO	External FIFO Indicates whether External FIFOs are enabled. If External FIFOs are not enabled, then check FIFOTX and FIFORX for the internal FIFO.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	001b - STD_EXT_FIFO: Standard available/free external FIFO 011b - Reserved
22 —	Reserved
21 TIMECTRL	Time control Specifies if any time-control type is supported or not. 0b - NO_TIME_CONTROL_TYPE. No time control is enabled 1b - NO_TIME_CONTROL_TYPE. ATLEAST1_TIME_CONTROL: at least one time-control type is supported
20-16 IBI_MR_HJ	In-Band Interrupts, Master Requests, Hot Join events Indicates which events (In-Band Interrupts, Master Requests, Hot Join) are to be allowed. For example, if this field is set to 00011b, it means that IBI (bit 0) and IBI_HAS_DATA (bit 1) functionality are both enabled. • Bit 0: IBI: supports the application generating an In-Band Interrupt (IBI) • Bit 1: IBI_HAS_DATA: when bit 0=1, the In-Band Interrupt (IBI) has data from the SCTRL register • Bit 2: MASTER_REQUEST: supports the application generating a Master Request for a secondary master or a Peer-to-Peer • Bit 3: HOT_JOIN: supports the application generating Hot-Join • Bit 4: BAMATCH_FOR_BUS: use the BAMATCH register for bus-available timing
15-12 CCCHANDLE	Common Command Codes (CCC) handling Indicates who handles Common Command Codes (CCC) between I3C module and the user application. • Bit 0: BLOCK_HANDLE: the block (I3C module) handles events, activities, status, HDR, and if enabled for it, ID and static address related things • Bit 1: MAX_READ_WRITE: the block handles maximum read and write lengths, and max data speed • Bit 2: PENDINT_ACTSTATE: GETSTATUS CCC (command code) will return the SCTRL register's Pending interrupt (PENDINT) and activity state (ACTSTATE) fields • Bit 3: VENDINFO: GETSTATUS CCC will return the SCTRL register's VENDINFO (vendor info) bits
11-10 SADDR	Static address Indicates how the static address is handled. 00b - NO_STATIC. No static address 01b - STATIC. Static address is fixed in hardware 10b - HW_CONTROL. Hardware controls the static address dynamically (for example, from the pin strap) 11b - CONFIG. SCONFIG register supplies the static address

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 MASTER	Master Specifies if a master capability is supported or not. 0b - MASTERNOTSUPPORTED. Master capability is not supported. 1b - MASTERSUPPORTED. Master capability is supported.
8-7 —	Writing has no effect.
6 HDRSUPP	HDR support Indicates which High Data Rate (HDR) modes are supported. 0b - No HDR modes supported 1b - Double Data Rate mode supported
5-2 IDREG	ID register Bits indicate what is in the registers vs. what is in the hardware. • Bit 0: ID_INSTANCE: ID Instance is a register, and is used if there is no PARTNO register. • Bit 1: IDRAND: an ID Random field is available • Bit 2: DCR: a Device Characteristic Register (DCR) is available • Bit 3: BCR: a Bus Characteristics Register (BCR) is available
1-0 IDENA	ID 48b handler Indicates who handles the ID 48b value. 00b - APPLICATION. Application handles ID 48b 01b - HW. Hardware handles ID 48b 10b - HW_BUT. In hardware but the I3C module instance handles ID 48b. 11b - PARTNO. A part number register (PARTNO) handles ID 48b

52.7.19 Slave Dynamic Address Register (SDYNADDR)

Offset

Register	Offset
SDYNADDR	64h

Function

Contains the dynamic address after being assigned; otherwise =0.

The Slave Dynamic Address register is filled in with the assigned address once the Master has assigned it via SETDASA or ENTDAACCC commands. It will clear if the RESETDAA CCC is used. The current validity state is also indicated via the SSTATUS.DAVALID bit, which can be used to interrupt the processor

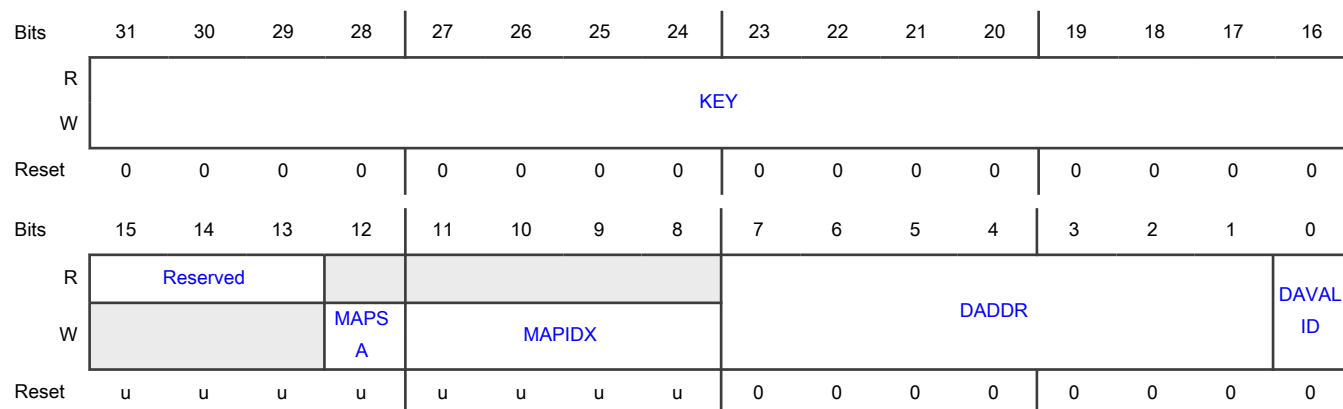
If configured to allow write, this is normally only used to restore the DA after a power-down (an ultra-low power state that loses power to peripherals but retains the DA somewhere else). This is not needed if state-retention flops are used for the DA. This mechanism only allows writes when Slave is disabled (and it will be ignored otherwise). If the Master uses RSTDAA or SETNEWDA, then it will over-ride this mechanism and cede (yield) to the master-assigned DA. Note that when enabling the Slave, the SCONFIG.OFFLINE bit should also be set. This will wait for evidence that the bus is not in I3C HDR mode; and will exit when an HDR Exit pattern is seen or when 60 usec has expired. This makes it safe to monitor START and STOP. If the application needs to do an IBI, then the application should either wait for a STOP (see STATUS) or make sure that 200 usec have gone by with no activity (no START or STOP) before the app emits the IBI.

The MAPIDX/MAPSA model allows writing additional DAs (and SAs) into a list (the number in the list is pre-configured); any DA or SA with DAVALID=0 are never matched. The additional DAs may be based on the bridge target CCC, or done by a move (read current DA and then copy into upper map, then invalidate the 0 map) when matching ENTDA over and over. Any mapped location can be invalidated by writing the MAPIDX and DAVALID=0. The mapped ones are not reset by the RSTDAA CCC. See the SMSGMAPADDR register.

To copy the base DA to the mapped set, the configuration has to be set up to allow it, otherwise they are distinct mechanisms.

- If used for the copy model, then a special reset feature is allowed. In this case, the SDYNADDR register is written with KEY=CB19 and the rest 0. This will clear the DA and then self-clear.
- If used in I3C mode, a base DA is required, so the last one should not be cleared (or writing one with DAVALID=1 is necessary).

Diagram



Fields

Field	Function
31-16 KEY	Key Must set to 0xA4D9 to write DADDR field (and set DAVALID bit to 1). Only writable when a slave is not enabled (for restoring after power-down sleep with auto-restore). The mapped locations and base may be written when a slave is enabled, but care should be taken to not do that when there are transactions on the I3C bus. KEY reads back as 1 if overwritten, else KEY is 0 if assigned by the master, including when the master changes it. If address mapping is allowed, then writing with KEY=0xCB19 is used to clear the base DA.
15-13 —	Reserved
12	Map a Static Address

Table continues on the next page...

Table continued from the previous page...

Field	Function
MAPSA	If MAPSA=1 on a write with MAPIDX!=0, then this sets a static address into the list;, otherwise a dynamic address is used.
11-8 MAPIDX	Mapped Dynamic Address Selects which mapped DA to write to, with MAPIDX=0 meaning the DA is not mapped (which is the normal use of write SDYNADDR if allowed). This allows for a list of matching DAs or SAs (I2C static addresses). Note that the mechanism is write-only.
7-1 DADDR	Dynamic address This is the assigned Dynamic Address, when DAVALID is 1.
0 DAVALID	DAVALID Determines if a Dynamic Address is assigned. 0b - DANOTASSIGNED. A Dynamic Address is not assigned 1b - DAASSIGNED. A Dynamic Address is assigned

52.7.20 Slave Maximum Limits Register (SMAXLIMITS)

Offset

Register	Offset
SMAXLIMITS	68h

Function

Indicates the limits set by the master (or the original requested limits). The maximum limits may or may not be enabled in the hardware design, including maximum read and write lengths. If those maximum read/write lengths are enabled, then the current setting (including default request) shows up in this register (SMAXLIMITS).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				MAXWR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				MAXRD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	u	u	u	u

Fields

Field	Function
31-28 —	Reserved
27-16 MAXWR	Maximum write length The maximum write length must be between 8 to 4095 (saturation). The application should not set the maximum write length longer than what the master sets the maximum write length to; the application should set the maximum write length to be less than what the master sets the maximum write length to
15-12 —	Reserved
11-0 MAXRD	Maximum read length The maximum read length must be between 16 to 4095 (saturation). The application should not set the maximum read length longer than what the master sets the maximum read length to; the application should set the maximum read length to be less than what the master sets the maximum read length to.

52.7.21 Slave ID Part Number Register (SIDPARTNO)

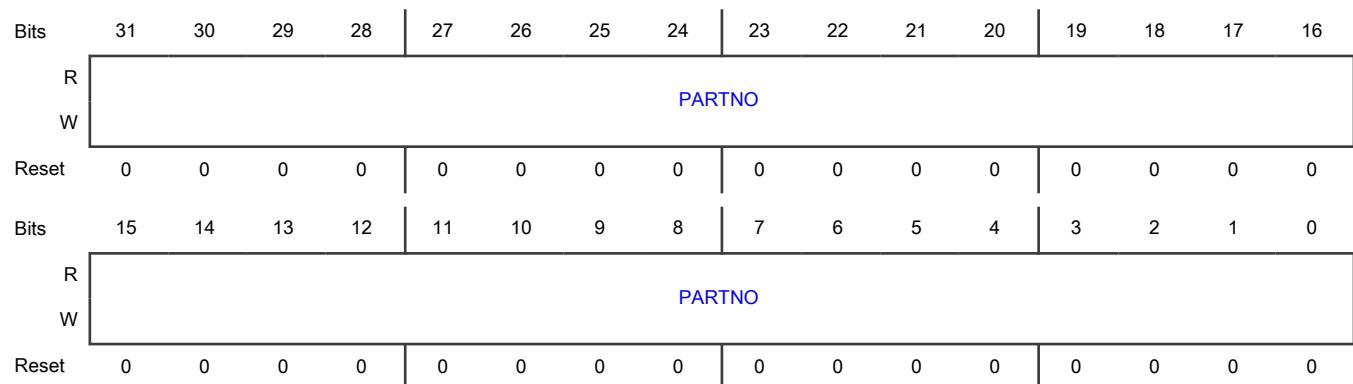
Offset

Register	Offset
SIDPARTNO	6Ch

Function

Allows an application to write the ID part-number. The application must write a value into the PARTNO field, because normally, 0 is not valid.

Diagram



Fields

Field	Function
31-0 PARTNO	Part number

52.7.22 Slave ID Extension Register (SIDEXT)

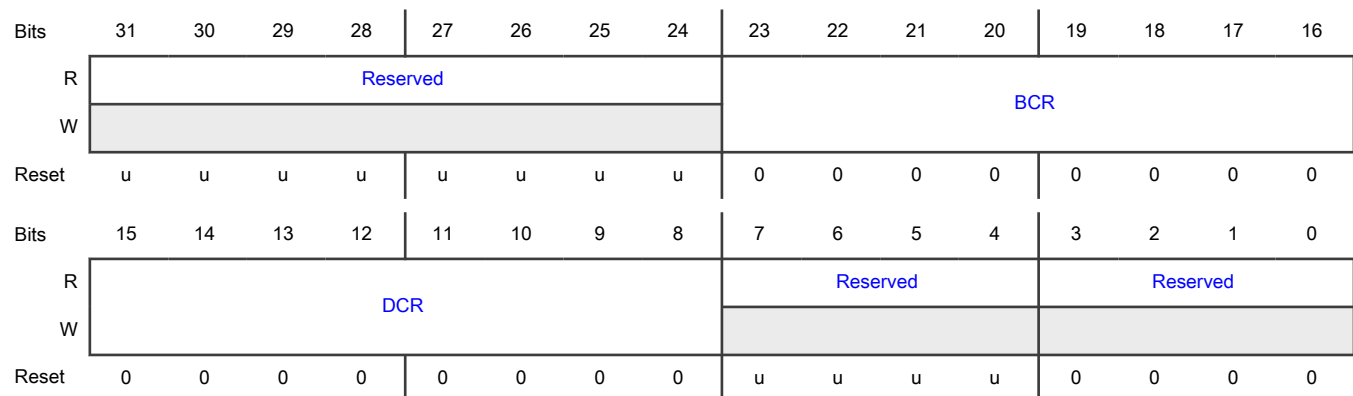
Offset

Register	Offset
SIDEXT	70h

Function

Allows an application to write the ID extension of the Device Characteristic Register (DCR) and/or the Bus Characteristics Register (BCR).

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 BCR	Bus Characteristics Register Set the Bus Characteristics Register (BCR) if configured for it. This controls features like Peer-to-Peer or Secondary Master, slow speed requirements, and others.
15-8 DCR	Device Characteristic Register Set the Device Characteristic Register (DCR) if configured for it.
7-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
3-0	Reserved
—	

52.7.23 Slave Vendor ID Register (SVENDORID)

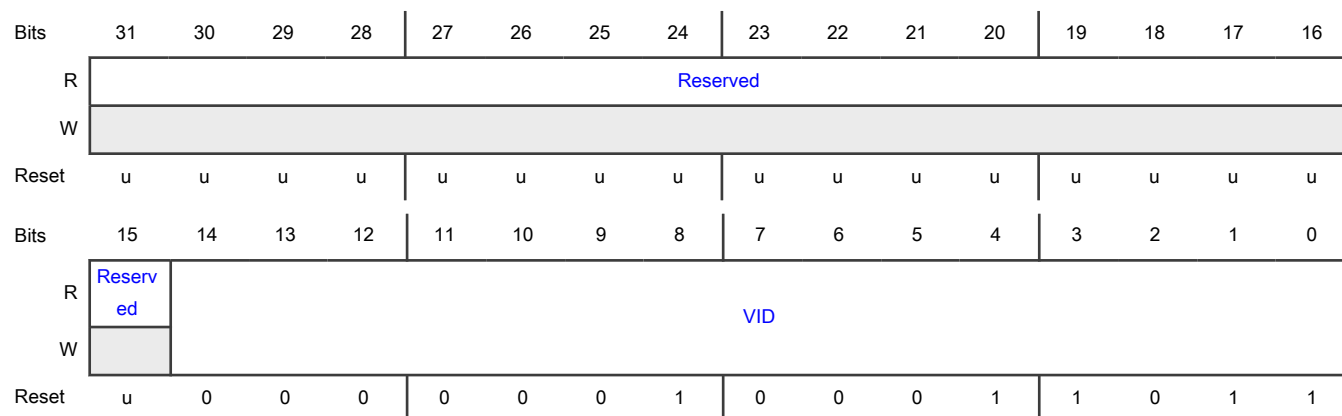
Offset

Register	Offset
SVENDORID	74h

Function

Allows an application to write the Vendor ID. The default will be set from the constant field and so usually will be the chip vendor. If using the chip vendor ID, the PARTNO must then not collide with other uses. The MIPI Vendor ID is available to all companies (MIPI membership is not required); to get a vendor ID, make a request at the mipi.org website.

Diagram



Fields

Field	Function
31-15	Reserved
—	
14-0	Vendor ID
VID	May be set to the 15-bit MIPI Vendor ID, if used.

52.7.24 Slave Time Control Clock Register (STCCLOCK)

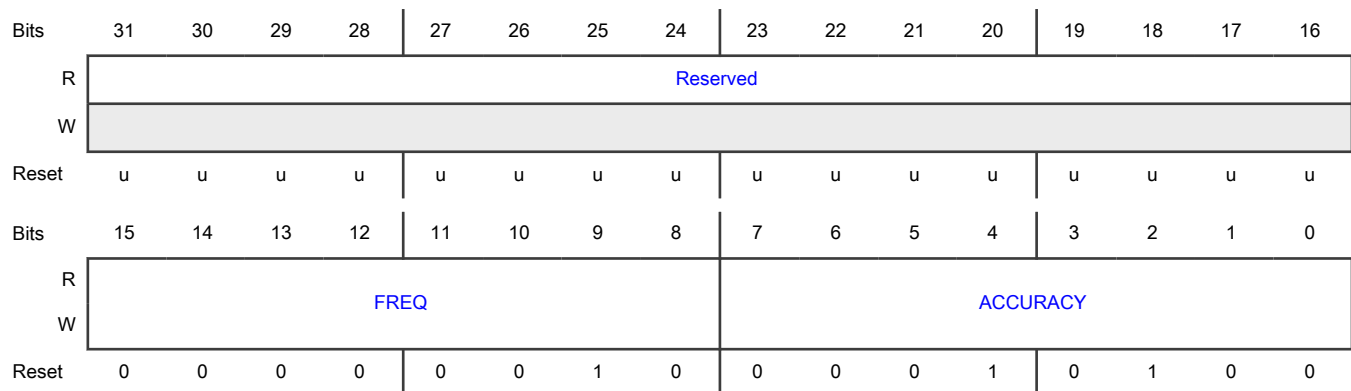
Offset

Register	Offset
STCCLOCK	78h

Function

Allows an application to dynamically set the time control clock and accuracy information. The clock frequency and accuracy is normally a constant set by the hardware. But if the clock can be adjusted (i.e., divided) or if the accuracy could vary with knowable information, then the clock may be set via this register. This register should be updated whenever the clock source is changed.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 FREQ	Clock frequency Clock frequency in 0.5-MHz steps. For example, 10 MHz is FREQ= 20. Default: set by parameters if configured.
7-0 ACCURACY	Clock accuracy Clock accuracy in 1/10ths of %. For example, 1.5% is 15. Default: set by parameters if configured.

52.7.25 Slave Message-Mapped Address Register (SMSGMAPADDR)

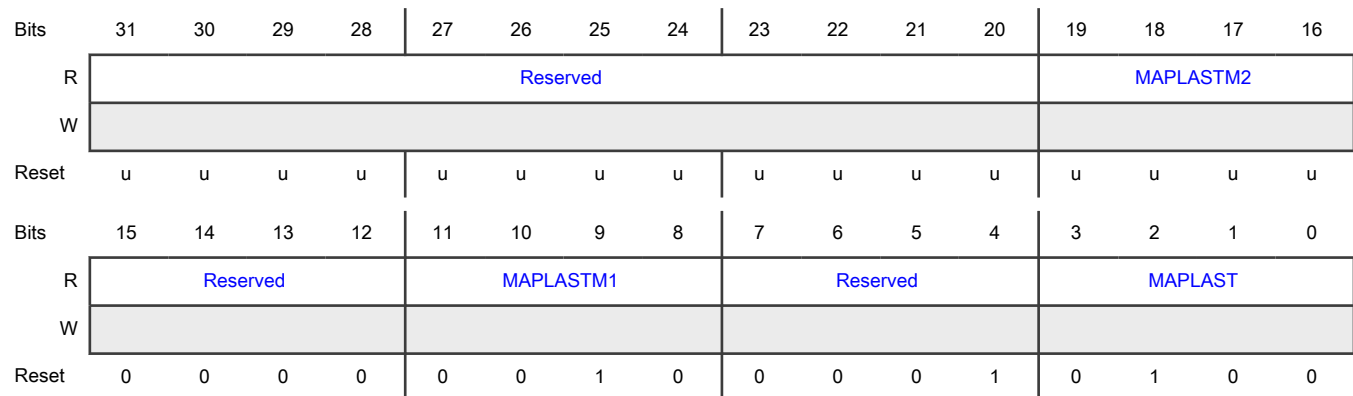
Offset

Register	Offset
SMSGMAPADDR	7Ch

Function

Allows an application to dynamically set the time control clock and accuracy information. When the SDYNADDR register is used to build a list of extra DAs or SAs to match, then the SMSGMAPADDR register enables the software to determine which address was matched when the SSTATUS.MATCHED bit is set. The SMSGMAPADDR register holds the last 3 matches (MAPLAST, MAPLASTM1, MAPLASTM2).

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 MAPLASTM2	Previous match index 2 0 for the base address.
15-12 —	Reserved
11-8 MAPLASTM1	Previous match index 1 0 for the base address.
7-4 —	Reserved
3-0 MAPLAST	Matched address index Matched address index for current or last matched message. 0 for the base address. Only valid if mapped address list is enabled.

52.7.26 Master Main Control Register (MCTRL)

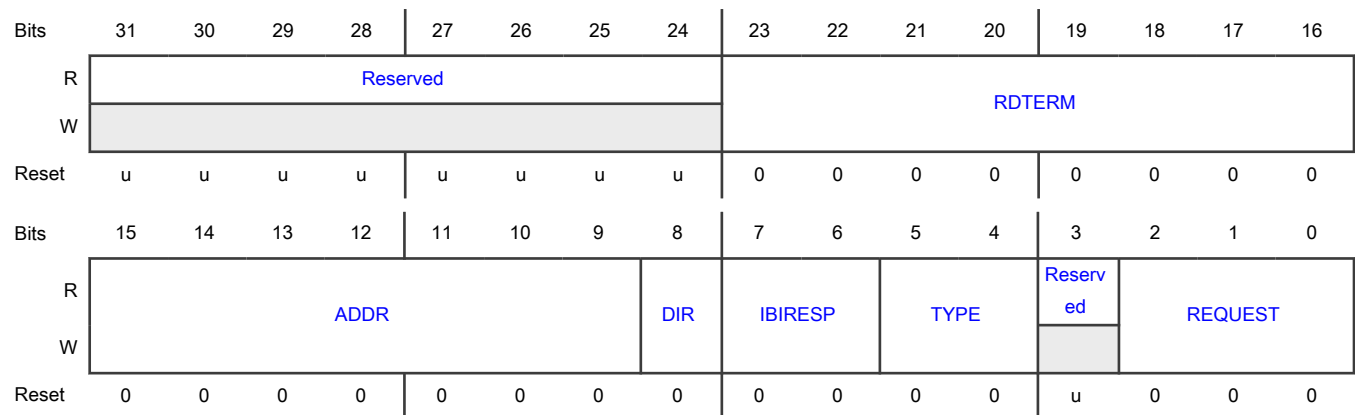
Offset

Register	Offset
MCTRL	84h

Function

Starts activities on the I3C or I2C bus.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 RDTERM	Read terminate The termination counter for reads. - For I2C, RDTERM controls when to NACK the read. - For I3C, RDTERM can be use to terminate (end) a read that is too long. - RDTERM=0 has no effect - RDTERM=1 will terminate after the next character - RDTERM=2 will terminate after the next 2 characters Supports up to 255 characters. If in Double Data Rate (DDR) mode, then RDTERM will terminate the read, based on word counts (for DDR) vs. byte counts (for SDR).
15-9 ADDR	ADDR Address with START for I3C or I2C
8	DIR

Table continues on the next page...

Table continued from the previous page...

Field	Function
DIR	0b - DIRWRITE: Write 1b - DIRREAD: Read
7-6 IBIRESP	In-Band Interrupt (IBI) response The response to use when you get an In-Band Interrupt (IBI) from START, and to force using a IbiAckNack request when completing a manual In-Band Interrupt. Completion of a manual In-Band Interrupt means that the slave Dynamic Address (DA) is known, and so the mandatory byte (or not) is specified by the application when ACKing. MIBIRESP register is also used when a message is emitted in Message Mode using the MWMSG_SDR or MWMSG_DDR registers. 00b - ACK. Acknowledge. A mandatory byte (or not) is decided by the Master In-band Interrupt Registry and Rules Register (MIBIRULES). To limit the maximum number of IBI bytes, configure the Read Termination field (MCTRL.RDTERM). 01b - NACK. Not acknowledge 10b - ACK_WITH_MANDATORY. Acknowledge with mandatory byte (ignores the MIBIRULES register). Acknowledge with mandatory byte should not be used, unless only slaves <i>with a mandatory byte</i> can cause an In-Band Interrupt. 11b - MANUAL. Stop and wait for a decision using the IBIAckNack request
5-4 TYPE	Bus type with START 00b - I3C. Normally the SDR mode of I3C. For ForceExit, the Exit pattern. 01b - I2C. Normally the Standard I2C protocol. 10b - DDR. (Double Data Rate): Normally the HDR-DDR mode of I3C. Enter DDR mode (7E and then ENTHDR0), if the module is not already in DDR mode. The 1st byte written to the TX FIFO should be a command, and should already be in the FIFO. To end DDR mode, use ForceExit. For ForceExit, the normal IBHR (In-Band Hardware Reset). 11b - For ForcedExit, this is forced IBHR.
3 —	Reserved
2-0 REQUEST	Request Emits the requested operation when doing in pieces vs. doing by message. The MSTATUS register should be checked because the state of the Master must be such that the request makes sense; for example, the system cannot do SDR from DDR (but can do DDR from SDR since it enters), cannot use an incorrect request if in DAA mode, and other similar situations. 000b - NONE. Returns to this when finished with any request. The MSTATUS register indicates the master's state. See also AutoIBI mode. NONE is only written as 0: when setting RDTERM to 1 (to stop a read in progress) or when setting IBI reponse field (IBIRESP) for MSG use. 001b - EMITSTARTADDR. Emit START with address and direction from a stopped state or in the middle of a Single Data Rate (SDR) message. If from a stopped state (IDLE), then emit start may

Table continues on the next page...

Table continued from the previous page...

Field	Function
	be prevented by an event (like IBI, MR, HJ), in which case the appropriate interrupt is signaled; note that Emit START can be resubmitted. 010b - EMITSTOP. Emit a STOP on bus. Must be in Single Data Rate (SDR) mode. If in Dynamic Address Assignment (DAA) mode, Emit stop will exit DAA mode. 011b - IBIACKNACK. Manual In-Band Interrupt (IBI) Acknowledge (ACK) or Not Acknowledge (NACK). When IBIRESP has indicated a hold on an In-Band Interrupt to allow a manual decision, this request completes it. Uses IBIRESP to provide the information. 100b - PROCESSDAA. If not in Dynamic Address Assignment (DAA) mode now, will issue START, 7E, ENTDA, and then will emit 7E/R to process first Slave. Will stop just before the new Dynamic Address (DA) is to be emitted. The DA is written using MWDATAB bits 6:0 and then ProcessDAA is requested again to write the new address, and then starts the next; an MSTATUS indicating NACK means DA was not accepted (e.g. parity error). If Process DAA is NACKed on the 7E/R, the interrupt will indicate so and this will emit a STOP. 101b - Reserved 110b - FORCEEXIT and IBHR. Emit an Exit Pattern from any state, but end Double Data Rate (DDR) (including MSGDDR), if in DDR mode now. Includes a STOP afterward. If TYPE != 0, then it will perform an IBHR (In-Band Hardware Reset). If TYPE=2, then it does a normal reset (DEFIRST can prevent the reset). If TYPE=3, it does a forced reset (will always reset). 111b - AUTOIBI. Hold in a stopped state, but auto-emit START,7E when the slave is holding down SDA to get an In-Band Interrupt (IBI). Actual In-Band Interrupt handling is defined by IBIRESP.

52.7.27 Master Status Register (MSTATUS)

Offset

Register	Offset
MSTATUS	88h

Function

Status for the master, including which events caused the interrupts. The peripherals share the IRQ (called Parallel-to-Slave status). Because a peripheral can only be in a Master or Slave mode, but not be in both modes at the same time, then only one (Slave or Master peripheral) can be the cause of the IRQ. So, if there is an IRQ and the peripheral is a Master, then this register (MSTATUS) has the status. If there is an IRQ and the peripheral is a Slave, then the SSTATUS register has the status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed	IBIADDR								Reserved				NOWM AST...	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	u	u	u	u	0	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERRW ARN	Reserv ed	IBIWON	TXNO TFU...	RXPE ND	COMPLETE	MCTRLDO...	SLVSTART	IBITYPE		NACK ED	BETWEEN	Reserv ed	STATE		
W																
Reset	0	u	0	1	0	0	0	0	0	0	0	0	u	0	0	0

Fields

Field	Function
31 —	Reserved
30-24 IBIADDR	IBI address The address of the In-Band Interrupt (while IBITYPE=1) or the Master Request (MR)(while IBITYPE=2) or is 7'h2 if Hot-Join (while IBITYPE=3).
23-20 —	Reserved
19 NOWMASTER	Now master (now this module is a master) NOWMASTER=1 if the module is now a master (it was previously a slave, acceptance was requested from the previous master and it was accepted). Note that the reverse operation (master becomes a slave) does not need an interrupt, because the application is granting it via the Common-Command-Code status bit MSTATUS.CCC.
18-16 —	Reserved
15 ERRWARN	Error or warning ERRWARN=1, if an error occurred (like improper register use, overrun or underrun of FIFO/buffer, invalid parity or CRC in DDR read, and others). See the Master Errors and Warnings Register (MERRWARN).
14 —	Reserved
13 IBIWON	In-Band Interrupt (IBI) won IBIWON is set to 1 if IBI or MR or HJ won the arbitration on a header address, regardless of whether it was NACKed or ACKed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 TXNOTFULL	TX buffer/FIFO not yet full TXNOTFULL=1 if TX buffer/FIFO or message register can accept another byte or half-word.
11 RXPEND	RXPEND The receiving message from a slave and byte(s) are in the input buffer/FIFO. • If using a FIFO, this is one FIFO trigger's worth. • If DMA is enabled for RX, then DMA will be signaled. RXPEND will self-clear when the data is read.
10 COMPLETE	COMPLETE A message has completed: • With MWMSG_SDR or MWMSG_DDR, this is count to 0. • With MCTRL using EmitStartAddr, this is the end of write or read by terminate or end. • With IBI (AutoIBI or EmitStartAddr), this is the end of IBI data (if any).
9 MCTRLDONE	Master control done After the module completes an MCTRL request, MCTRLDONE is set to 1. MCTRLDONE self-clears when writing a new control; otherwise write to MCTRLDONE to clear it (MCTRLDONE). • When MCTRL.REQUEST=EmitStartAddr, MCTRLDONE fires (sets to 1) when the address went out (and was ACKed or NACKed or ended in an IBI). However if ACKed, it will then fire COMPLETE when the write or read data has completed (finished). • When MCTRL.REQUEST=ProcessDAA, this fires when the module is ready to emit the Dynamic Address (DA) for the slave, or when no more slaves are ACKing. This can be determined using the MSTATUS.BETWEEN and MSTATUS.STATE fields.
8 SLVSTART	Slave start If a slave is/was requesting a START by holding SDA low, SLVSTART is set to 1. Handling will start automatically if MCTRL.REQUEST=AutoIBI.
7-6 IBITYPE	In-Band Interrupt (IBI) type Indicates the type of In-Band Interrupt (IBI) of the last event that won the arbitration (whether the interrupt is ACKed or NACKed or pending) 00b - NONE. cleared when IBI Won bit (MSTATUS.IBIWON) is cleared 01b - IBI. In-Band Interrupt 10b - MR. Master Request 11b - HJ. Hot-Join
5 NACKED	Not acknowledged If NACKED=1, then the last Start and Address was NACKed (was not ACKed by the addressed slave).
4	Between

Table continues on the next page...

Table continued from the previous page...

Field	Function
BETWEEN	Between messages or Dynamic Address Assignments (DAA) Active if: - STATE is MSGSDR, DDR, or DAA, the state is between messages/DAA's and so is expecting a new messages/DAA's to start (or STOP or Exit). - STATE is NORMACT, the module is stuck waiting on the TX FIFO to be not-empty or the RX FIFO to be not full. 0b - Inactive. For other cases. 1b - Active. If STATE is MSGSDR, DDR, or DAA; or if STATE is NORMACT.
3 —	Reserved
2-0 STATE	State of the master Indicates the current master state 000b - IDLE. The bus has STOPped. 001b - SLVREQ. (Slave Request state) The bus has STOPped but a slave is holding SDA low. If using auto-emit IBI (MCTRL.AutoIBI), then the master will not stay in the Slave Request state. 010b - MSGSDR. In Single Data Rate (SDR) Message state (from using MWMSG_SDR) 011b - NORMACT. Normal active Single Data Rate (SDR) state (from using MCTRL and MWDATAN and MRDATAN registers). The master will stay in the NORMACT state until a STOP is issued. 100b - MSGDDR. In Double Data Rate (DDR) Message mode (from using MWMSG_DDR or using the normal method with DDR). The master will stay in the DDR state, until the master exits using EXIT (emits the Exit pattern). 101b - DAA. In Enter Dynamic Address Assignment (ENTDAA) mode 110b - IBIACK. Waiting for an In-Band Interrupt (IBI) ACK/NACK decision 111b - IBIRCV. Receiving an In-Band Interrupt (IBI); this IBIRCV state is used after IBI/MR/HJ has won the arbitration, and IBIRCV state is also used for IBI mandatory byte (if any) and any bytes that follow.

52.7.28 Master In-band Interrupt Registry and Rules Register (MIBIRULES)

Offset

Register	Offset
MIBIRULES	8Ch

Function

Contains the rules for using In-band Interrupts, and keeps a registry of the slaves who use the IBI byte.

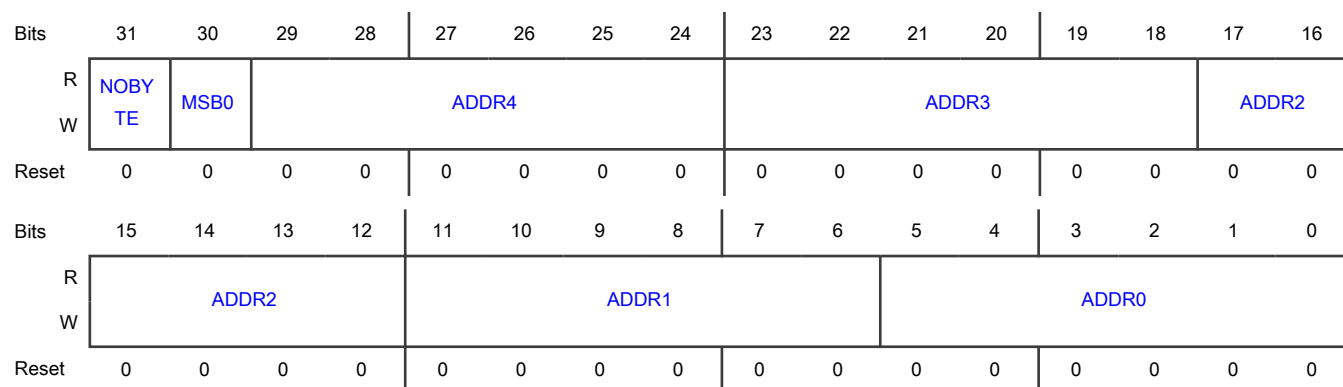
Concerning ADDRn fields: The address is 6 bits, and assuming that IBIRULES.MSB0=1, then each address has its most significant bit = 0. This means is that each address is 7 bits (usually written as A7 to A1), and the A7 must be 0. If the application does not use that optimal convention, then the Manual method of IBI ACK handling must be used (see MCTRL).

- The default is that the ADDRn values indicate the slaves with a mandatory byte.
- If MIBIRULES.NOBYTE is set, then the ADDRn values indicate slaves that do not have a mandatory IBI byte.

NOTE

A7=0 is only needed for Slaves that use IBI. For legacy I2C devices, A7 can use any valid value, because I2C devices cannot cause an IBI.

Diagram



Fields

Field	Function
31 NOBYTE	No IBI byte • If NOBYTE=1, then the ADDR_x fields refer to slaves without a mandatory IBI byte. • If NOBYTE=0, then the ADDR_x fields refer to slaves with a mandatory IBI byte.
30 MSB0	Set Most Significant address Bit to 0 If MSB0=1, then all I3C dynamic addresses will be assigned with the MSb==0 (most significant bit); this allows the START header to be optimized.
29-24 ADDR4	ADDR4 Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.
23-18 ADDR3	ADDR3 Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.
17-12 ADDR2	ADDR2 Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.
11-6 ADDR1	ADDR1 Address of slave with or without Mandatory IBI byte. See ADDR0 field description for more details.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-0 ADDR0	ADDR0 Address of slave with or without Mandatory IBI byte. If 0, then the address does not apply.

52.7.29 Master Interrupt Set Register (MINTSET)

Offset

Register	Offset
MINTSET	90h

Function

Set interrupt enables for select bits in MSTATUS. Reading the MINTSET register returns the status of the interrupt enables.

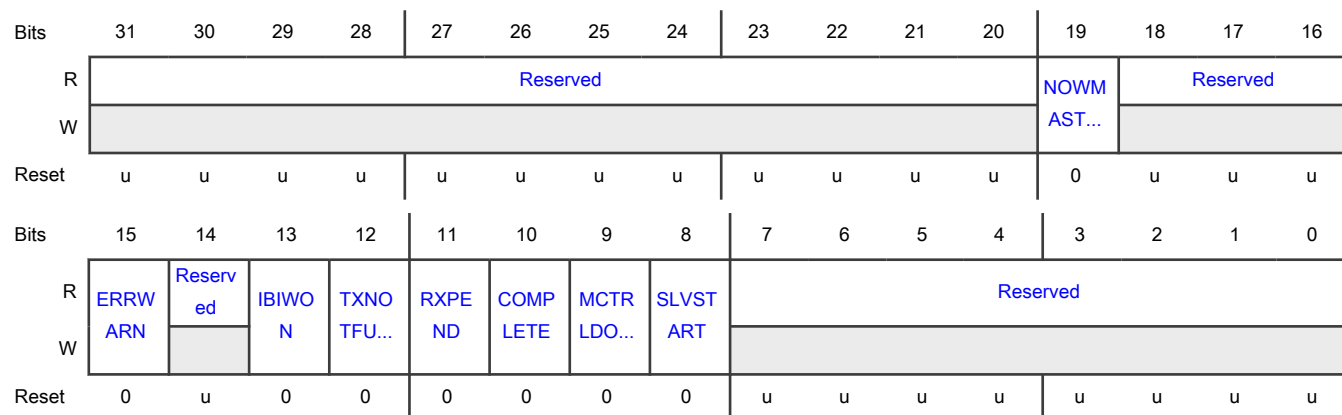
- To activate an interrupt enable, write 1 to it.
- To disable an interrupt enable, write 1 to the appropriate bit in the Interrupt Clear Register (MINTCLR). Writing 0 to the interrupt enable in this register (MINTSET) does not disable the interrupt.

The Interrupt registers allow masking interrupt sources, as well as checking which interrupts have activated in the MSTATUS register. The normal method is to enable an interrupt and then once that interrupt fires, the interrupt is either cleared (by writing the MSTATUS register) or cleared by action (on the corresponding data register). The interrupt is level held, meaning that the interrupt stays set until the cause is cleared, by some method. The module prevents races, so that if a new event comes in, that new event will not be lost.

These interrupts are parallel to the Slave INTSET/INTCLR set, and only one interrupt set is active depending on state (Master or Slave operation).

- MINTSET: sets interrupt enables for MSTATUS bits. Reading MINTSET register returns the status of the interrupt enables.
- MINTCLR: clears interrupt enables for MSTATUS bits.
- MINTMASKED: returns the value of the MSTATUS bits ANDed with their interrupt enables.

Diagram



Fields

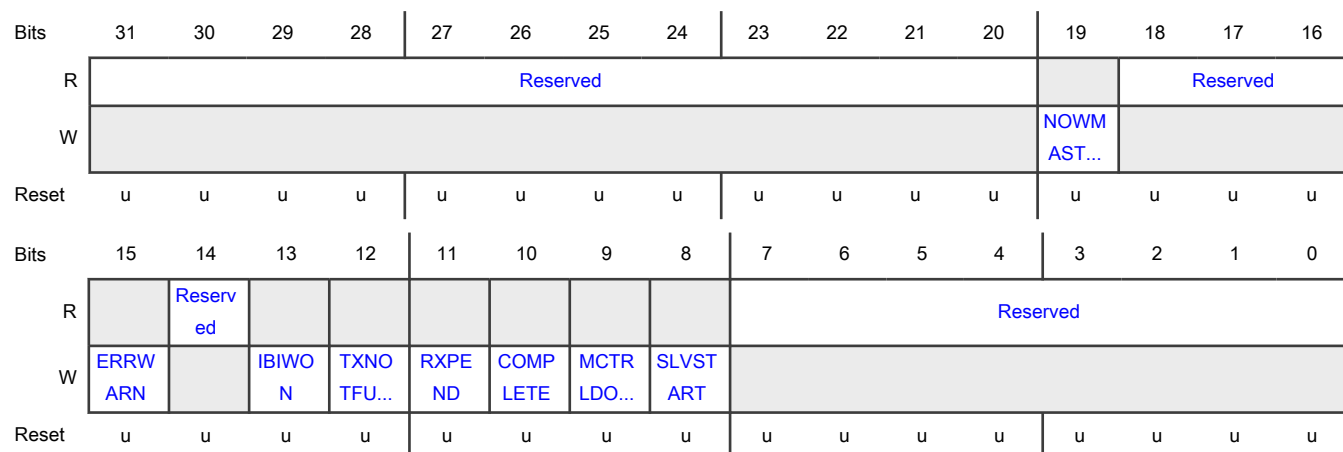
Field	Function
31-20 —	Reserved
19 NOWMASTER	Now master (now this I3C module is a master) interrupt enable
18-16 —	Reserved
15 ERRWARN	Error or warning (ERRWARN) interrupt enable
14 —	Reserved
13 IBIWON	In-Band Interrupt (IBI) won interrupt enable
12 TXNOTFULL	TX buffer/FIFO is not full interrupt enable
11 RXPEND	RX pending interrupt enable
10 COMPLETE	Completed message interrupt enable
9 MCTRLDONE	Master control done interrupt enable
8 SLVSTART	Slave start interrupt enable
7-0 —	Reserved

52.7.30 Master Interrupt Clear Register (MINTCLR)**Offset**

Register	Offset
MINTCLR	94h

Function

Clear interrupt enables for select MSTATUS bits. Writing a 1 clears the corresponding interrupt enable; writing a 0 has no effect.

Diagram**Fields**

Field	Function
31-20 —	Reserved
19 NOWMASTER	NOWMASTER interrupt enable clear
18-16 —	Reserved
15 ERRWARN	ERRWARN interrupt enable clear
14 —	Reserved
13 IBIWON	IBIWON interrupt enable clear
12 TXNOTFULL	TXNOTFULL interrupt enable clear
11 RXPEND	RXPEND interrupt enable clear
10	COMPLETE interrupt enable clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
COMPLETE	
9 MCTRLDONE	MCTRLDONE interrupt enable clear
8 SLVSTART	SLVSTART interrupt enable clear
7-0 —	Reserved

52.7.31 Master Interrupt Mask Register (MINTMASKED)

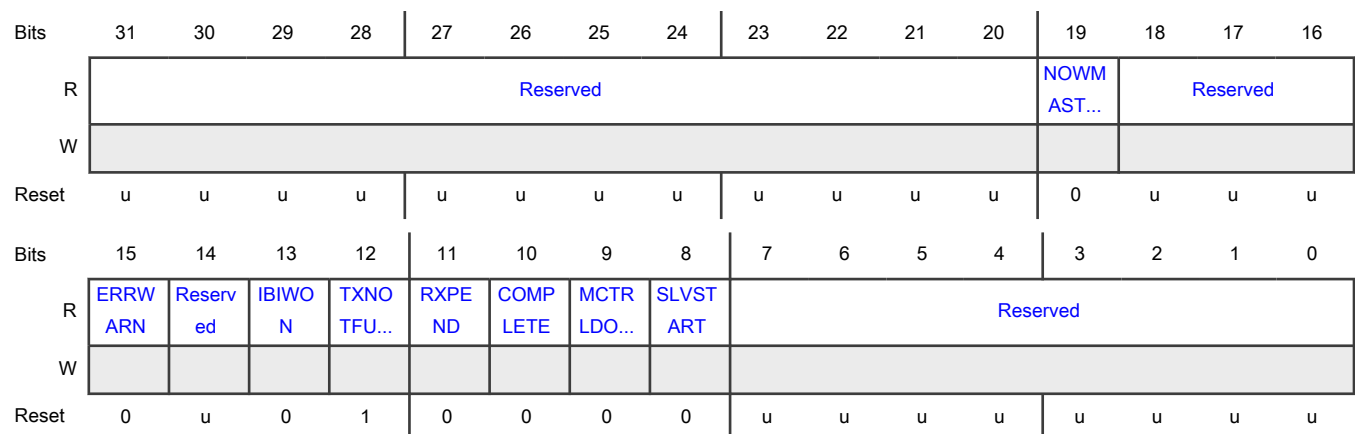
Offset

Register	Offset
MINTMASKED	98h

Function

Returns the status of enabled interrupts (the value of MSTATUS ANDed with MINTSET).

Diagram



Fields

Field	Function
31-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 NOWMASTER	NOWMASTER interrupt mask
18-16 —	Reserved
15 ERRWARN	ERRWARN interrupt mask
14 —	Reserved
13 IBIWON	IBIWON interrupt mask
12 TXNOTFULL	TXNOTFULL interrupt mask
11 RXPEND	RXPEND interrupt mask
10 COMPLETE	COMPLETE interrupt mask
9 MCTRLDONE	MCTRLDONE interrupt mask
8 SLVSTART	SLVSTART interrupt mask
7-0 —	Reserved

52.7.32 Master Errors and Warnings Register (MERRWARN)

Offset

Register	Offset
MERRWARN	9Ch

Function

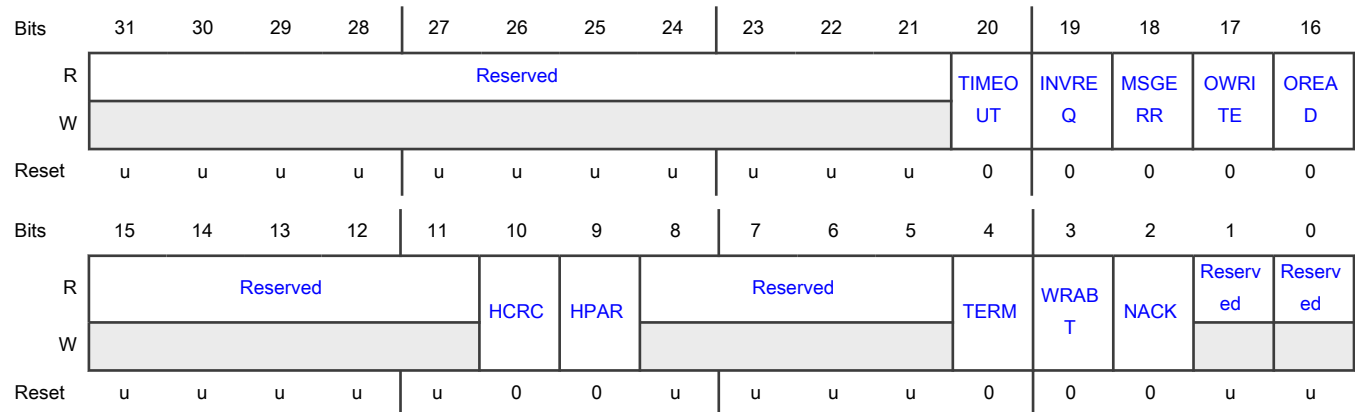
Contains errors and warnings from I3C/I2C protocol. When any errors or warnings are not 0, then the MSTATUS.ERRWARN bit will be set.

Parallel-to-slave ERRWARN:

- In Master mode, use this register (MERRWARN).
- In Slave mode, use the slave register SERRWARN.

The error bits in both registers (MERRWARN and SERRWARN) are similar in meaning.

Diagram



Fields

Field	Function
31-21 —	Reserved
20 TIMEOUT	TIMEOUT error The module has stalled too long in a frame. This happens: • When the TX FIFO or RX FIFO is not handled and the bus is stuck in the middle of a message, • When no STOP was issued and between messages, • When IBI manual is used and no decision was made. The maximum stall period is 10 KHz or 100 us.
19 INVREQ	Invalid request error Invalid use of a request: • Not using IBIAckNack when stopped in manual hold for IBI acknowledge. • Using other than ForceStop or ForceExit while in a message. Others are OK when the message is done. • Other mismatched uses (for example, IBIAckNack when in normal states).
18 MSGERR	Message error • Trying to write to or read from MWMSG_SDR register when in a DDR message (double data rate). • Trying to write to or read from MWMSG_DDR register when in a SDR message (single data rate). • Trying to read from HRMSG_SDR or HRMSG_DDR registers when no message has yet started.

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 OWRITE	Over-write error Trying to write to the Master Write Data Byte Register (MWDATAB) when the FIFO is full.
16 OREAD	Over-read error Trying to read from the Master Read Data Byte Register (MRDATAB) when the FIFO is empty.
15-11 —	Reserved
10 HCRC	High data rate CRC error A Cyclic Redundancy Check (CRC) error occurred from a Double Data Rate (DDR) read.
9 HPAR	High data rate parity - Parity error from a Double Data Rate (DDR) read; this includes a bad preamble on a read. - Does not stop the read, because it is not safe to terminate (because the read data may get mis-framed). - Will end on a run of 1 second.
8-5 —	Reserved
4 TERM	Terminate error - This master terminated a slave read because the read exceeded the count for the message; only valid when using the MWMSG_SDR or MWMSG_DDR register. - TERM self-clears if the MWMSG_SDR or MWMSG_DDR register is written.
3 WRABT	WRABT (Write abort) error - The I2C slave NACKed the write data, terminating the message. - WRABT self-clears if the MCTRL register is written. For example, the Master is writing in I2C and the Slave NACKed the write.
2 NACK	Not acknowledge (NACK) error - The slave or slaves NACKed (not acknowledged) the last address. - If 7E was the address, then all slaves NACKed the last address. - NACK self-clears if the MCTRL register is written.
1 —	Reserved
0 —	Reserved

52.7.33 Master DMA Control Register (MDMACTRL)

Offset

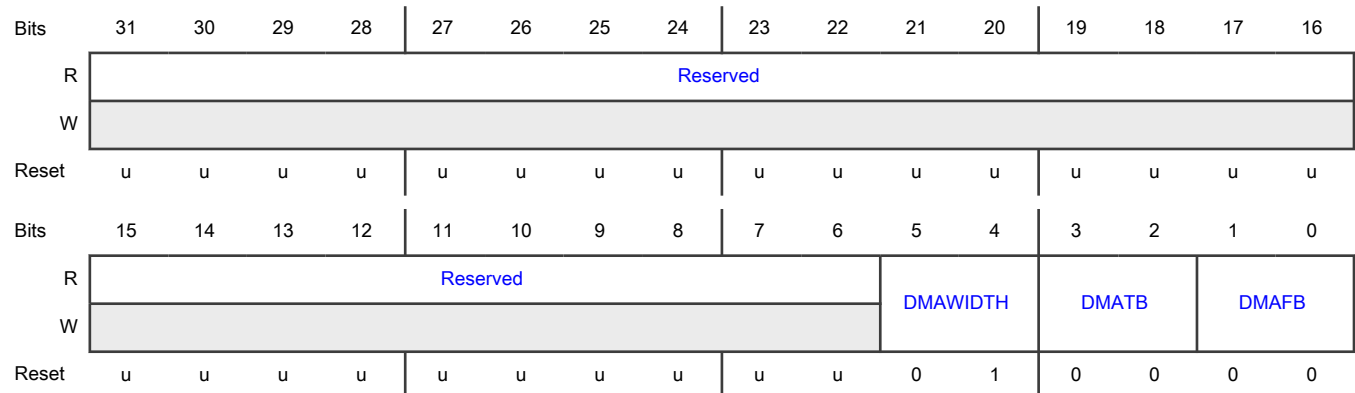
Register	Offset
MDMACTRL	A0h

Function

The DMA Control register allows for DMA to be used for inbound and outbound messages. DMA is much more useful for a Master than for a Slave, because the Master is directing the bus traffic and actions, so setting up DMA is more natural for a Master. DMA can be used with a Master in one of two ways:

- Push or Pull data to go with an MCTRL.REQUEST=EMITSTARTADDR request written by the processor.
- Implementing a message mode, to be completely DMA-controlled.

Diagram



Fields

Field	Function
31-6 —	Reserved
5-4 DMAWIDTH	DMA width Specifies the data width of DMA operations. 00b - BYTE 01b - BYTE_AGAIN 10b - HALF_WORD. Half-word (16 bits). This will make sure that 2 bytes are free/available in FIFO. 11b - Reserved
3-2 DMATB	DMA to bus

Table continues on the next page...

Table continued from the previous page...

Field	Function
	DMA Write (to-bus) trigger. If enabled with DMATB=1 or 2, then the I3C module will start request DMA on TX trigger. See the Master Data Control Register (MDATACTRL). The I3C module will request until full, unless DMA is set up as a trigger. DMAFB will cancel on MSTATUS.ERRWARN. 00b - NOT_USED. DMA is not used 01b - ENABLE_ONE_FRAME. DMA is enabled for 1 frame (ended by DMA or Terminated). DMATB auto-clears on STOP or START. See MCONFIG.MATCHSS. 10b - ENABLE. DMA is enabled until DMA is turned off. Normally DMA ENABLE should only be used in Master Message mode.
1-0 DMAFB	DMA from bus DMA Read (from-bus) trigger. If enabled with DMAFB=1 or 2, then the I3C module will request DMA on a RX trigger (see MDATACTRL register). The I3C module will request until empty, unless the DMA is set up as a trigger. DMAFB will cancel on MSTATUS.ERRWARN. 00b - NOT_USED. DMA is not used 01b - ENABLE_ONE_FRAME. DMA is enabled for 1 frame. DMAFB auto-clears on STOP or repeated START. See MCONFIG.MATCHSS. 10b - ENABLE. DMA is enabled until the DMA is turned off.

52.7.34 Master Data Control Register (MDATACTRL)

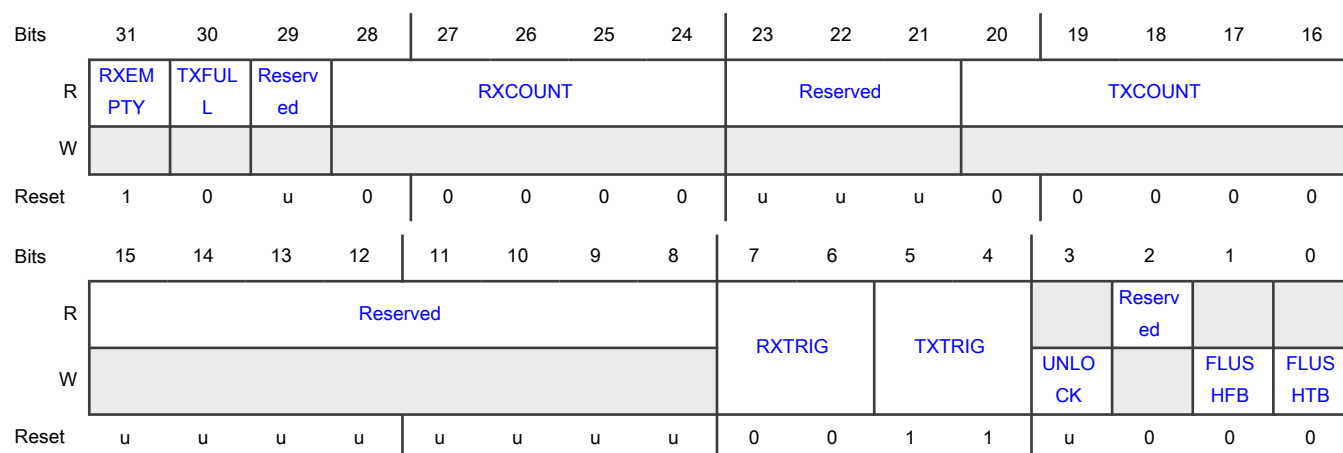
Offset

Register	Offset
MDATACTRL	ACh

Function

Controls data buffering and indicates the current buffer state. The MDATACTRL register is the alias of slave DATACTRL register (SDATACTRL). The MDATACTRL register is also the FIFO control.

Diagram



Fields

Field	Function
31 RXEMPTY	RX is empty • RXEMPTY=1 if RX is empty • RXEMPTY=0 if RX is not yet empty
30 TXFULL	TX is full • TXFULL=1 if TX is full • TXFULL=0 if TX is not yet full
29 —	Reserved
28-24 RXCOUNT	RX byte count The count of bytes in RX
23-21 —	Reserved
20-16 TXCOUNT	TX byte count The count of bytes waiting in the TXFIFO. This is how many bytes the application has written to the TXFIFO, that have not yet gone onto the I3C bus.
15-8 —	Reserved
7-6 RXTRIG	RX trigger level Trigger level for RX fullness when using a FIFO. Affects interrupts and DMA (if enabled). The default is off, and it triggers on any byte in the RXFIFO (meaning that the RXFIFO is not empty).

Table continues on the next page...

Table continued from the previous page...

Field	Function
5-4 TXTRIG	TX trigger level Trigger level for TX emptiness when using a FIFO. Affects interrupts and DMA (if enabled). The default is 3.
3 UNLOCK	Unlock - If 0 is written to UNLOCK, then the RXTRIG and TXTRIG fields cannot be changed on a write. - If 1 is written to UNLOCK, then the RXTRIG and TXTRIG fields can be changed on a write.
2 —	Reserved
1 FLUSHFB	Flush from-bus buffer/FIFO Flushes the from-bus buffer/FIFO. FLUSHFB is not normally used.
0 FLUSHTB	Flush to-bus buffer/FIFO Flush the to-bus buffer/FIFO. Used when the master terminates a to-bus message (read) prematurely.

52.7.35 Master Write Data Byte Register (MWDTAB)

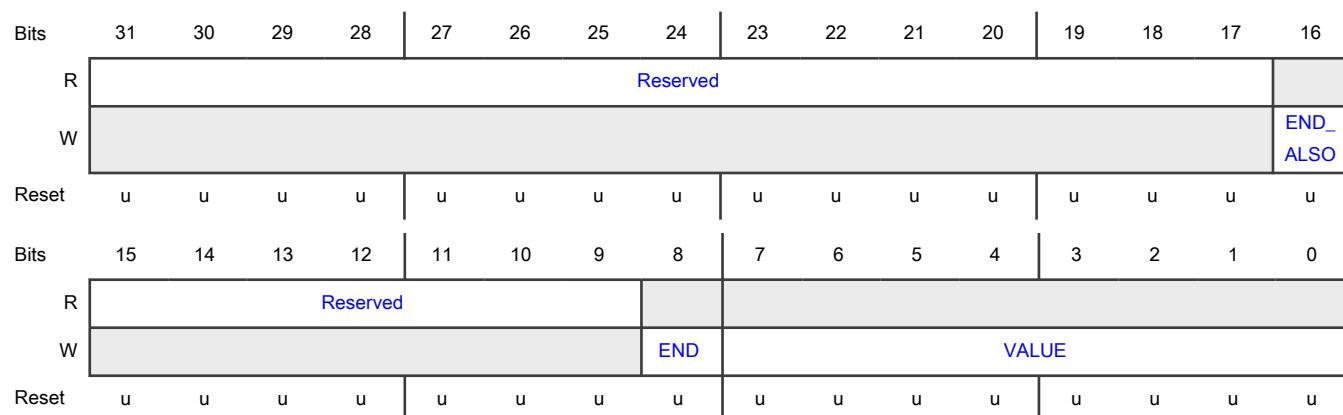
Offset

Register	Offset
MWDTAB	B0h

Function

Write a byte of data, possibly the last byte. The MWDTAB register is the alias of the WDTAB register. The MWDTAB register allows writing bytes to send onto the bus. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Diagram



Fields

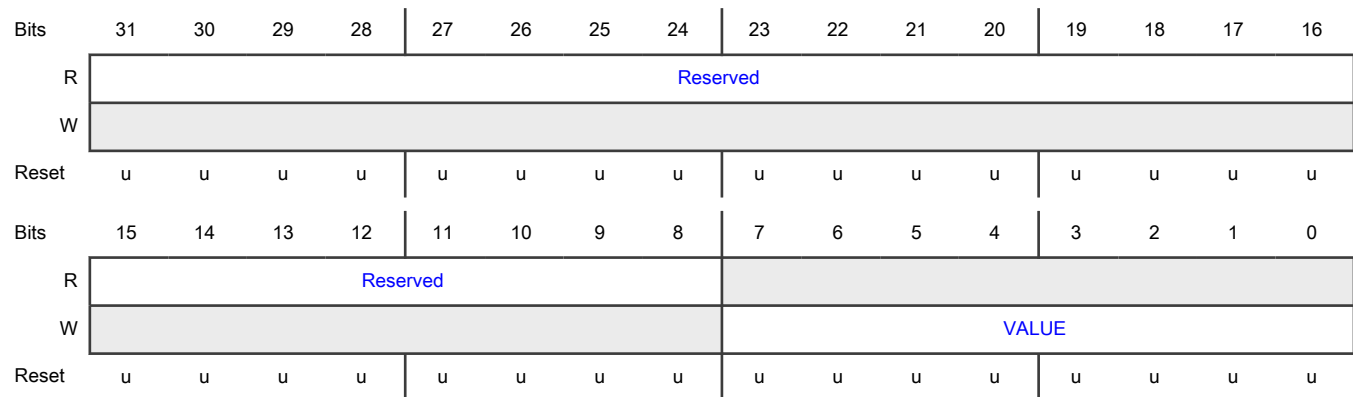
Field	Function
31-17 —	Reserved
16 END_ALSO	End of message also End outbound message normally. Any message has to end, this just indicates that it is the last message to go out. This method can be used, and also the MDATE register. • If END_ALSO=1, this marks the last byte of the message. • If END_ALSO=0, it is assumed there are more bytes. This is required to be used in I3C and is optional in I2C. For HDR-DDR (High Data Rate, Double Data Rate), the byte with the END_ALSO must be an even byte (2nd, 4th, 6th, etc) because DDR uses byte-pairs.
15-9 —	Reserved
8 END	End of message • If END=1, then the END bit marks the last byte of the message. • If END=0, then it is assumed there are more bytes. This is required to be used in I3C and is optional in I2C. For HDR-DDR (High Data Rate, Double Data Rate), the byte with the END must be an even byte (2nd, 4th, 6th, etc) because DDR uses byte-pairs.
7-0 VALUE	Data byte The byte to send to (write to) the master. • If I3C, the block will compute the parity. • If I2C, the block will handle the ACK/NACK.

52.7.36 Master Write Data Byte End Register (MWDATE)**Offset**

Register	Offset
MWDATE	B4h

Function

The MWDATE register is the alias of the Slave WDATE register (SWDATE). The MWDATE register allows writing the last byte to send onto the bus. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. Note that MWDATE can also indicate END using bit 8.

Diagram**Fields**

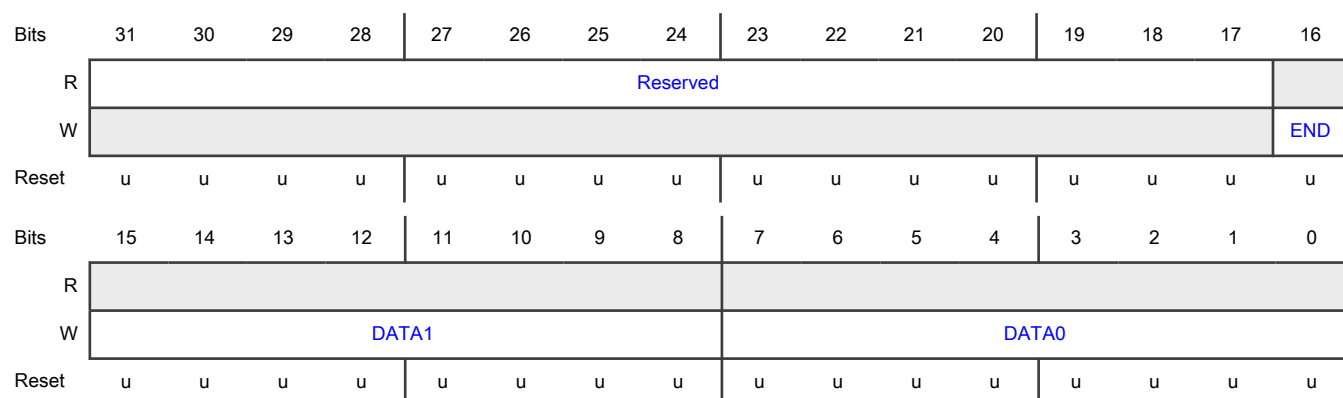
Field	Function
31-8 —	Reserved
7-0 VALUE	Data The last byte to send to (write to) the master. • If I3C, the block will compute the parity. • If I2C, the block will handle the ACK/NACK.

52.7.37 Master Write Data Half-word Register (MWDATAH)**Offset**

Register	Offset
MWDATAH	B8h

Function

Write a half-word of data, possibly the last half-word. The MWDATAH register is the alias of the WDATAH register. The Master Write Data Half-word register allows writing a half-word (pair of bytes) to the bus, unless an external FIFO is used. This takes a half-word, which will send out the Low byte and then the High byte. An end-of-data (last) marker bit is allowed (or must be 0). A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the MDATACTRL register.

Diagram**Fields**

Field	Function
31-17 —	Reserved
16 END	End of message - If END=1, then END bit marks the last byte of the message. - If END=0, then there are more bytes in the message. For this register, this always marks DATA1 as the end. END is required to be used in I3C, but is optional in I2C. For HDR-DDR, the byte with the END must be an even byte (2nd, 4th, 6th, etc.) because DDR uses byte-pairs.
15-8 DATA1	Data byte 1 The 2nd byte to send to the master.
7-0 DATA0	Data byte 0 The 1st byte to send to the master.

52.7.38 Master Write Data Byte End Register (MWDATAHE)**Offset**

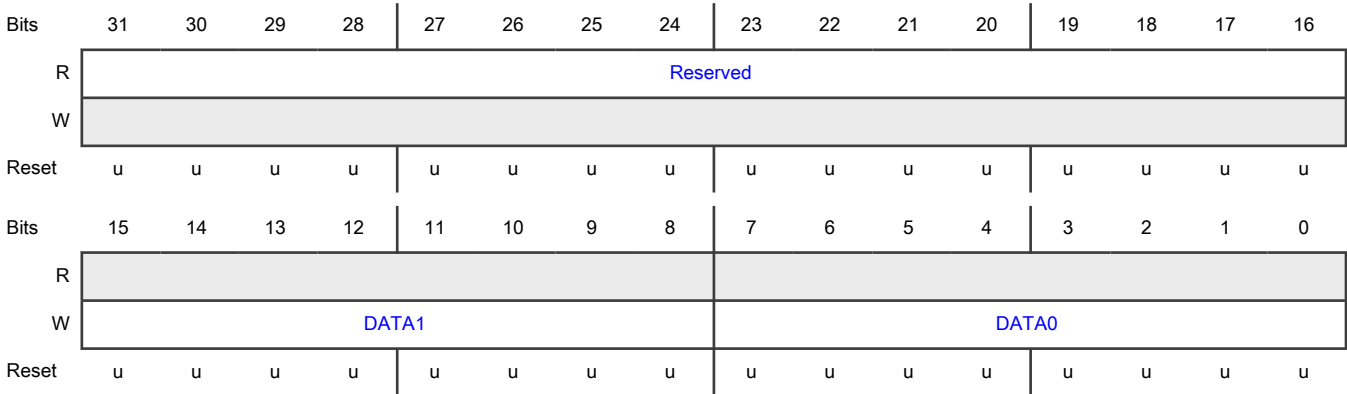
Register	Offset
MWDATAHE	BCh

Function

Write the last half-word of data. The MWDATAHE register is the alias of the Slave WDATAHE register (SWDATAHE). Write a half-word (byte pair) just like the MWDATAH register, but mark the 2nd byte as end-of-data (last byte). Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) because DDR uses byte-pairs. A half-word should not

be written unless there is room for both half-words, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the MDATACTRL register.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-8 DATA1	DATA 1 The 2nd byte to send to the master.
7-0 DATA0	DATA 0 The 1st byte to send to the master.

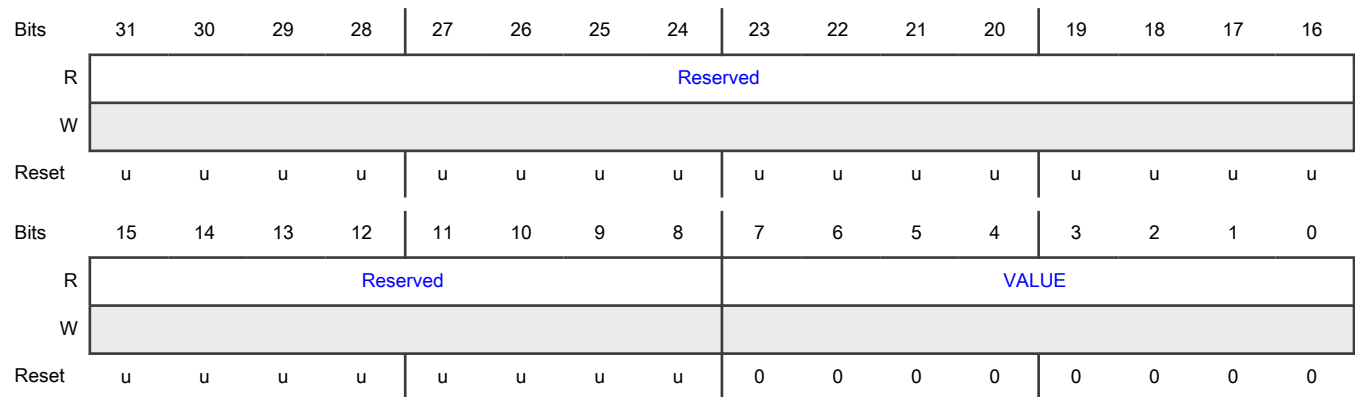
52.7.39 Master Read Data Byte Register (MRDATAB)

Offset

Register	Offset
MRDATAB	C0h

Function

Read a byte of data. The MRDATAB register is the alias of the Slave RDATAB register (SRDATAB). The MRDATAB register is used to read bytes written by the Slave on an SDR Read or DAA or DDR. This is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Diagram**Fields**

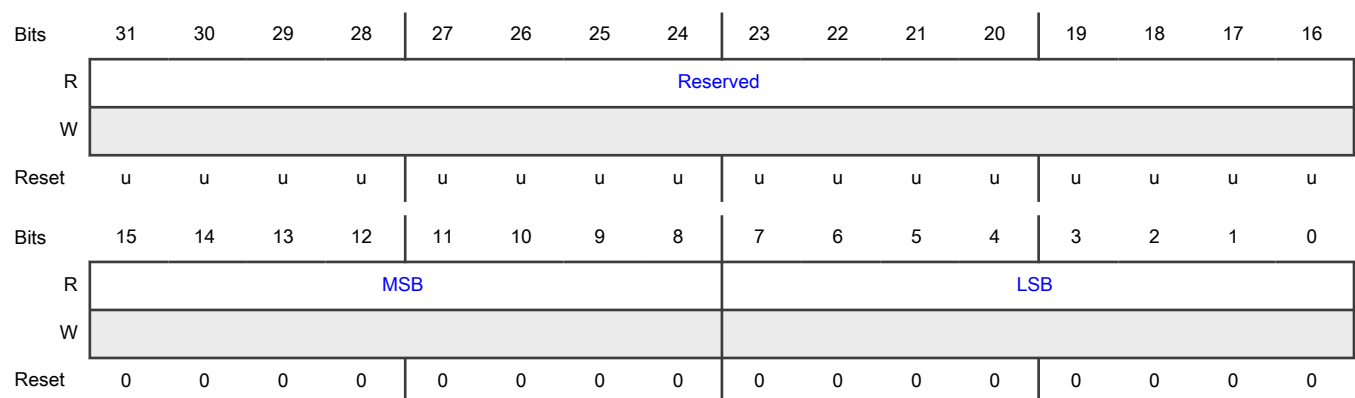
Field	Function
31-8 —	Reserved
7-0 VALUE	VALUE The byte read from the master (and written by the Slave).

52.7.40 Master Read Data Half-word Register (MRDATAH)**Offset**

Register	Offset
MRDATAH	C8h

Function

Read a half-word of data. The MRDATAH register is the alias of the RDATAH register.

Diagram

Fields

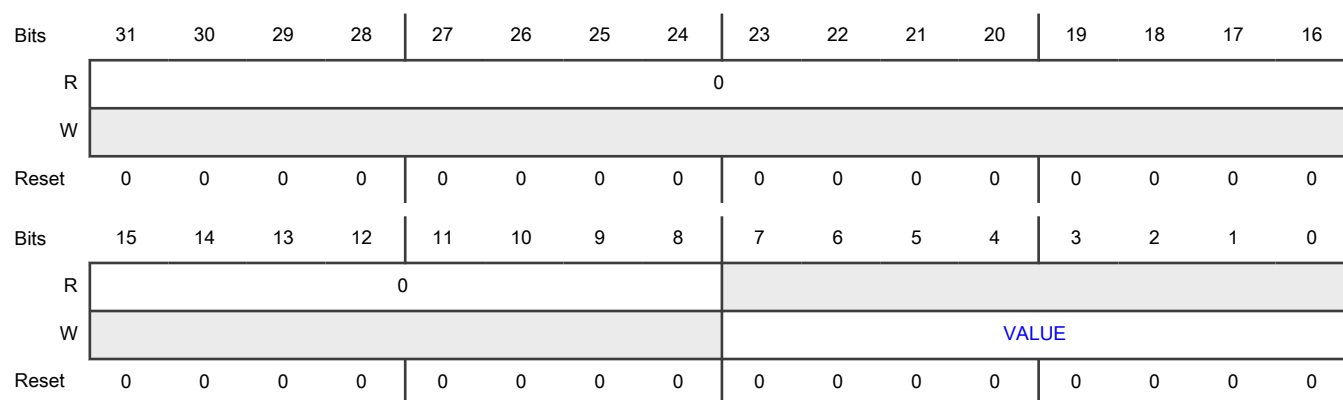
Field	Function
31-16 —	Reserved
15-8 MSB	MSB The 2nd byte read from the master.
7-0 LSB	LSB The 1st byte read from the master.

52.7.41 Write Byte Data 1 (to bus) (MWDTAB1)**Offset**

Register	Offset
MWDTAB1	CCh

Function

The MWDTAB1 register allows writing bytes to send onto the bus, and is intended for DMAs, which do not clear the upper bits of the word – MWDTAB1 does not have the END bits. MWDTAB1 is only used when using MCTRL to start the message. If MWMSG_SDR or MWMSG_DDR starts a message, that interface must be used exclusively.

Diagram**Fields**

Field	Function
31-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0	Value
VALUE	Byte to write out. The block computes the parity if I3C, or handles the ACK/NACK if I2C.

52.7.42 Master Write Message in SDR mode (MWMSG_SDR_CONTROL)

Offset

Register	Offset
MWMSG_SDR_CONTROL	D0h

Function

Set up and write 16-bit words in Single Data Rate (SDR) mode. The MWMSG_SDR_ register is modal, and has 2 modes: control and data.

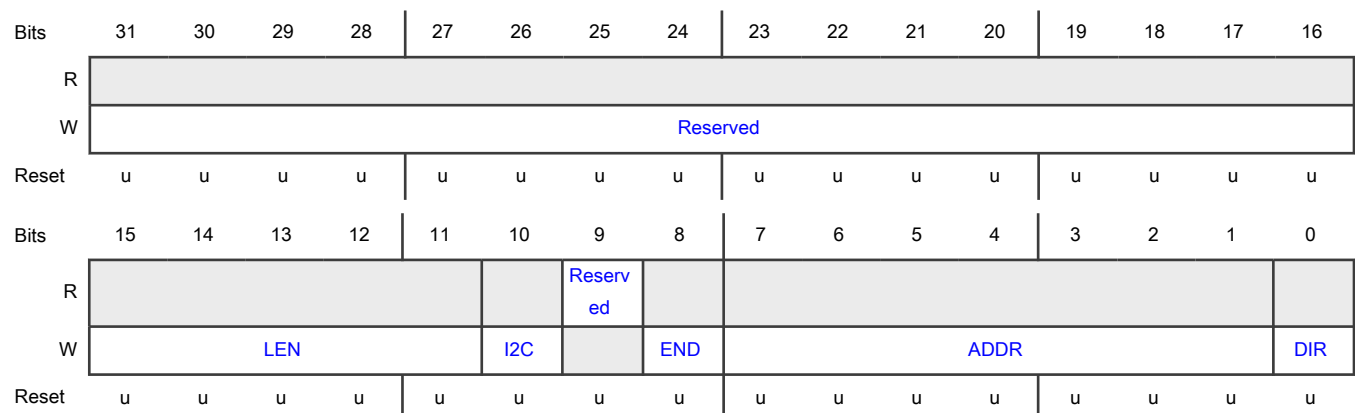
- On the first write to set up a new message, this register functions as the MWMSG_SDR_CONTROL register.
- On subsequent writes, this register functions as the [MWMSG_SDR_DATA register](#).

When not in the middle of a message, the MWMSG_SDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_SDR_DATA register is used until the Length (see LEN field) counts down, or until data with END = 1 is used.

The MWMSG_SDR_CONTROL register is written with the 16-bit Control word if currently stopped or past the end of the previous message. If MSTATUS.STATE is MSGSDR and MSTAT.BETWEEN is 0, then the register (at this offset address) is functioning as the MWMSG_SDR_DATA register; otherwise, this register (at this offset address) is functioning as the MWMSG_SDR_CONTROL register, as long as MSTATUS.STATE is not in another mode. The control word contains the byte length (LEN), the address (ADDR), the direction (DIR), and how it ends (END). Note that if START and if an event (IBI, MR, HJ) occurs, the MCTRL register's IBIRESP field will be used to determine action, and the corresponding interrupt will occur. The message will be restarted in that case.

The MWMSG_SDR_CONTROL and MWMSG_SDR_DATA registers are oriented to DMA operations; but the MWMSG_SDR_CONTROL register can also be written by the processor if desired (instead of using MCTRL and MxDATAB).

Diagram



Fields

Field	Function
31-16 —	Reserved
15-11 LEN	Length The byte length of the message. If LEN=0, then only END is used (and it will not use the address if STOPped). LEN=1 should not be used; the minimal LEN size is 2 bytes (LEN=2).
10 I2C	I2C Specifies if the message is I2C or I3C. 0b - I3C message 1b - I2C message
9 —	Reserved
8 END	End of SDR message • If END=1, then the single data rate message ends on STOP. • If END=0, then the single data rate message ends waiting for a new SDR message (will issue a repeated START for a new message). Sets MSTATUS.COMPLETE when done. This can happen before LEN bytes are read in I3C if a slave ends sooner; or this can happen before LEN bytes are written in I2C if a slave NACKs.
7-1 ADDR	Address to be written to
0 DIR	Direction 0b - Write 1b - Read

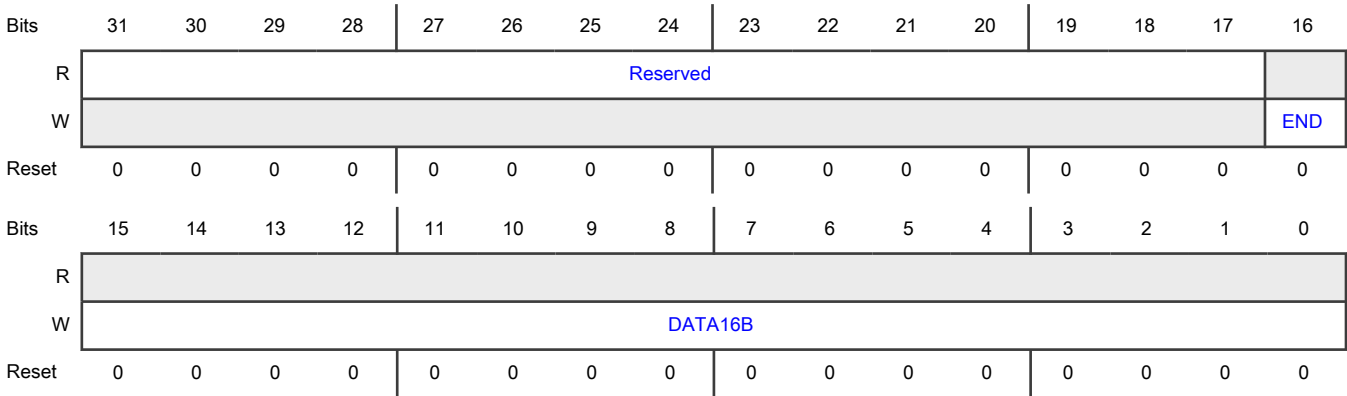
52.7.43 Master Write Message Data in SDR mode (MWMSG_SDR_DATA)**Offset**

Register	Offset
MWMSG_SDR_DATA	D0h

Function

This is the 16-bit word to be written in Single Data Rate (SDR) mode. This register also functions as the [MWMSG_SDR_CONTROL register](#).

Diagram



Fields

Field	Function
31-17 —	Reserved
16 END	End of message - If END=1, then END bit marks the last byte of the message. - If END=0, then the normal message count is used. The default is to use END=0 and let the header count count down normally. END allows terminating a write earlier than MWMSG_SDR_CONTROL.LEN field does.
15-0 DATA16B	Data

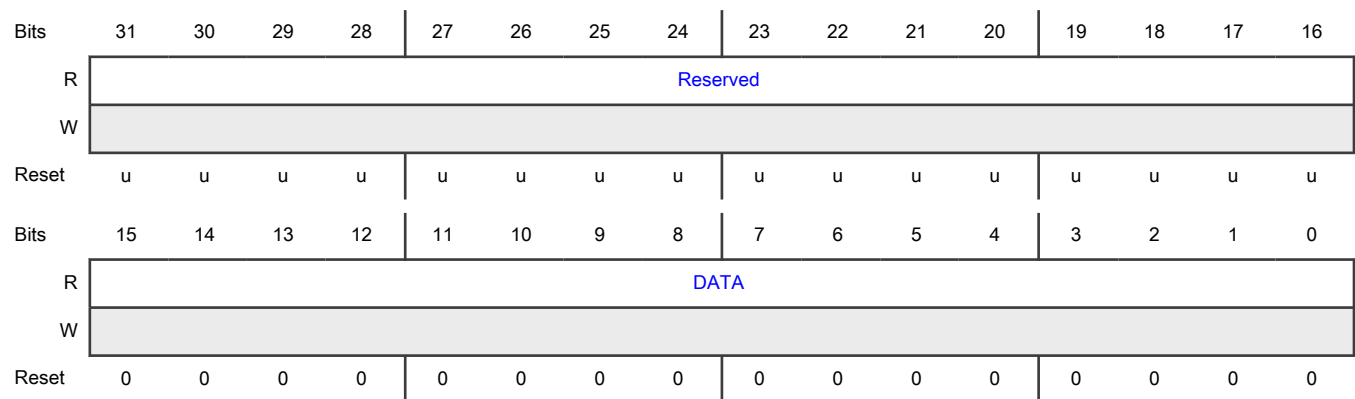
52.7.44 Master Read Message in SDR mode (MRMSG_SDR)

Offset

Register	Offset
MRMSG_SDR	D4h

Function

Read 16-bit words from a slave in Single Data Rate (SDR) message mode. The MRMSG_SDR register is used to read 16-bit words from an active message started with MWMSG_SDR. These words are intended to be read by DMA.

Diagram**Fields**

Field	Function
31-16 —	Reserved
15-0 DATA	Data DATA is the 16-bit word read from the slave. • If the length (LEN) was an uneven number, then the upper byte will be 0. • If the slave ends before LEN is finished, then it will treat the read as completed. • If the slave is not done before LEN is finished and END is not a continuation, then the read will be terminated.

52.7.45 Master Write Message in DDR mode (MWMSG_DDR_CONTROL)**Offset**

Register	Offset
MWMSG_DDR_CONTR OL	D8h

Function

Set up and write 16-bit words in Double Data Rate (DDR) mode. The MWMSG_DDR_ register is modal, and has 2 modes: control and data.

- On the first write to set up a new message, this register functions as the MWSMSG_DDR_CONTROL register.
- On subsequent writes, this register functions as the [MWMSG_DDR_DATA](#) register.

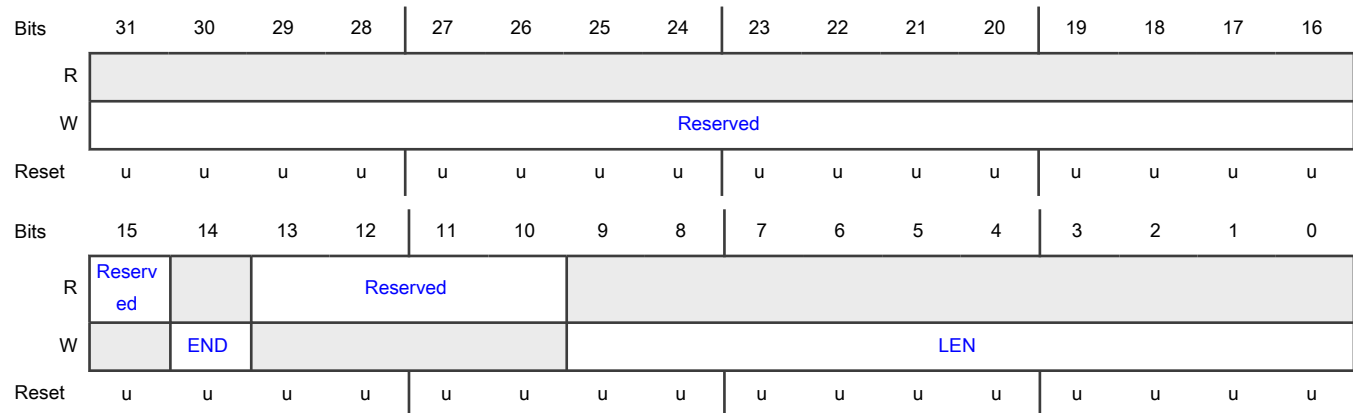
When not in the middle of a message, the MWMSG_DDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_DDR_DATA register is used until the Length (see LEN field) counts down, or until data with END=1 is used.

The MWMSG_DDR_CONTROL register is written with the 16-bit Control word if currently stopped or past the end of the previous message. If MSTATUS.STATE is MSGDDR and MSTAT.BETWEEN is 0, then the register (at this offset address) is functioning as

the MWMSG_DDR_DATA register; otherwise, this register (at this offset address) is functioning as the MWMSG_DDR_CTRL register, as long as MSTATUS.STATE is not in another mode. The main control word contains the 16-bit word length, and how it ends (stop, ready for next, continuation with more length). Then the 1st data word has the command and address for read or write. Note that if START and if an event (IBI, MR, HJ) occurs, then the MCTRL register's IBIRESP field will be used to determine action, and the corresponding interrupt will occur. The message will be restarted in that case. Note that the module handles preamble, parity, and CRC.

The MWMSG_DDR_CONTROL and MWMSG_DDR_DATA registers are oriented to DMA operations; but the MWMSG_DDR_CONTROL register can also be written by the processor if desired (instead of using MCTRL and MxDATAB).

Diagram



Fields

Field	Function
31-16 —	Reserved
15 —	Reserved
14 END	End of message - If END=1, then the double data rate message ends on HDR Exit. - If END=0, then the double data rate message ends waiting for a new DDR message (will issue a HDR Restart for the new message). Sets MSTATUS.COMPLETE when done. This can be happen before LEN bytes are read if the slave ends sooner.
13-10 —	Reserved
9-0 LEN	Length of message The length of the message (including the command) in half-words, so up to 2046 bytes. If LEN=0, then END is applied only. For Read, +1 for the CRC; for example, to read 4 bytes, use 1+2+1 for CMD + 2 halves + CRC.

52.7.46 Master Write Message Data in DDR mode (MWMSG_DDR_DATA)

Offset

Register	Offset
MWMSG_DDR_DATA	D8h

Function

This is the 16-bit word to be written in Double Data Rate (DDR) mode. This register also functions as the [MWMSG_DDR_CONTROL](#) register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																END
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA16B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-17 —	Reserved
16 END	End of message - If END=1, then END bit marks the last byte of the message. - If END=0, then the normal message count is used. The default is to use END=0 and let the header count count down normally. END allows terminating a write earlier than MWMSG_DDR_CONTROL.LEN field does.
15-0 DATA16B	Data

52.7.47 Master Read Message in DDR mode (MRMSG_DDR)

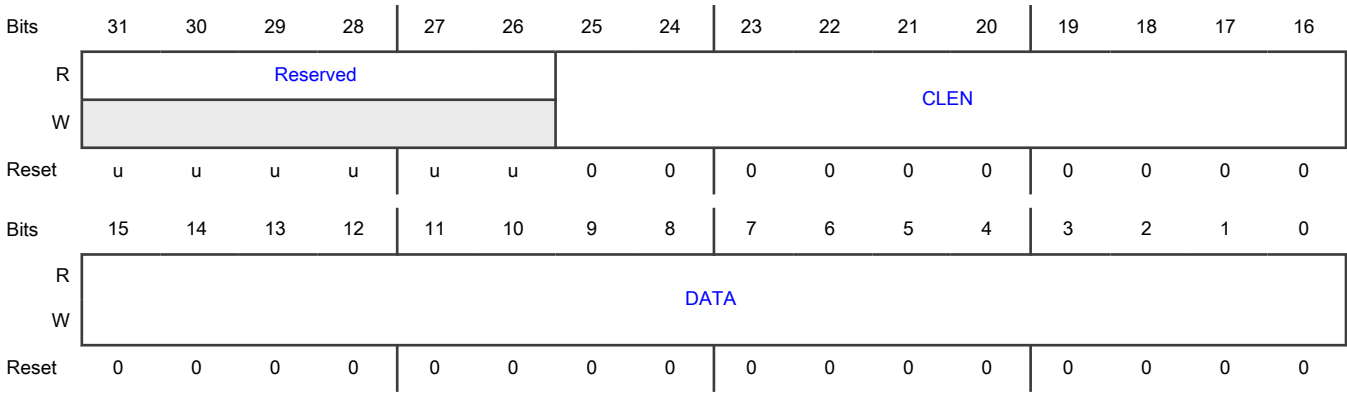
Offset

Register	Offset
MRMSG_DDR	DCh

Function

Read 16-bit words from a slave in Double Data Rate (DDR) message mode. The MRMSG_DDR register is used to read 16-bit words from an active message started with MWMSG_DDR. This is intended to be read by DMA.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 CLEN	Current length Current length. The DMA would normally only read 16 bits, and so not use the CLEN field. However, a processor can read CLEN to determine the current message length (for example, to see what the message ended on).
15-0 DATA	Data DATA is the 16-bit word read from a slave: the 1st byte is the LSB and the 2nd byte is the MSB. The 1st byte is in DATA[7:0] and the 2nd byte is in DATA[15:8]. • If the slave ends before the entire length of the message (MWMSG_DDR.LEN) is read, then the module will consider the DATA read as completed. In I3C, the slave can indicate the end of message (the last byte); otherwise, the master would terminate the message if the message is more than the master will accept. • If the slave has not yet finished sending DATA before the entire length of the message (MWMSG_DDR.LEN) is read and END is not a continuation, then the DATA read will be terminated.

52.7.48 Master Dynamic Address Register (MDYNADDR)

Offset

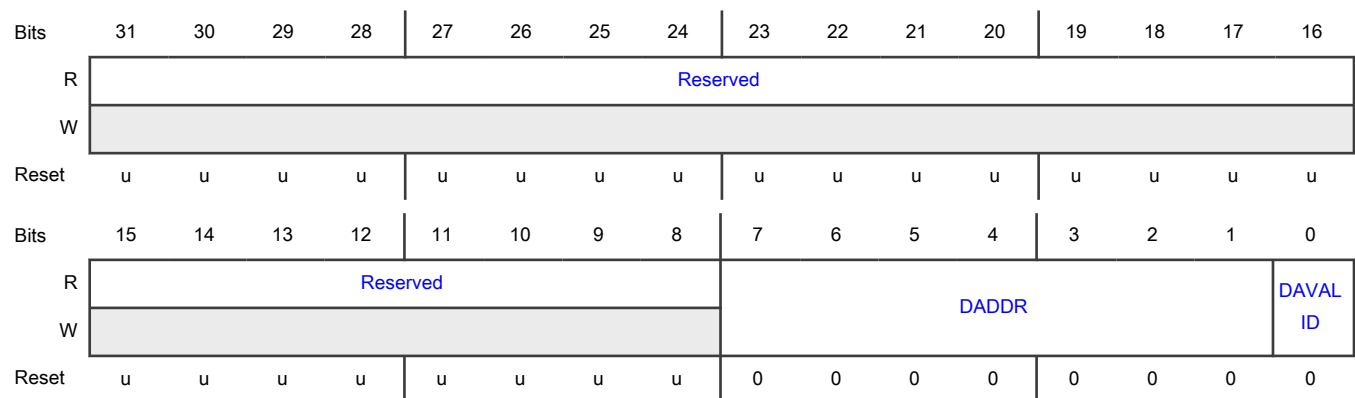
Register	Offset
MDYNADDR	E4h

Function

Allows an I3C module to write its own Dynamic Address when the I3C module changes from master mode to slave mode.

If this device is the Main Master (the Master at bus initialization), then this device may use this mechanism to assign itself its own Dynamic Address, for cases where the device hands off control to a secondary Master, and so becomes a Slave itself. This should be written before switching to Slave mode and should not be changed once in Slave mode (it is not clock safe for that). It should only be written with a valid address value in DADDR if DAVALID=1. Note that the Main master will also use DEFSLVS to define the slave addresses, including itself; this is how secondary Masters know this address. If the Master is not the Main Master, then this should not be used. If this peripheral needs to retain its DA for use during power cycle (when this peripheral is Slave), then the Slave DYNADDR register (SDYNADDR) should be used.

Diagram



Fields

Field	Function
31-8 —	Reserved
7-1 DADDR	Dynamic address The assigned Dynamic Address when DAVALID is 1.
0 DAVALID	Dynamic address valid DAVALID=1 if a Dynamic Address is assigned.

52.7.49 Slave Module ID (SID)

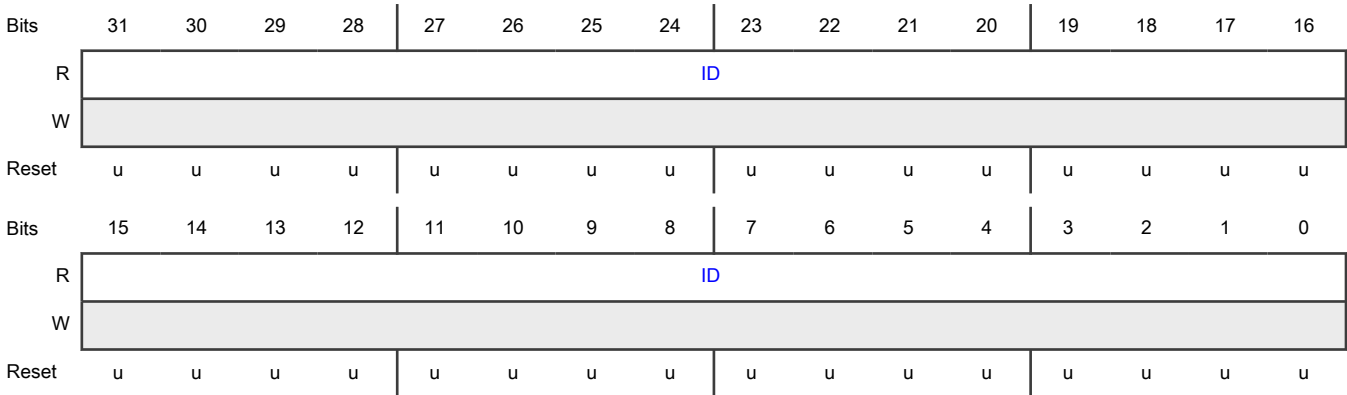
Offset

Register	Offset
SID	FFCh

Function

Allows software to detect the module and its version information.

Diagram



Fields

Field	Function
31-0	ID
ID	The ID meaning is specific to each use of the I3C module.

Chapter 53

Flexcomm Serial Communication

53.1 Chip-specific Flexcomm information

Table 545. Reference links to related information

Topic	Related module	Reference
Full description	Flexcomm	Flexcomm
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

53.1.1 Module instances

This device has 12 standard Flexcomm modules and 3 dedicated Flexcomm modules. Each Flexcomm module has a separate base address.

- Flexcomm modules 0-8 and 10-12 can be configured by the user for one of these protocols: USART, SPI, I²C, or I²S.

NOTE

Please disregard references to Flexcomm 9 and 13 as they are not pinned out.

- Flexcomm module 14 and 16 are dedicated high-speed SPI protocol modules (HS_SPI0 and HS_SPI1 respectively).
- Flexcomm module 15 is a dedicated I²C module intended for communication with the Power management IC (PMIC).

Specific part numbers and package variations may include a subset of this list.

Table 546. Flexcomm module base addresses and functions

Flexcomm number	Base address	USART	SPI	I ² C	I ² S
0	0x4010_6000	Yes	Yes	Yes	Yes
1	0x4010_7000	Yes	Yes	Yes	Yes
2	0x4010_8000	Yes	Yes	Yes	Yes
3	0x4010_9000	Yes	Yes	Yes	Yes
4	0x4012_2000	Yes	Yes	Yes	Yes
5	0x4012_3000	Yes	Yes	Yes	Yes
6	0x4012_4000	Yes	Yes	Yes	Yes
7	0x4012_5000	Yes	Yes	Yes	Yes
8	0x4020_9000	Yes	Yes	Yes	Yes

Table continues on the next page...

Table 546. Flexcomm module base addresses and functions (continued)

Flexcomm number	Base address	USART	SPI	I ² C	I ² S
9	0x4020_A000	Not applicable	Not applicable	Not applicable	Not applicable
10	0x4020_B000	Yes	Yes	Yes	Yes
11	0x4020_C000	Yes	Yes	Yes	Yes
12	0x4020_D000	Yes	Yes	Yes	Yes
13	0x4020_E000	Not applicable	Not applicable	Not applicable	Not applicable
14	0x4012_6000	-	Yes (HS_SPI0)	-	-
15	0x4012_7000	-	-	Yes (PMIC-I2C)	-
16	0x4012_8000	-	Yes (HS_SPI1)	-	-

53.1.2 Initialization

To initialize the Flexcomm module:

1. For the Peripheral clock: Enable the related Flexcomm module in the AHBCLKCTRL1 register.
2. For the Flexcomm module clock: Select a clock source for the related Flexcomm module.
3. Select the desired Flexcomm function by writing to the PSELID register of the chosen Flexcomm module.
4. If using DMA, set the INPUTMUX DMACx_REQ_ENAx bit corresponding to the Flexcomm instance. For some Flexcomm instances, a DMA device selection must also be made in the DMACx_CHMUX_SEL registers.

See specific peripheral subsections for information on configuration:

- [USART](#)
- [SPI](#)
- [I²C](#)
- [I²S](#)

53.1.3 Chip-specific USART information

53.1.3.1 USART initialization

To initialize the USART module:

- If needed, use the PRESETCTRL1 register to reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the desired Flexcomm function by writing to the PSELID register of the related Flexcomm module.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
 - Enable or disable the related Flexcomm interrupt in the NVIC.
 - Configure the related Flexcomm pin functions through IOPCTL.

— Configure the Flexcomm clock (FCLK) and USART baud rate.

NOTE

The Flexcomm clock frequency should not be above 100 MHz.

- Configure the USART to wake up the part from low power modes.
- Configure the USART to receive and transmit data in synchronous slave mode.

53.1.3.2 Configure the Flexcomm clock (FCLK) and USART baud rate

Each Flexcomm module has a separate Flexcomm clock selection, which can include a shared fractional divider (FRG). The Flexcomm clock (FCLK) and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by Syscon register FRGCTRL). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

$$\text{FCLK} = (\text{FRG input clock}) / (1 + (\text{MULT} / \text{DIV}))$$

The following rules apply for MULT and DIV:

- Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
 - Set the MULT to any value between 0 and 255.
2. In asynchronous mode: configure the baud rate divider with BRG[BRGVAL]. The baud rate divider modifies the FCLK to create the clock needed to produce the desired baud rate.

Generally: baud rate = [FCLK / oversample rate] / BRG divide

With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

Generally: BRG divide = [FCLK / oversample rate] / baud rate

With specific register values: BRGVAL = [(FCLK / (OSRVAL + 1)) / baud rate] - 1

3. In synchronous master mode: The serial clock is SCLK = FCLK / (BRGVAL+1). Note that if the USART BRG is set to 0, the FCLK input to the USART is sent directly to the SCLK pin in synchronous master mode. If the FRG is used to divide the source clock to produce the Flexcomm FCLK, that clock will not have a 50% duty cycle. Therefore if the FRG is used, the BRG must also be set to divide the clock by some integer factor before it is used.

The USART can also be clocked by the 32 kHz RTC oscillator. Set the CFG[MODE32K] bit to enable this 32 kHz mode.

53.1.3.3 Configure the USART for wake-up

A USART can wake up the system from Sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In Deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks (when it is in Deep-sleep mode).

As long as the USART receives a clock signal from the master, USART can receive up to one byte in the RXDAT register while in Deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.

- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

53.1.3.3.1 USART Wake-up from Sleep mode

- Configure the USART in either Asynchronous mode or Synchronous mode.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from Sleep mode.

53.1.3.3.2 USART Wake-up from Deep-sleep mode

- Configure the USART in Synchronous Slave mode. The SCLK function must be connected to a pin and also connect that pin to the master. Alternatively, the 32 KHz mode can be enabled and the USART operated in Asynchronous mode with the 32 KHz RTC oscillator.
- Enable the USART interrupt in the STARTEN0 register in SYSCON.
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from Deep-sleep mode on all events that are enabled and cause an interrupt. Typical wake-up events are:
 - A start bit has been received.
 - Received data becomes available.
 - In Synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
 - A change in the state of the CTS pin if the CTS function is connected.

NOTE

By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform a needed DMA before returning to Deep-sleep mode, without waking the CPU. To accomplish this:

- Set up the appropriate USART function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller, including a transfer complete interrupt.
- Disable the related USART interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in SYSCON.

53.1.3.4 USART signals description

The USART receive, transmit, and control signals are movable Flexcomm module functions assigned to external pins through IOPCTL.

Table 547. USART signals

Signal	I/O	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm module. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm module. Serial receive data.
RTS_B	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm module. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.

Table continues on the next page...

Table 547. USART signals (continued)

Signal	I/O	Name used in Pin Configuration chapter	Description
CTS_B	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm module. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART completes transmitting any character already in progress, then stops until CTS is again asserted (low).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm module in synchronous mode. Clock input or output in synchronous mode. NOTE When the USART module is configured as a master, such that SCK is an output, it must actually be connected to a pin for the USART to work properly.

53.1.3.5 Suggested I/O USART pin configuration

Recommended USART pin settings are shown in the next table.

Table 548. Suggested USART pin settings

IOPCTL bit(s)	Name	Comment
3 - 0	FSEL	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-up/pull-down resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 (pull-down selected). Unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless Pseudo open-drain output is desired).
11	IICENA	Set to 0 (input function not inverted).

53.1.4 Chip-specific SPI information

53.1.4.1 SPI initialization

To initialize the SPI peripheral :

- If needed, use the PRESETCTRL1 register to reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the desired Flexcomm module function by writing to the PSELID register of the related Flexcomm module.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
 - Enable or disable the related Flexcomm module interrupts in the NVIC.
 - Configure the required Flexcomm pin functions through IOPCTL.
 - Configure the Flexcomm clock and SPI data rate.

NOTE

The Flexcomm function clock frequency should not be above 48 MHz.

- Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.
- Configure the SPI function to wake up the part from low power modes.

53.1.4.2 Configure the SPI for wake-up

In Sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in Sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in Master or Slave mode.

In Deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in Slave mode and an external master provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

53.1.4.2.1 SPI Wake-up from sleep mode

- Configure the SPI in either master or slave mode.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from Sleep mode.

53.1.4.2.2 SPI Wake-up from Deep-sleep mode

- Configure the SPI in Slave mode. The SCK function must be connected to a pin and the pin connected to the master.
- Enable the SPI interrupt in the STARTER0 register.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
 - A change in the state of the SSEL pins.
 - Data available to be received.
 - Receive FIFO overflow.

SPI Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to Deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Set up the appropriate SPI function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller appropriately, including a transfer complete interrupt.

- Disable the related SPI interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in Syscon.

53.1.4.3 SPI Signals

The SPI signals are movable Flexcomm module functions and are assigned to external pins via IOPCTL.

Table 549. SPI Signals

Signal	I/O	Pin name used in Pin Description chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for SPI on Flexcomm Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
	I/O	HS_SPI0_SCK (FC14), HS_SPI1_SCK (FC16)	High Speed Serial Clock for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.
MOSI	I/O	FCn_RXD_SDA_MOSI_DATA	Master Out Slave In for SPI on Flexcomm Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
	I/O	HS_SPI0_MOSI_DATA (FC14), HS_SPI1_MOSI_DATA (FC16)	High Speed Master Out Slave In for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.
MISO	I/O	FCn_TXD_SCL_MISO_WS	Master In Slave Out for SPI on Flexcomm Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
	I/O	HS_SPI0_MISO_WS (FC14), HS_SPI1_MISO_WS (FC16)	High Speed Master In Slave Out for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.
SSEL0	I/O	FCn_CTS_SDA_SSEL0	Slave Select 0 for SPI on Flexcomm Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
	I/O	HS_SPI0_SSEL0 (FC14), HS_SPI1_SSEL0 (FC16)	High Speed Slave Select 0 for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.

Table continues on the next page...

Table 549. SPI Signals (continued)

Signal	I/O	Pin name used in Pin Description chapter	Description
SSEL1	I/O	FCn_RTS_SCL_SSEL1	Slave Select 1 for SPI on Flexcomm Interface n.
	I/O	HS_SPI0_SSEL1 (FC14), HS_SPI1_SSEL1 (FC16)	High Speed Slave Select 1 for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.
SSEL2	I/O	FCn_SSEL2	Slave Select 2 for SPI on Flexcomm Interface n.
	I/O	HS_SPI0_SSEL2 (FC14), HS_SPI1_SSEL2 (FC16)	High Speed Slave Select 2 for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.
SSEL3	I/O	FCn_SSEL3	Slave Select 3 for SPI on Flexcomm Interface n.
	I/O	HS_SPI0_SSEL3 (FC14), HS_SPI1_SSEL3 (FC16)	High Speed Slave Select 3 for SPI on Flexcomm Interface 14 and Flexcomm Interface 16.

53.1.4.4 Suggested I/O SPI pin configuration

Recommended SPI pin settings are shown in the next table.

Table 550. Suggested SPI pin settings

IOPCTL bit(s)	Name	Comment
3 - 0	FSEL	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-up/pull-down resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 (pull-down selected). Unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless Pseudo open-drain output is desired).
11	IENA	Set to 0 (input function not inverted).

53.1.5 Chip-specific I²C information

53.1.5.1 I²C initialization

Initial configuration of the I²S peripheral is accomplished as follows:

- Enable and configure the Flexcomm module as noted [Initialization](#)
- Configure the I²C for the desired functions:
 - Enable or disable the related Flexcomm interrupts in the NVIC.
 - Configure the related Flexcomm pin functions via IOPCTL.

- Configure the I²C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME for master mode.

NOTE

The Flexcomm function clock frequency should not be above 50 MHz

NOTE

The I²C function of the Flexcomm Module does not use the Flexcomm module FIFOs.

53.1.5.2 I²C signals

The I²C signals are movable Flexcomm module functions and are assigned to external pins via IOPCTL.

Table 551. SPI Signals

Signal	I/O	Pin name used in Pin Description chapter	Description
SCL	I/O	FCn_TXD_SCL_MISO_WS, or FCn_RTS_SCL_SSEL1	I ² C serial clock
SDA	I/O	FCn_RXD_SDA_MOSI_DATA, or FCn_CTS_SDA_SSEL0	I ² C serial data

53.1.5.3 Suggested I/O I²C pin configuration

Recommended I²C pin settings are shown in the next table.

Table 552. Suggested I²C pins settings

IOPCTL bit(s)	Name	Comment
3 - 0	FSEL	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-up/pull-down resistor not enabled).
5	PUPDSEL	Set to 0 (pull-down selected).
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 1 (Pseudo open-drain output).
11	IIENA	Set to 0 (input function not inverted).

53.1.6 Chip-specific I²S information

53.1.6.1 I²S initialization

To initialize the I²S module:

- Peripheral clock: Make sure that the I²S module is enabled in the AHBCLKCTRL1 register.
- Flexcomm clock: Select a clock source for the I²S module.

NOTE

The Flexcomm clock frequency should not be above 48 MHz.

- If needed, use the PRESETCTRL1 register to reset the Flexcomm module to I²S.
- Select the I²S function by writing to the PSELID register of the related Flexcomm module.
- Pins: Make sure that the IOPCTL block is enabled in the AHBCLKCTRL0 register. Select I²S pins and pin modes through the relevant IOPCTL registers.

In more complex applications, I²S bridging (bypass) and signal sharing connectivity is available to help conserve pins, see the I²S bridging and signal sharing section in the System Controller (SYSCTL) chapter.

- I²S rate: For master operation, the I²S rate is determined by the clock selected for Flexcomm clock above, optionally modified using the DIV register. Slave functions typically use the incoming I²S clock directly.
- Interrupts: To enable I²S channel pair interrupts, see the FIFOINENSET, FIFOINTENCLR, and FIFOINTSTAT registers. The related Flexcomm module interrupt must be enabled in the NVIC using the appropriate Interrupt Set Enable register.
- DMA: I²S channel pair master and slave functions can operate with the system DMA controller and must be enabled in the FIFOCFG register.

53.1.6.2 Configure the I²S for wake-up

In Sleep mode, any I²S interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and in the NVIC. As long as the I²S clock is configured to be active in Sleep mode, the I²S can wake up the part independently of whether the I²S block is configured in master or slave mode.

In Deep-sleep mode, I²S clocks are normally turned off. However, if the I²S is configured to use the MCLK input or as a slave receiving an external SCK, the I²S can generate an interrupt and wake up the device. The appropriate interrupt(s) must be enabled in the I²S and the NVIC.

53.1.6.2.1 I²S Wake-up from Sleep mode

- Configure the I²S for the desired mode and other operational details.
- Enable the I²S interrupt in the NVIC.
- Any enabled I²S interrupt wakes up the part from Sleep mode.

53.1.6.2.2 I²S Wake-up from Deep-sleep mode

- Configure the I²S for the desired mode and other operational details. The selected function clock of the I²S. (whether internal or externally sourced) must be running.
- Enable the I²S interrupt in the STARTER0 register.
- Enable the I²S interrupt in the NVIC.
- Any enabled I²S interrupt wakes up the part from sleep mode.

I²S Wake-up for DMA only

The device can optionally be woken up only far enough to perform a needed DMA before returning to Deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Configure the I²S for the desired mode and other operational details. The selected function clock of the I²S. (whether internal or externally sourced) must be running.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related I²S interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in SYSCON.

53.1.6.3 I²S signals description

Remark: When the I²S function is outputting SCK and/or WS, it uses a return signal from the related signal to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOPCTL selection, in order for the I²S to operate.

Table 553. I²S Pin Description

Signal	I/O	Name used in Pin Configuration chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for I2Sn. Clock signal used to synchronize the transfer of data on the SDA pin. It is normally driven by the master and received by one or more slaves. Remark: When the primary I ² S channel pair of a Flexcomm Interface is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the I ² S to work properly.
WS	I/O	FCn_TXD_SCL_MISO_WS	Word Select for I2Sn. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. Remark: When the primary I ² S channel pair of a Flexcomm Interface is configured as a master, such that WS is an output, it must actually be connected to a pin in order for the I ² S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA	Serial Data for a single data stream used by one or more I ² S channel pairs of I2Sn. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	MCLK	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm Interface in order to locally generate SCK and/or WS. This clock is not created inside the I ² S block. If MCLK is supported as an input to the device, it can be routed to the I ² S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I ² S block and used to operate its functions.

53.1.6.4 Suggested I/O I²S pin configuration

Recommended I²S pin settings are shown in the next table.

Table 554. Suggested I²S pins settings

IOPCTL bit(s)	Name	Comment
3 - 0	FSEL	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-up/pull-down resistor not enabled).
5	PUPDSEL	Set to 0 (pull-down selected).
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).
8	FULLDRIVE	Generally, set to 0 (normal output drive).

Table continues on the next page...

Table 554. Suggested I²S pins settings (continued)

IOPCTL bit(s)	Name	Comment
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 (unless pseudo open-drain output is desired).
11	IIENA	Set to 0 (input function not inverted).

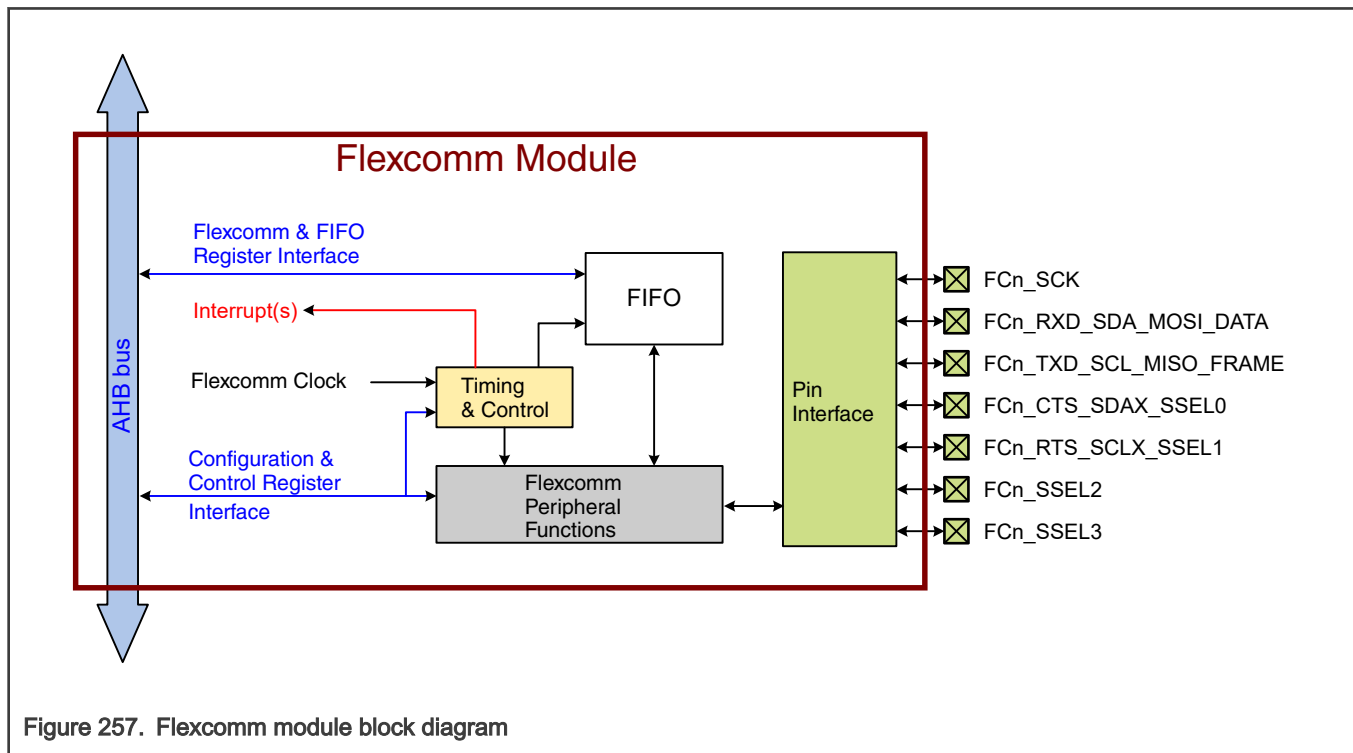
53.2 Flexcomm Overview

53.2.1 Introduction

This chapter describes the overall Flexcomm module and how to choose and configure each function. Multiple Flexcomm modules are available. Each Flexcomm module provides one peripheral function from a choice of several. Details of the different peripheral functions are found in the separate subsections: USART, SPI, I²C, and I²S.

53.2.1.1 Block diagram

The Flexcomm module block diagram shows the overall structure of one Flexcomm module.



53.2.1.2 Features

The Flexcomm modules provide a choice of peripheral functions:

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to 4 slave selects.
- I²C, including separate master, slave, and monitor functions.
- Some Flexcomm modules may also provide an I²S function. If so, there may be from one to four I²S channel pairs, one of which may optionally be a master and the rest slaves, configured together for either transmit or receive.
- Data for USART, SPI, and I²S traffic uses the Flexcomm FIFO. The I²C function does not use the FIFO.

53.2.2 Functional description

The following sections describe functional details of the Flexcomm module.

53.2.2.1 Choosing a peripheral function

Use software to select a peripheral function, from among those supported by a particular Flexcomm module.

- Choose the peripheral function by writing to the PSELID register.
- Read the PSELID register to see which peripheral functions are available on that Flexcomm module.

After software selects a specific peripheral function, the PID register supplies an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

53.2.2.1.1 FIFO usage

See the specific peripheral function for information on how the FIFO is used.

53.2.2.1.2 DMA

The Flexcomm module generates DMA requests if wanted, based on a selectable FIFO level. To enable the DMA requests, set the bit for the related Flexcomm module instance in the INPUTMUX DMA controller request enable register. For some Flexcomm module instances, a DMA device selection must also be made in the DMA controller channel mux select register.

53.2.3 Signals

Each Flexcomm module allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all 7 pin connections, and some Flexcomm instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the subsection for that peripheral.

Table 555. Flexcomm pins

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I ² S function (if present).
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I ² C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I ² S function (if present).
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I ² C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I ² S function (if present).
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.

Table continues on the next page...

Table 555. Flexcomm pins (continued)

Pin	Type	Description
	I/O	SDA (data) input/output for the I ² C function.
	I/O	Slave Select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I ² C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

53.2.4 Initialization

To Initialize the Flexcomm module:

1. Enable the peripheral clock for the Flexcomm module.
2. Select a clock source for the related Flexcomm module.
3. Select the desired Flexcomm function by writing to the PSELID register of the chosen Flexcomm module.
4. See specific peripheral subsections for information on configuration: USART, SPI, I²C, I²S.

53.2.5 Memory map and register definition

This section includes the Flexcomm module memory map and detailed descriptions of Flexcomm ID registers.

Each Flexcomm module contains registers that are related to the specific Flexcomm peripheral function, including registers related to peripheral FIFOs and data access. These registers depend on the chosen peripheral functions (USART, SPI, I²C, and I²S, if present in a specific Flexcomm module). They are described in the subsections for each Flexcomm peripheral. Each Flexcomm module has a separate base address.

NOTE

Generally, the bus interface to the Flexcomm registers (including the serial peripheral functions) support only word writes. Do not use byte and halfword writes. An exception is the FIFOWR register when used with SPI. When the Flexcomm module is configured for SPI, Flexcomm allows byte and halfword writes. This allows support for control information embedded in the DMA buffers. See the SPI subsections for more information.

53.2.5.1 Flexcomm register descriptions

53.2.5.1.1 Flexcomm memory map

FLEXCOMM0.SELECT base address: 4010_6000h

FLEXCOMM1.SELECT base address: 4010_7000h

FLEXCOMM2.SELECT base address: 4010_8000h

FLEXCOMM3.SELECT base address: 4010_9000h

FLEXCOMM4.SELECT base address: 4012_2000h

FLEXCOMM5.SELECT base address: 4012_3000h

FLEXCOMM6.SELECT base address: 4012_4000h

FLEXCOMM7.SELECT base address: 4012_5000h

FLEXCOMM8.SELECT base address: 4020_9000h

FLEXCOMM9.SELECT base address: 4020_A000h

FLEXCOMM10.SELECT base address: 4020_B000h

FLEXCOMM11.SELECT base address: 4020_C000h

FLEXCOMM12.SELECT base address: 4020_D000h

FLEXCOMM13.SELECT base address: 4020_E000h

FLEXCOMM14.SELECT base address: 4012_6000h

FLEXCOMM15.SELECT base address: 4012_7000h

FLEXCOMM16.SELECT base address: 4012_8000h

Offset	Register	Width (In bits)	Access	Reset value
FF8h	Peripheral Select and Flexcomm module ID (PSELID)	32	RW	See section
FFCh	Peripheral Identification (PID)	32	R	0000_0000h

53.2.5.1.2 Peripheral Select and Flexcomm module ID (PSELID)

Offset

Register	Offset
PSELID	FF8h

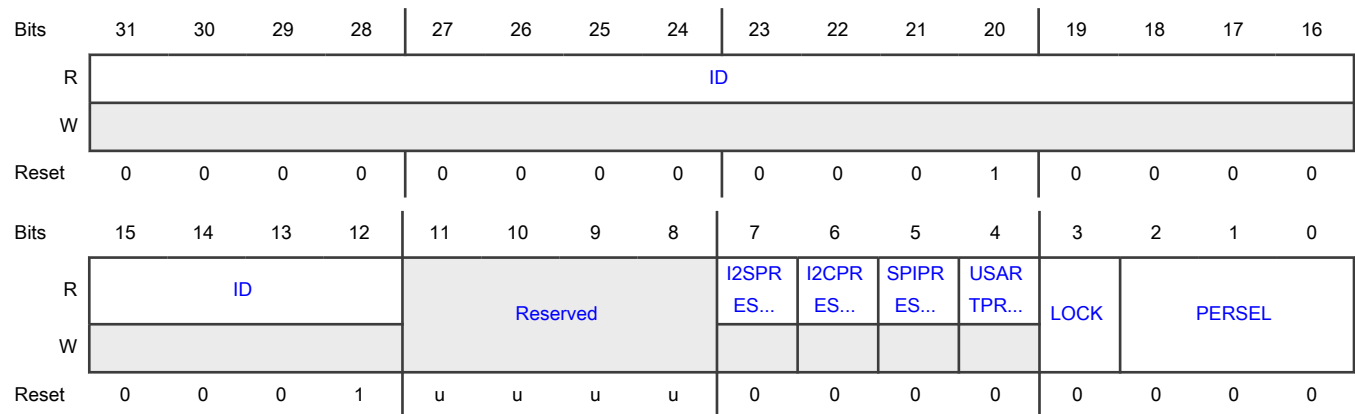
Function

The PSELID register identifies the Flexcomm module and the peripheral function supported by each Flexcomm module. The PSELID register provides the means to select one peripheral function for each Flexcomm module.

NOTE

Reset value reflects the data stored in used fields only. The reset value does not include reserved field content.

Diagram



Fields

Field	Function
31-12 ID	Flexcomm ID ID is the IP identifier. Bits 31 - 20 indicate the IP version. Bits 19 - 12 indicate the IP revision. Flexcomm ID = 0x00102
11-8 —	Reserved Read value is undefined, only zero should be written.
7 I2SPRESENT	I2S Present I ² S present indicator. This field is read-only. 0b - I2S Not Present. This Flexcomm module does not include the I ² S function. 1b - I2S Present. This Flexcomm module includes the I ² S function.
6 I2CPRESENT	I2C present indicator This field is read-only. 0b - I2C Not Present. This Flexcomm module does not include the I ² C function. 1b - I2C Present. This Flexcomm module includes the I ² C function.
5 SPIPRESENT	SPI present indicator This field is read-only. 0b - This Flexcomm module does not include the SPI function. 1b - This Flexcomm module includes the SPI function.
4	USART present indicator

Table continues on the next page...

Table continued from the previous page...

Field	Function
USARTPRESEN	This field is read-only. 0b - This Flexcomm module does not include the USART function. 1b - This Flexcomm module includes the USART function.
3 LOCK	Lock the peripheral select This field is writable by software. 0b - Peripheral select can be changed by software. 1b - Peripheral select is locked and cannot be changed until this Flexcomm module or the entire device is reset.
2-0 PERSEL	Peripheral Select This field is writable by software. 000b - No peripheral selected. 001b - USART function selected 010b - SPI function selected 011b - I2C. I ² C function selected 100b - I2S Transmit. I ² S transmit function selected 101b - I2S Receive. I ² S receive function selected 110b - Reserved 111b - Reserved

53.2.5.1.3 Peripheral Identification (PID)

Offset

Register	Offset
PID	FFCh

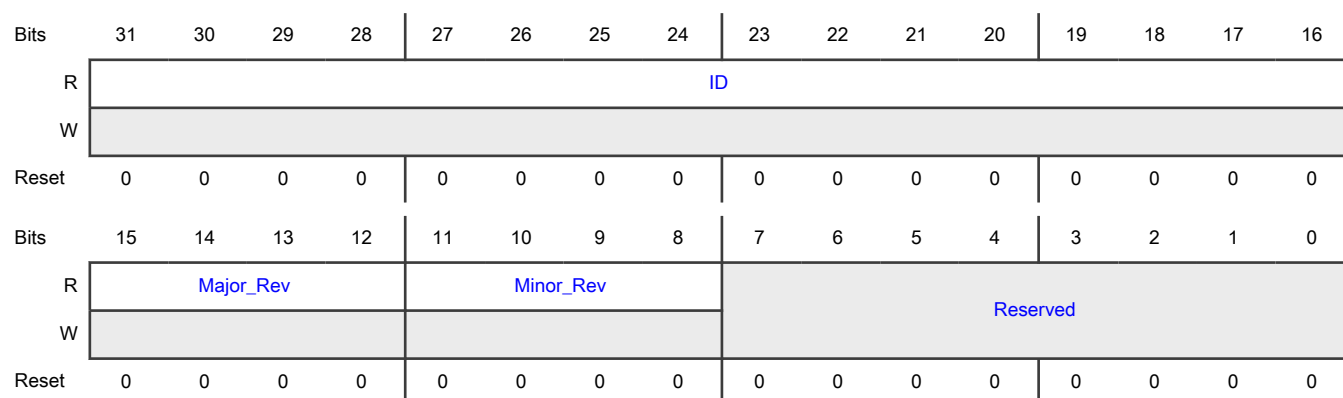
Function

After the Flexcomm module is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision-specific behavior.

NOTE

Reset value reflects the data stored in used fields only. The reset value does not include reserved field content.

This register is read-only and will read as 0 until a specific Flexcomm module function is selected through the PID register.

Diagram**Fields**

Field	Function
31-16 ID	Module identifier for the selected function
15-12 Major_Rev	Major revision of module implementation
11-8 Minor_Rev	Minor revision of module implementation
7-0 —	Reserved Read value is undefined, only zero should be written.

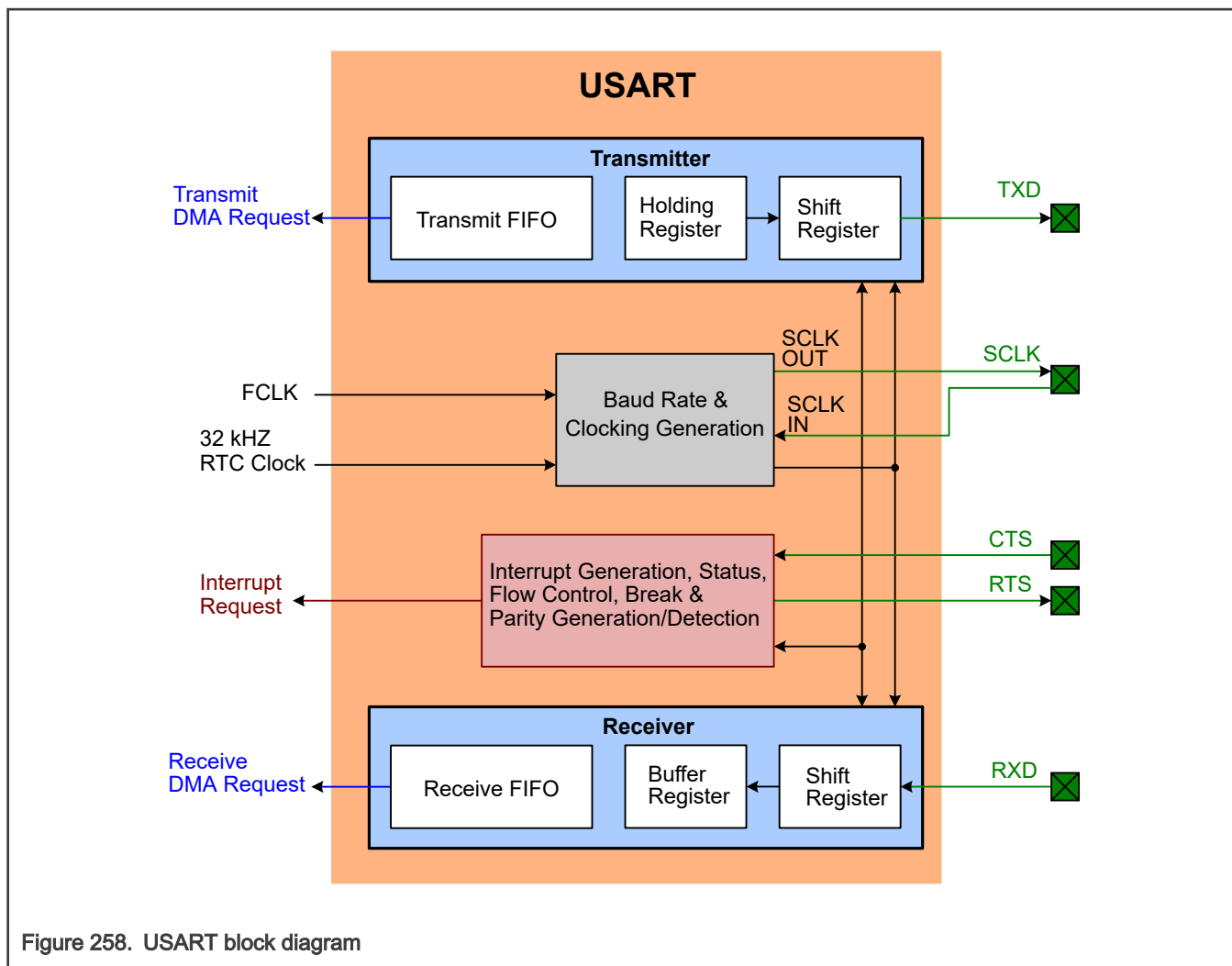
53.3 USART

53.3.1 Overview

This section provides an introduction to the USART function.

- The USART receiver monitors the serial input, RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.
- The USART transmitter accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, TXD.
- The Baud Rate Generator divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.
- In Synchronous Slave mode, data is transmitted and received using the serial clock directly. In Synchronous Master mode, data is transmitted and received using the baud rate clock without division.
- Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

53.3.1.1 Block diagram



53.3.1.2 Features

- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- One transmit and one receive data buffer
- USART transmit and receive functions can operated with the system DMA controller.
- Multiprocessor/multidrop (9-bit) mode with software address compare
- 7, 8, or 9 data bits and 1 or 2 stop bits
- RS-485 transceiver output enable

Transmit and receive FIFOs

- The USART function supports separate transmit and receive FIFOs with 16 entries each.
- Available Interrupts include: FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected.

Data checking

- Parity generation and checking: odd, even, or none

- Software selectable oversampling from 5 to 16 clocks in asynchronous mode
- Receive data is 2 out of 3 sample "voting". Status flag set when one sample differs.

Flow control

- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Loopback mode for testing of data and flow control
- Break generation and detection

Clocking

- Built-in Baud Rate Generator
- Auto-baud mode for automatic baud rate detection
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in Deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs

53.3.2 Functional description

53.3.2.1 Operating modes

This section describes all functional operation modes of the USART module.

53.3.2.1.1 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and the Baud Rate Generator (BRG), which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking or oversampling in synchronous mode.

53.3.2.1.2 Asynchronous mode

This section describes the oversampling and auto-baud function available in asynchronous mode.

53.3.2.1.2.1 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. The USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

53.3.2.1.2.2 Auto-baud function (asynchronous mode only)

The auto-baud function attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an auto-baud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Auto-baud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that auto-baud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off auto-baud.

When the USART is in synchronous mode, auto-baud has no meaning and should not be enabled.

53.3.2.1.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

53.3.2.2 Flow control

The USART supports both hardware and software flow control.

53.3.2.2.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signaling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled. RTS is asserted (active low) when the RX FIFO is not full AND the receiver is not receiving new data (i.e. RX is idle). RTS is de-asserted (high) when a new data transmission is in progress. When the new data is received and the RX FIFO is not full, RTS will be asserted indicating it is ready to receive more data. As soon as the new data transmission is started, RTS is de-asserted again and so on.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together. CTS is sampled continuously with the bit clock by the transmitter. The CTS from external device is synchronized with the bit clock. This CTS_sync is the CTS delayed by 1 to 2 bit clock cycles due to synchronization. When the current data transmission is completed and CTS_sync is asserted, the transmitter will continue to send new data if the TX FIFO is not empty. If CTS_sync is not asserted, the transmitter keeps sampling CTS, and will start sending new data as soon as CTS_sync is asserted.

This figure shows an overview of RTS and CTS within the USART.

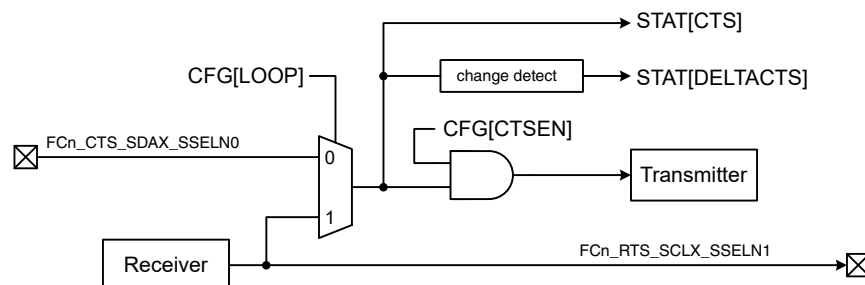


Figure 259. Hardware flow control using RTS and CTS

53.3.2.2.2 Software flow control

Software flow control could include XON/XOFF flow control, or other mechanisms. These are supported by the ability to check the current state of the CTS signal input (STAT[CTS]), and/or have an interrupt when the CTS signal changes state (STAT[DELTACTS]), and by the ability of software to gracefully turn off the transmitter (using the CTL[TXDIS] bit).

53.3.2.3 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see CFG[AUTOADDR] and the ADDR register), as well as software address recognition (see CTL[ADDRDET]).

To set up automatic data direction control with the RTS pin, use CFG[OESEL], CFG[OEPOL], and CFG[OETA]. Data direction control can also be implemented in software using a GPIO pin.

53.3.2.4 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the receive input remains low for 16 bit times. Both the beginning and end of a received break are noted by the STAT[DELTARXBRK] status flag, which can be used as an interrupt. See [LIN bus](#) for details of LIN mode break.

To avoid corrupting any character currently being transmitted:

1. Disable the transmitter: CTL[TXDIS] = 1
2. Wait until the transmitter is idle: STAT[TXDISSTAT] = 1
3. Start sending line break: CTL[TXBRKEN] = 1
4. Wait for the desired length
5. Stop sending line break: CTL[TXBRKEN] = 0

53.3.2.5 LIN bus

The only difference between standard mode and LIN (Local Interconnect Network) mode is that LIN mode alters the way that line or received break generation/detection is performed. See [Break generation and detection](#) for more information about the standard break process.

When a break is requested (by setting CTL[TXBRKEN] = 1 and then sending a dummy character) a 13-bit time break is sent.

- A received break is flagged when the RX input remains low for 11 bit times.
- For non-LIN mode (standard mode), a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the auto-baud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as a confirmation of synchronization success.

Wake-up for LIN can be handled in a number of ways, depending on the system, and which clocks may be running in a slave device. For example, as long as the USART is receiving internal clocks (allowing the USART to function), USART can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, then the GPIO function of the USART RX pin can be programmed to wake up the device.

53.3.2.6 Clocking and baud rates

For the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see [Configure the Flexcomm clock and USART baud rate](#)

53.3.2.6.1 Fractional Rate Generator (FRG)

Use the FRG to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

A typical set up has the FRG produce an integer multiple of the highest required baud rate, or a very close approximation. Then use the BRG to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm module. The FRG creates a lower rate output clock by suppressing selected input clocks. When FRG is not needed, set FRG = 0, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by $1 + (\text{MULT} / 256)$, where MULT is in the range of 1 to 255. This produces an output clock that ranges from the input clock divided by $1 + 1/256$ to $1 + 255/256$ (just more than 1 to just less than 2). Make further clock division to each USART block with the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

53.3.2.6.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see [BRG register](#)) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm clock (FCLK) to produce the clock used for USART internal operations. A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz. Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data. Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

NOTE

To change a baud rate after a USART is running, use the following sequence:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing `CFG[ENABLE] = 0` (0 may be written to the entire register).
3. Write the new `BRG[BRGVAL]`.
4. Set `CFG[ENABLE] = 1`.

53.3.2.6.3 32 KHz mode

For the 32 KHz clock to operate a USART at a reasonable speed, make these changes:

- Do not use 16x overclocking, because then the maximum data rate would be very low.

- Multiple samples of each data bit must be reduced to one, also to maximize the data rate.
- Use special clocking for individual bit times because 32 KHz is not close to an integer multiple of any standard baud rate.

When the 32 KHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error are less than one half percent in this mode, when the RTC oscillator is operating at the intended frequency of 32.768 kHz.

53.3.3 Signals

The USART receive, transmit, and control signals are movable Flexcomm functions and are assigned to external pins through I/O pin configuration.

Table 556. USART signals

Signal	I/O	Description
TXD	O	Transmitter output for USART on Flexcomm module. Serial transmit data.
RXD	I	Receiver input for USART on Flexcomm module. Serial receive data.
RTS_B	O	Request To Send output for USART on Flexcomm module. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
CTS_B	I	Clear To Send input for USART on Flexcomm module. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	Serial clock input/output for USART on Flexcomm module in synchronous mode. Clock input or output in synchronous mode.
NOTE When the USART is configured as a master, such that SCK is an output, SCK must be connected to a pin in order for the USART to work properly.		

53.3.4 Initialization

To initialize the USART module:

- Reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the USART module function.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
 - Enable the clock to the USART register interface.
 - Enable or disable the related Flexcomm module interrupts in the SoC.
 - Configure the related Flexcomm module pin functions in the SoC.
 - Configure the Flexcomm module clock and USART baud rate.

NOTE

The Flexcomm clock frequency should not be above 48 MHz.

- Configure the USART to wake up the chip from any low power modes.
- Configure the USART to receive and transmit data in synchronous slave mode.

See the chip-specific USART information for configuration details.

53.3.4.1 Configure the Flexcomm clock and USART baud rate

Each Flexcomm module has a separate clock selection, which can include a shared fractional divider.

- If a fractional value is needed to obtain a particular baud rate, program the chip.
- In Asynchronous mode: configure the baud rate divider BRG[BRGVAL]. The baud rate divider divides the Flexcomm Clock (FCLK) to create the clock needed to produce the desired baud rate.
- In Synchronous Master mode: The serial clock is $SCLK = FCLK / (BRGVAL + 1)$.

The USART can also be clocked by the 32 KHz RTC oscillator. Set the MODE32K bit to enable this 32 KHz mode. See [32 KHz mode](#)

53.3.4.2 Configure the USART for wake-up

A USART can wake up the system from Sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In Deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in Deep-sleep mode.

As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in Deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.

- If the 32 KHz mode is enabled, the USART can run in asynchronous mode using the 32 KHz RTC oscillator and create interrupts.

53.3.4.2.1 Wake-up from Sleep mode

- Configure the USART in either asynchronous mode or synchronous mode.
- Enable the USART interrupts in the SoC.
- Any enabled USART interrupt wakes up the part from Sleep mode.

53.3.4.2.2 Wake-up from Deep-sleep mode

- Configure the USART in synchronous slave mode.
- Enable the USART interrupt on the chip.
- The USART wakes up the chip from Deep-sleep mode on all events that are enabled and cause an interrupt. Typical wake-up events are:
 - A start bit has been received.
 - Received data becomes available.
 - In Synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
 - A change in the state of the CTS pin if the CTS function is connected.

NOTE

By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform a needed DMA before returning to Deep-sleep mode, without waking the CPU. To accomplish this:

- Set up the appropriate USART function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller, including a transfer complete interrupt.
- Disable the related USART interrupt on the chip.
- Enable the DMA interrupt on the chip.
- Enable the wake-up controls on the chip.

See Chip-specific USART information.

53.3.5 Memory map and register definition

This section includes the USART module memory map and detailed descriptions of all registers.

NOTE

When writing to USART registers, only word accesses are supported. Byte and halfword writes are not supported.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm module address space after the USART function has been selected for that Flexcomm module.

53.3.5.1 Flexcomm USART register descriptions

53.3.5.1.1 USART memory map

FLEXCOMM0.USART base address: 4010_6000h

FLEXCOMM1.USART base address: 4010_7000h

FLEXCOMM2.USART base address: 4010_8000h

FLEXCOMM3.USART base address: 4010_9000h

FLEXCOMM4.USART base address: 4012_2000h

FLEXCOMM5.USART base address: 4012_3000h

FLEXCOMM6.USART base address: 4012_4000h

FLEXCOMM7.USART base address: 4012_5000h

FLEXCOMM8.USART base address: 4020_9000h

FLEXCOMM9.USART base address: 4020_A000h

FLEXCOMM10.USART base address: 4020_B000h

FLEXCOMM11.USART base address: 4020_C000h

FLEXCOMM12.USART base address: 4020_D000h

FLEXCOMM13.USART base address: 4020_E000h

Offset	Register	Width (In bits)	Access	Reset value
0h	USART Configuration (CFG)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4h	USART Control (CTL)	32	RW	See section
8h	USART Status (STAT)	32	RW	See section
Ch	Interrupt Enable Read and Set for USART (not FIFO) Status (INTENSET)	32	RW	See section
10h	Interrupt Enable Clear (INTENCLR)	32	RW	See section
20h	Baud Rate Generator (BRG)	32	RW	See section
24h	Interrupt Status (INTSTAT)	32	R	See section
28h	Oversample Selection Register for Asynchronous Communication (OSR)	32	RW	See section
2Ch	Address Register for Automatic Address Matching (ADDR)	32	RW	See section
E00h	FIFO Configuration (FIFOCFG)	32	RW	See section
E04h	FIFO Status (FIFOSTAT)	32	RW	See section
E08h	FIFO Trigger Settings for Interrupt and DMA Request (FIFOTRIG)	32	RW	See section
E10h	FIFO Interrupt Enable (FIFOINTENSET)	32	RW	See section
E14h	FIFO Interrupt Enable Clear (FIFOINTENCLR)	32	RW	See section
E18h	FIFO Interrupt Status (FIFOINTSTAT)	32	R	See section
E20h	FIFO Write Data (FIFOWR)	32	RW	See section
E30h	FIFO Read Data (FIFORD)	32	R	See section
E40h	FIFO Data Read with No FIFO Pop (FIFORDNOPOP)	32	R	See section
E48h	FIFO Size (FIFOSIZE)	32	R	0000_0010h
FFCh	Peripheral Identification (ID)	32	R	E010_2100h

53.3.5.1.2 USART Configuration (CFG)

Offset

Register	Offset
CFG	0h

Function

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured only once in an application.

NOTE

Only the CFG register can be written when the ENABLE = 0. The CFG register can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

NOTE

If software needs to change configuration values, use this sequence:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing ENABLE = 0 (0 may be written to the entire register).
3. Write the new configuration value, with ENABLE = 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXPOL	RXPOL	OEPOL	OESSEL	AUTO ADDR	OETA	Reserved	
W									L	L	L	L				
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	u	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOOP	SYNC	Reserved	CLKPOL	SYNC	Reserved	CTSEN	LINMODE	MODE	STOP	PARITYSEL		DATALEN		Reserved	ENABLE
W		MST		OL	EN		N	DE	32K	LEN						LE
Reset	0	0	u	0	0	u	0	0	0	0	0	0	0	0	u	0

Fields

Field	Function
31-24 —	Reserved Read value is undefined; only zero should be written.
23 TXPOL	Transmit data polarity 0b - Standard. Standard. The TX signal is sent out without change. This means that the TX rest value is 1, the start bit is 0, data is not inverted, and the stop bit is 1. 1b - Inverted. Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
22 RXPOL	Receive Data Polarity 0b - Standard. Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1. 1b - Inverted. Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
21 OEPOL	Output Enable Polarity 0b - Low. If selected by OESSEL, the output enable is active low. 1b - High. If selected by OESSEL, the output enable is active high.
20	Output Enable Select

Table continues on the next page...

Table continued from the previous page...

Field	Function
OESEL	0b - Standard. The RTS signal is used as the standard flow control function. 1b - RS-485. The RTS signal is configured to provide an output enable signal to control an RS-485 transceiver.
19 AUTOADDR	Automatic Address Matching Enable 0b - Disabled. Disabled. When addressing is enabled by CTL[ADDRDET], address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address). 1b - Enabled. Enabled. When addressing is enabled by CTL[ADDRDET], address matching is done by hardware, using the value in the ADDR register as the address to match.
18 OETA	Output Enable Turnaround Time Enable for RS-485 Operation. 0b - Disabled. Disabled. If selected by OESEL, the Output Enable signal is deasserted at the end of the last stop bit of a transmission. 1b - Enabled. Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. The Output Enable signal also remains asserted if another transmit begins before Output Enable is deasserted.
17-16 —	Reserved Read value is undefined; only zero should be written.
15 LOOP	Loopback Mode Selects data Loopback mode. 0b - Normal operation 1b - Loopback mode. Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (TXD) is connected internally to serial input of the receiver (RXD). TXD and RTS activity also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.
14 SYNCMST	Synchronous mode Master Select 0b - Slave. When synchronous mode is enabled, the USART is a slave. 1b - Master. When synchronous mode is enabled, the USART is a master.
13 —	Reserved Read value is undefined; only zero should be written.
12 CLKPOL	Clock Polarity Selects the clock polarity and sampling edge of received data in synchronous mode. 0b - Falling edge. RXD is sampled on the falling edge of SCLK. 1b - Rising edge. RXD is sampled on the rising edge of SCLK.
11	Synchronous Enable. Selects synchronous or asynchronous operation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
SYNCEN	0b - Asynchronous mode 1b - Synchronous mode
10 —	Reserved Read value is undefined; only zero should be written.
9 CTSEN	CTS Enable Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if Loopback mode is enabled. 0b - No flow control. The transmitter does not receive any automatic flow control signal. 1b - Flow control enabled. The transmitter uses the CTS input (or RTS output in Loopback mode) for flow control purposes.
8 LINMODE	LIN Break Mode Enable 0b - Disabled. Break detect and generate is configured for normal operation. 1b - Enabled. Break detect and generate is configured for LIN bus operation.
7 MODE32K	Mode 32 kHz Selects standard or 32 kHz clocking mode. 0b - Disabled. USART uses standard clocking. 1b - Enabled. Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.
6 STOPLEN	Stop Length Number of stop bits appended to transmitted data. Only a single stop bit is required for received data. 0b - 1 stop bit 1b - 2 stop bits. This setting should be used only for asynchronous communication.
5-4 PARITYSEL	Parity Select. Selects what type of parity is used by the USART. 00b - No parity 01b - Reserved 10b - Even parity. Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even. 11b - Odd parity. Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.
3-2 DATALEN	Data Length. Selects the data size for the USART. 00b - 7 bit data length 01b - 8 bit data length

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - 9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET[CTL]. 11b - Reserved
1 —	Reserved Read value is undefined; only zero should be written.
0 ENABLE	USART Enable 0b - Disabled. Disabled. The USART is disabled and the internal state machine and counters are reset. While ENABLE = 0, all USART interrupts and DMA transfers are disabled. When ENABLE is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART is immediately ready to transmit because the transmitter has been reset and is therefore available. 1b - Enabled. The USART is enabled for operation.

53.3.5.1.3 USART Control (CTL)

Offset

Register	Offset
CTL	4h

Function

The CTL register controls USART operations that are more likely to change during operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R																AUTO	
W	Reserved															BAUD	
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserved								CLRC	Reserv	TXDIS	Reserved			ADDR	TXBR	Reserv
W	Reserved								CON...	ed		Reserved			DET	KEN	ed
Reset	u	u	u	u	u	u	0	0	u	0	u	u	u	0	0	u	

Fields

Field	Function
31-17	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined; only zero should be written.
16 AUTOBAUD	Autobaud Enable 0b - Disabled. Disabled. USART is in normal operating mode. 1b - Enabled. Enabled. USART is in Autobaud mode. AUTOBAUD should only be set when the USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.
15-10 —	Reserved Read value is undefined; only zero should be written.
9 CLRCCONRX	Clear Continuous Clock 0b - No effect. No effect on the CC bit. 1b - Auto-clear. Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.
8 CC	Continuous Clock Generation By default, SCLK is only output while data is being transmitted in synchronous mode. 0b - Clock on character. Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on TXD or to complete a character that is being received. 1b - Continuous clock. Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on RxD independently from transmission on TXD).
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDIS	Transmit Disable 0b - Not disabled. USART transmitter is not disabled. 1b - Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.
5-3 —	Reserved Read value is undefined; only zero should be written.
2 ADDRDET	Enable Address Detect Mode 0b - Disabled. The USART presents all incoming data. 1b - Enabled. Enabled. The USART receiver ignores incoming data that does not have the most significant bit (msb) of the data (typically the 9th bit) = 1. When the data msb = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.
1	Break Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXBRKEN	A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (CTL[TXDIS] = 1) and then waiting for the transmitter to be disabled (STAT[TXDISINT] = 1) before writing 1 to TXBRKEN. 0b - Normal operation 1b - Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared.
0	Reserved
—	Read value is undefined; only zero should be written.

53.3.5.1.4 USART Status (STAT)

Offset

Register	Offset
STAT	8h

Function

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits in STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register.

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															ABER
W																W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXNOI SE...	PARIT YE...	FRAM ERR...	STAR T	DELT ARX...	RXBR K	Reserv ed	Reserv ed	Reserv ed	TXDIS ST...	DELT ACTS	CTS	TXIDL E	Reserv ed	RXIDL E	Reserv ed
W	W1C	W1C	W1C	W1C	W1C						W1C					
Reset	u	u	u	u	u	0	u	u	u	0	u	0	1	u	1	u

Fields

Field	Function
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERR	Auto Baud Error An error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.
15 RXNOISEINT	Received Noise Interrupt Flag Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This sampling acts as a noise filter if one sample does not match. RXNOISEINT is set when a received data bit contains one nonmatching sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.
14 PARITYERRINT	Parity Error Interrupt Flag PARITYERRINT is set when a parity error is detected in a received character.
13 FRAMERRINT	Framing Error Interrupt Flag FRAMERRINT is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12 START	Start START is set when a start is detected on the receiver input. START's purpose is primarily to allow wake-up from Deep-sleep mode immediately when a start is detected. Cleared by software.
11 DELTARXBRK	Delta Received Break DELTARXBRK is set when a change in the state of receiver break detection occurs. Cleared by software.
10 RXBRK	Received Break RXBRK reflects the current state of the receiver break detection logic. RXBRK is set when the RXD pin remains low for 16 bit times. Note that FRAMERRINT is also set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the RXD pin goes high.
9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISSTAT	Transmitter Disabled Status Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Not Idle. Indicates that the USART transmitter is NOT fully idle after being disabled. 1b - Idle. Indicates that the USART transmitter is fully idle after being disabled (CTL[TXDIS] = 1).
5 DELTACTS	Delta CTS DELTACTS is set when the state of CTS changes. DELTACTS is cleared by software.
4 CTS	CTS value CTS reflects the current state of the CTS signal, regardless of the setting of the CFG[CTSEN] bit. The CTS bit has the same value as the value of the CTS input pin unless Loopback mode is enabled.
3 TXIDLE	Transmitter Idle TXIDLE is read only. 0b - The transmitter is currently sending data. 1b - The transmitter is not currently sending data.
2 —	Reserved Read value is undefined; only zero should be written.
1 RXIDLE	Receiver Idle RXIDLE is read only. 0b - The receiver is currently receiving data. 1b - The receiver is not currently receiving data.
0 —	Reserved Read value is undefined; only zero should be written.

53.3.5.1.5 Interrupt Enable Read and Set for USART (not FIFO) Status (INTENSET)

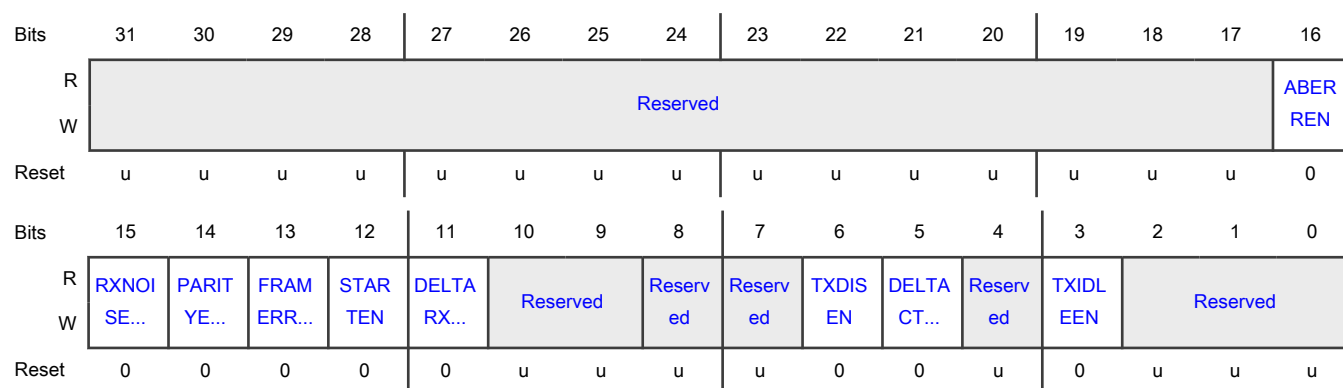
Offset

Register	Offset
INTENSET	Ch

Function

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in the INTENSET register are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing a 1 to any implemented bits in the INTENSET register causes those bits to be set. The INTENCLR register is used to clear bits in the INTENSET register.

Diagram



Fields

Field	Function
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERREN	Auto Baud Error Enable 1b - Enables an interrupt when an auto baud error occurs.
15 RXNOISEEN	Receive Noise Enable 1b - Enables an interrupt when noise is detected. See the description of the CTL[RXNOISEINT] bit.
14 PARITYERREN	Parity Error Enable 1b - Enables an interrupt when a parity error has been detected.
13 FRAMERREN	Frame Error Enable 1b - Enables an interrupt when a framing error has been detected.
12 STARTEN	Start Enable 1b - Enables an interrupt when a received start bit has been detected.
11 DELTARXBRKEN	Delta Receive Break Enable 1b - Enable. Enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).
10-9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined; only zero should be written.
6 TXDISEN	Transmit Disabled Flag 1b - Enables an interrupt when the transmitter is fully disabled as indicated by the STAT[TXDISINT] flag. See the description of the STAT[TXDISINT] flag.
5 DELTACTSEN	Delta CTS Input Flag 1b - Enables an interrupt when there is a change in the state of the CTS input.
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLEEN	Transmit Idle Flag 1b - Enables an interrupt when the transmitter becomes idle (STAT[TXIDLE] = 1).
2-0 —	Reserved Read value is undefined; only zero should be written.

53.3.5.1.6 Interrupt Enable Clear (INTENCLR)

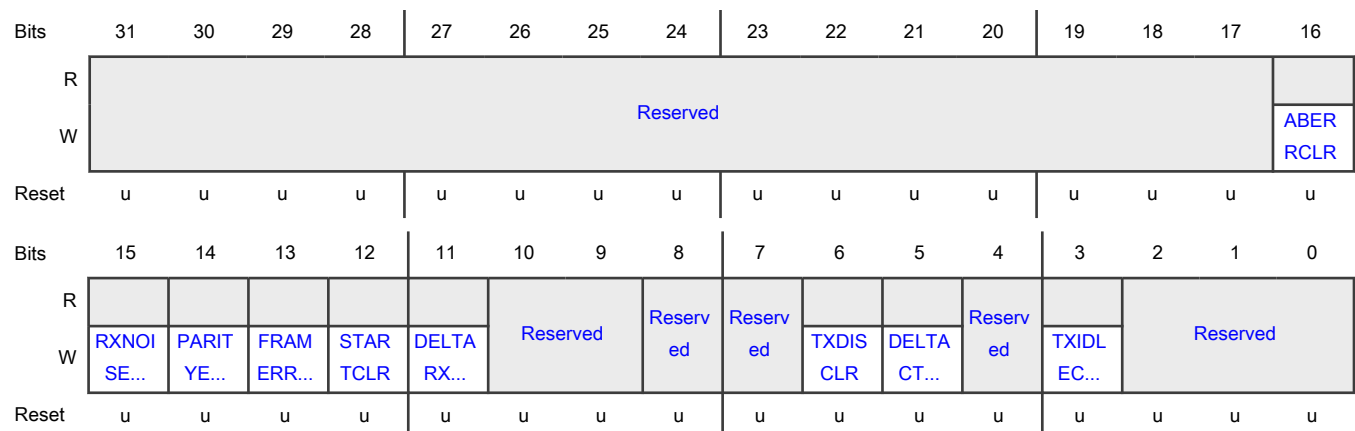
Offset

Register	Offset
INTENCLR	10h

Function

Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.

Diagram



Fields

Field	Function
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERRCLR	Auto Baud Error Clear Writing 1 clears INTENSET[ABERREN].
15 RXNOISECLR	Receive Noise Clear Writing 1 clears INTENSET[RXNOISEEN].
14 PARITYERRCLR	Parity Error Clear Writing 1 clears INTENSET[PARITYERREN].
13 FRAMERRCLR	Frame Error Clear Writing 1 clears INTENSET[FRAMERREN].
12 STARTCLR	Start Clear Writing 1 clears INTENSET[STARTEN].
11 DELTARXBRKCLR	Delta Receive Break Clear Writing 1 clears INTENSET[DELTARXBRKEN].
10-9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISCLR	Transmit Disable Clear Writing 1 clears INTENSET[TXDISEN].
5 DELTACTSCLR	Delta CTS Clear Writing 1 clears INTENSET[DELTACTSEN].
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLECLR	Transmit Idle Clear Writing 1 clears INTENSET[TXIDLEEN].

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0	Reserved
—	Read value is undefined; only zero should be written.

53.3.5.1.7 Baud Rate Generator (BRG)

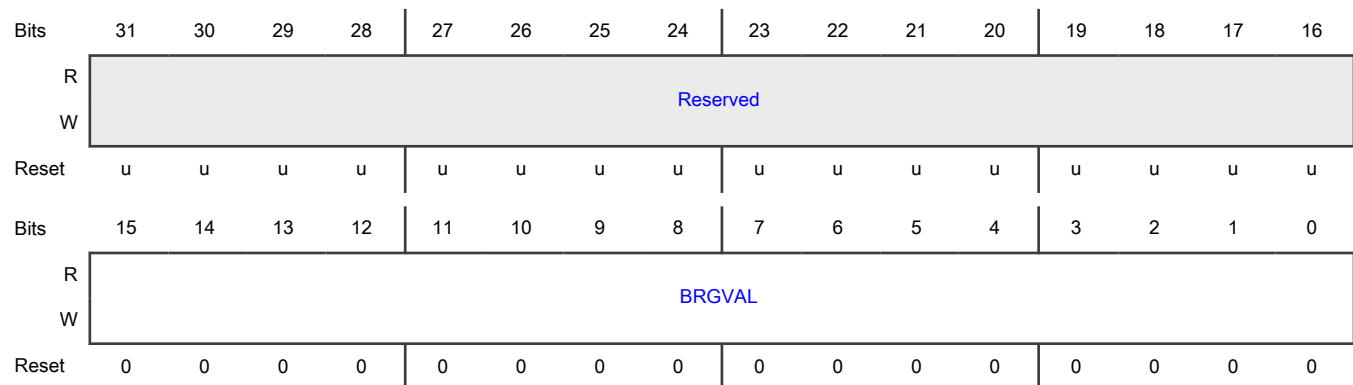
Offset

Register	Offset
BRG	20h

Function

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm Clock (FCLK) to produce the clock used for USART internal operations.

Diagram



Fields

Field	Function
31-16	Reserved
—	Read value is undefined; only zero should be written.
15-0 BRGVAL	Baud Rate Generator Value This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0000_0000_0000_0000b - FCLK is used directly by the USART function. 0000_0000_0000_0001b - FCLK is divided by 2 before use by the USART function. 0000_0000_0000_0010b - FCLK is divided by 3 before use by the USART function. 1111_1111_1111_1111b - FCLK is divided by 65,536 before use by the USART function.

53.3.5.1.8 Interrupt Status (INTSTAT)

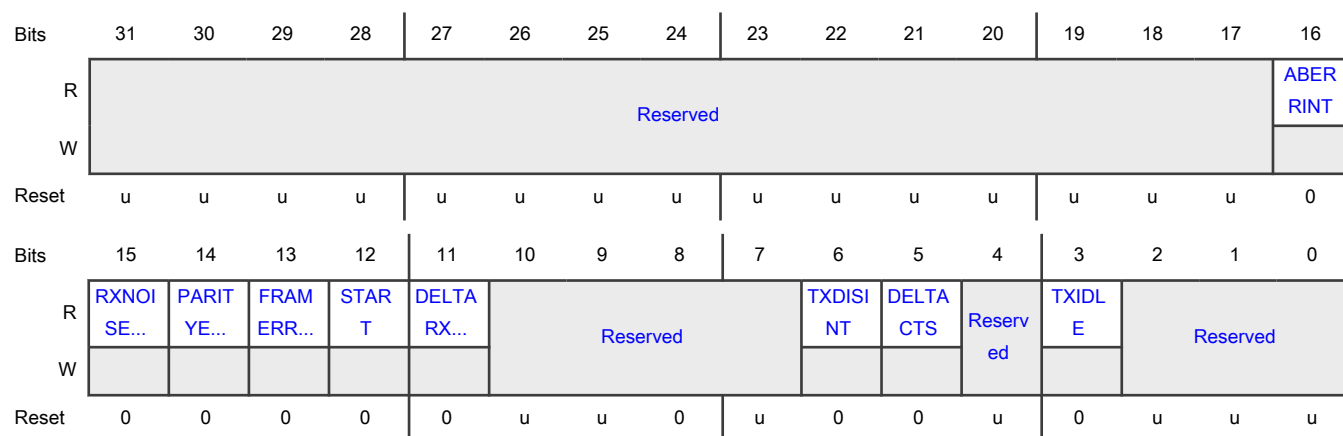
Offset

Register	Offset
INTSTAT	24h

Function

The read-only INTSTAT register provides a view of the status of the interrupt flags. This can simplify software handling of interrupts. See the STAT register for detailed descriptions of the interrupt flags. The interrupt status bit is set when it is enabled in INTENSET register and is set in the STAT register.

Diagram



Fields

Field	Function
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERRINT	Auto Baud Error Interrupt Flag
15 RXNOISEINT	Received Noise Interrupt Flag
14 PARITYERRINT	Parity Error Interrupt Flag
13 FRAMERRINT	Framing Error Interrupt Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 START	Start Detected on Receiver Flag This bit is set when a start is detected on the receiver input.
11 DELTARXBRK	Delta Receiver Break Change Flag This bit is set when a change in the state of receiver break detection occurs.
10-7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISINT	Transmitter Disabled Interrupt Flag
5 DELTACTS	Delta CTS Change Flag This bit is set when a change in the state of the CTS input is detected.
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLE	Transmitter Idle Flag
2-0 —	Reserved Read value is undefined; only zero should be written.

53.3.5.1.9 Oversample Selection Register for Asynchronous Communication (OSR)

Offset

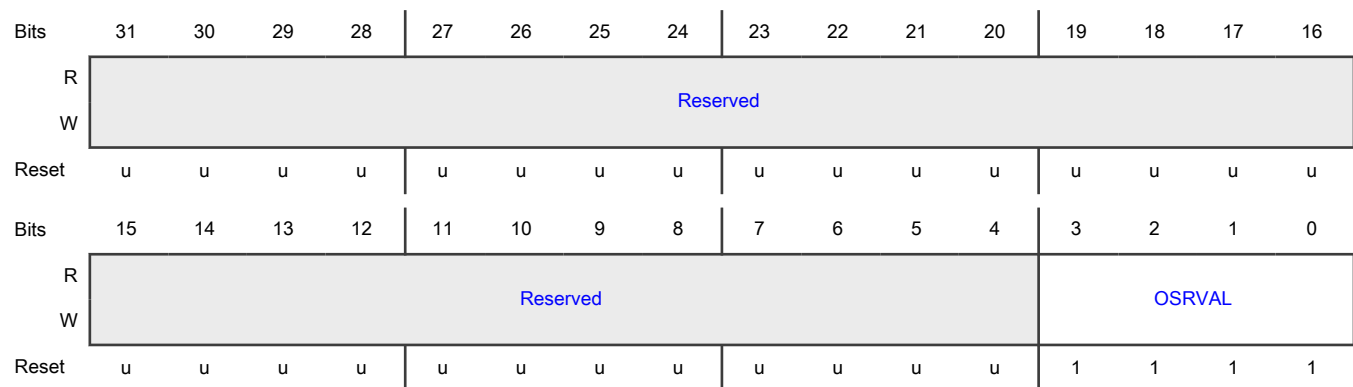
Register	Offset
OSR	28h

Function

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Diagram



Fields

Field	Function
31-4	Reserved
—	The value read from a reserved bit is not defined.
3-0 OSRVAL	Oversample Selection Value 0000b - Not supported 0001b - Not supported 0010b - Not supported 0011b - Not supported 0100b - 5 function clocks are used to transmit and receive each data bit. 0101b - 6 function clocks are used to transmit and receive each data bit. 0110b - 7 function clocks are used to transmit and receive each data bit. 0111b - 8 function clocks are used to transmit and receive each data bit. 1000b - 9 function clocks are used to transmit and receive each data bit. 1001b - 10 function clocks are used to transmit and receive each data bit. 1010b - 11 function clocks are used to transmit and receive each data bit. 1011b - 12 function clocks are used to transmit and receive each data bit. 1100b - 13 function clocks are used to transmit and receive each data bit. 1101b - 14 function clocks are used to transmit and receive each data bit. 1110b - 15 function clocks are used to transmit and receive each data bit. 1111b - 16 function clocks are used to transmit and receive each data bit.

53.3.5.1.10 Address Register for Automatic Address Matching (ADDR)

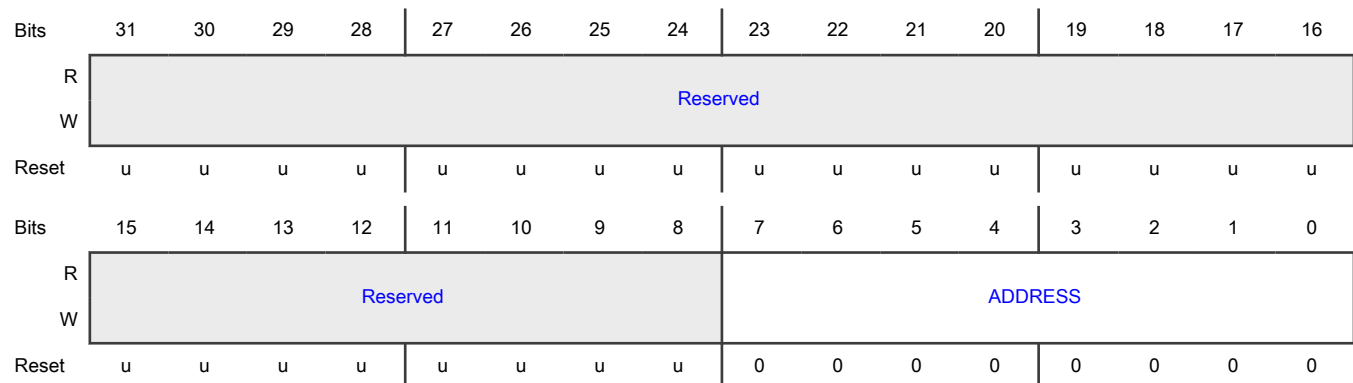
Offset

Register	Offset
ADDR	2Ch

Function

The ADDR register holds the value for hardware address matching.

Diagram



Fields

Field	Function
31-8	Reserved
—	The value read from a reserved bit is not defined.
7-0	Address
ADDRESS	The 8-bit value is used with automatic address matching. Used when address detection is enabled (CTL[ADDRDET] = 1) and automatic address matching is enabled (CFG[AUTOADDR] = 1).

53.3.5.1.11 FIFO Configuration (FIFOCFG)

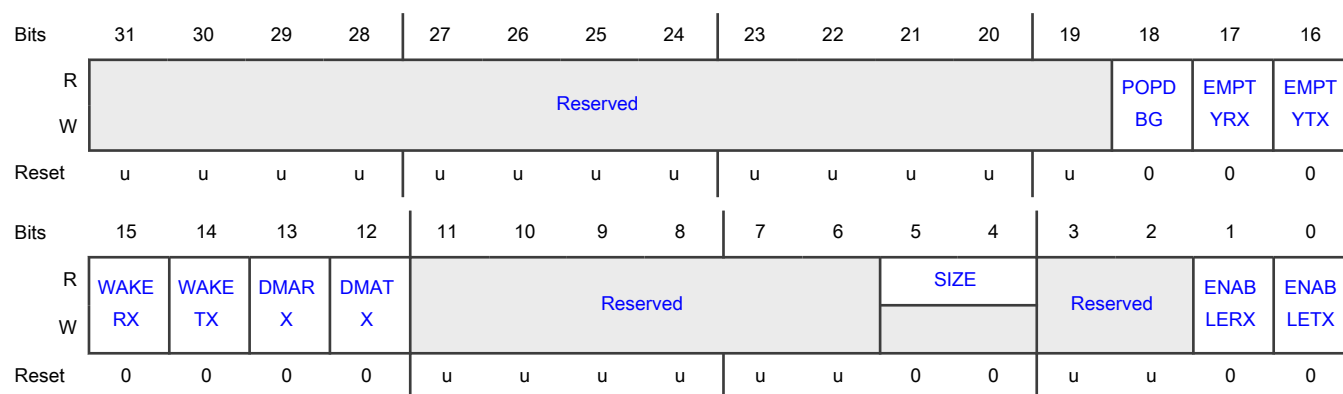
Offset

Register	Offset
FIFOCFG	E00h

Function

This register configures how the module uses the FIFO. Before configuring the FIFO, a peripheral function within the Flexcomm module must be selected.

Diagram



Fields

Field	Function
31-19 —	Reserved Read value is undefined; only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0b - Debug reads of the FIFO do not pop the FIFO. 1b - A debug read will cause the FIFO to pop.
17 EMPTYRX	Empty Command for the Receive FIFO When a 1 is written to EMPTYRX, the RX FIFO is emptied. 0b - No effect 1b - The RX FIFO is emptied.
16 EMPTYTX	Empty Command for the Transmit FIFO When a 1 is written to EMPTYTX, the TX FIFO is emptied. 0b - No effect 1b - The TX FIFO is emptied.
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from low power modes (as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. 0b - Only enabled interrupts will wake up the device from low power modes. 1b - A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14	Wake-up for Transmit FIFO Level

Table continues on the next page...

Table continued from the previous page...

Field	Function
WAKETX	WAKETX allows the device to be woken from low power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. 0b - Only enabled interrupts will wake up the device from low power modes. 1b - A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by FIFOTRIG[TXLVL], even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Configuration for Receive 0b - DMA is not used for the receive function. 1b - Triggers DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Configuration for Transmit 0b - DMA is not used for the transmit function. 1b - Triggers DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined; only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. The settings 0x1, 0x2, 0x3 are not applicable to USART. 00b - FIFO is configured as 16 entries of 8 bits. 01b - Not used 10b - Not used 11b - Not used
3-2 —	Reserved Read value is undefined; only zero should be written.
1 ENABLERX	Enable the Receive FIFO 0b - The receive FIFO is not enabled. 1b - The receive FIFO is enabled.
0 ENABLETX	Enable the Transmit FIFO. 0b - The transmit FIFO is not enabled. 1b - The transmit FIFO is enabled.

53.3.5.1.12 FIFO Status (FIFOSTAT)

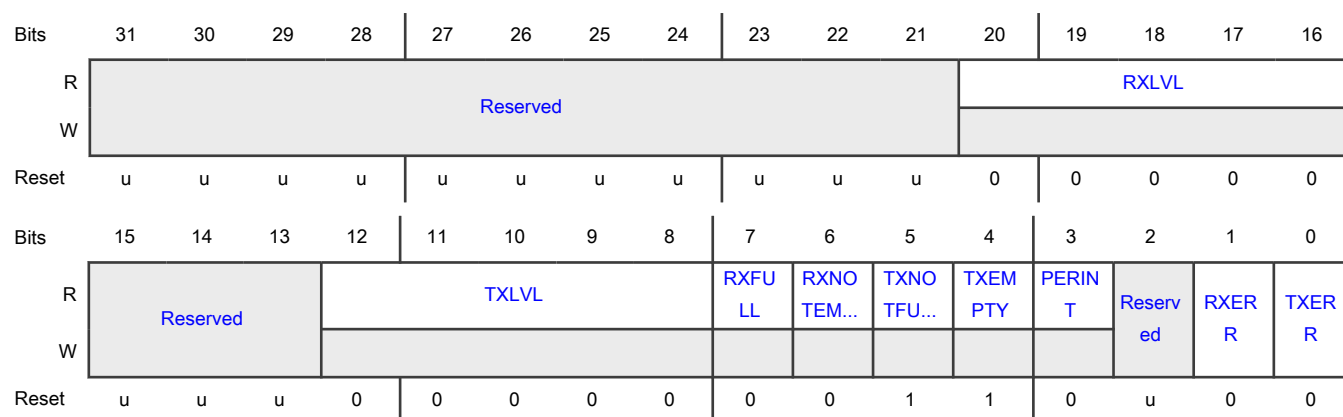
Offset

Register	Offset
FIFOSTAT	E04h

Function

Provides status information for the FIFO, and also indicates if a peripheral has issued an interrupt.

Diagram



Fields

Field	Function
31-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level RXLVL tells how much data is in the RX FIFO when the read occurs. RXLVL = 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags will be 1.
15-13 —	Reserved Read value is undefined; only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level TXLVL tells how much data is in the TX FIFO when the read occurs. TXLVL = 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags will be 0.
7 RXFULL	Receive FIFO is Full 0b - The receive FIFO is not full. The receive FIFO is not full.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - The receive FIFO is full. The receive FIFO is full. To prevent the peripheral from causing an overflow, data should be read out.
6 RXNOTEEMPTY	Receive FIFO is Not Empty 0b - The receive FIFO is empty. 1b - The receive FIFO is not empty, so data can be read.
5 TXNOTFULL	Transmit FIFO is Not Full 0b - The transmit FIFO is full and another write would cause it to overflow. 1b - The transmit FIFO is not full, so more data can be written.
4 TXEMPTY	Transmit FIFO Empty 0b - The transmit FIFO is not empty. 1b - The transmit FIFO is empty, although the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt 0b - No Peripheral Interrupt. The peripheral function has not asserted an interrupt. 1b - Peripheral Interrupt. Indicates that the peripheral function has asserted an interrupt. More information can be found by reading the peripheral's status register (STAT).
2 —	Reserved Read value is undefined; only zero should be written.
1 RXERR	RX FIFO Error Cleared by writing a 1 to itself. 0b - A receive FIFO overflow has not occurred 1b - A receive FIFO overflow has occurred, caused by software or DMA not emptying the FIFO fast enough
0 TXERR	TX FIFO Error Cleared by writing a 1 to itself. 0b - A transmit FIFO error has not occurred. 1b - A transmit FIFO error has occurred. This error could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed.

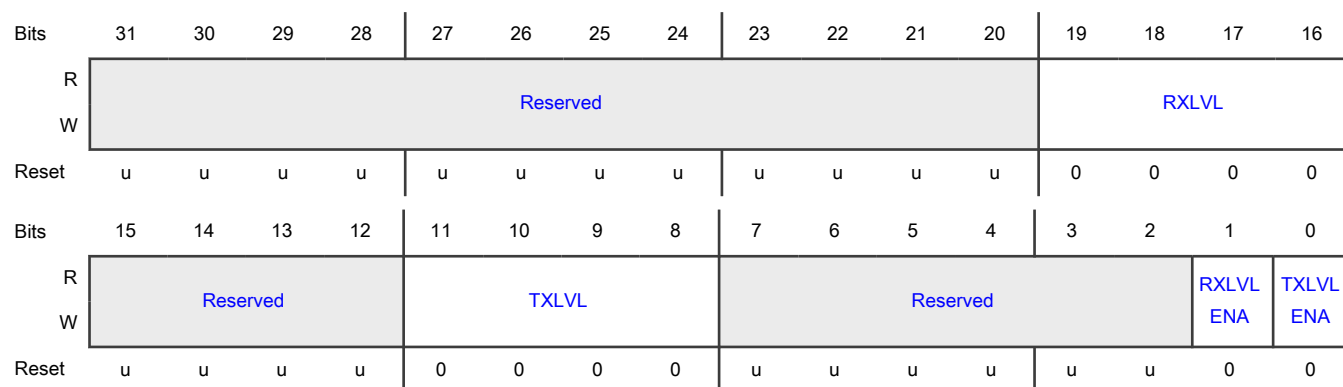
53.3.5.1.13 FIFO Trigger Settings for Interrupt and DMA Request (FIFOTRIG)

Offset

Register	Offset
FIFOTRIG	E08h

Function

Allows selecting when FIFO-level related interrupts occur, and includes FIFO trigger settings for interrupt and DMA requests.

Diagram**Fields**

Field	Function
31-20 —	Reserved Read value is undefined; only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the low power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. NOTE FIFOCFG[WAKERX] allows the device to be woken from reduced power modes. 0000b - Trigger when the RX FIFO has received 1 entry (is no longer empty) 0001b - Trigger when the RX FIFO has received 2 entries 1111b - Trigger when the RX FIFO has received 16 entries (has become full)
15-12 —	Reserved Read value is undefined; only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the low power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. NOTE FIFOCFG[WAKETX] allows the device to be woken from reduced power modes. 0000b - Trigger when the TX FIFO becomes empty

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0001b - Trigger when the TX FIFO level decreases to 1 entry 1111b - Trigger when the TX FIFO level decreases to 15 entries (is no longer full)
7-2 —	Reserved Read value is undefined; only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable RXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if FIFOCFG[DMARX] is set. 0b - Receive FIFO level does not generate a FIFO level trigger. 1b - An trigger will be generated if the receive FIFO level reaches the value specified by the RXLVL field.
0 TXLVLENA	Transmit FIFO Level Trigger Enable. TXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMATX in FIFOCFG is set. 0b - Transmit FIFO level does not generate a FIFO level trigger. 1b - A trigger will be generated if the transmit FIFO level reaches the value specified by the TXLVL field.

53.3.5.1.14 FIFO Interrupt Enable (FIFOINTENSET)

Offset

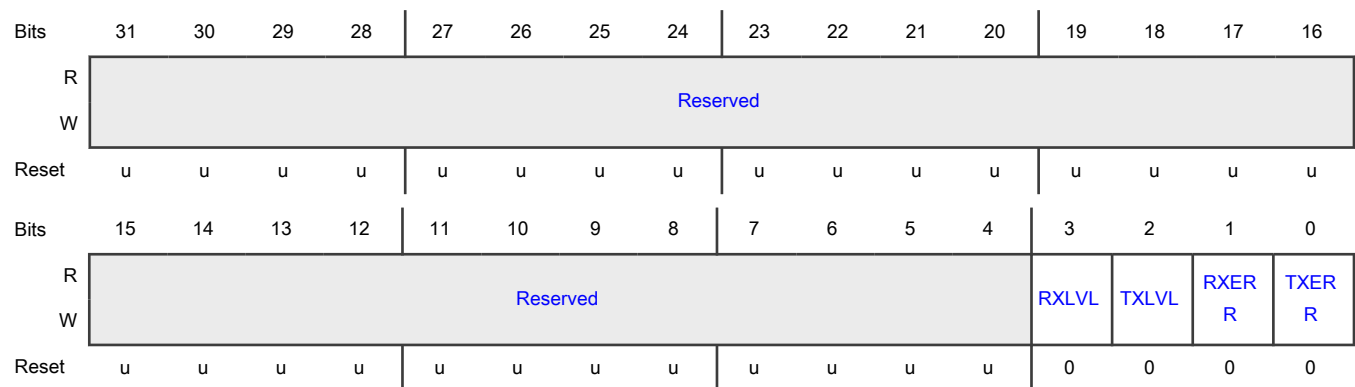
Register	Offset
FIFOINTENSET	E10h

Function

FIFOINTENSET includes the complete set of FIFO interrupt enables. Use the FIFO Interrupt Enable Register to enable various interrupt sources. Writing ones to implemented bits in FIFOINTENSET register forces those bits to be set.

Use the FIFOINTENCLR register to clear the interrupt enable bits in this register.

Diagram



Fields

Field	Function
31-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[RXLVL] field. 0b - No interrupt will be generated based on the RX FIFO level. 1b - If FIFOTRIG[RXLVLENA] = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL].
2 TXLVL	Transmit FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0b - No interrupt will be generated based on the TX FIFO level. 1b - If FIFOTRIG[TXLVLENA] = 1, then an interrupt will be generated when the TX FIFO level decreases to the level specified by FIFOTRIG[TXLVL]
1 RXERR	Receive Error Interrupt Enable Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag 0b - No interrupt will be generated for a receive error. 1b - An interrupt will be generated when a receive error occurs.
0 TXERR	Transmit Error Interrupt Enable Determines whether an interrupt occurs after a transmit error occurs, based on the FIFOSTAT[TXERR] flag 0b - No interrupt will be generated for a transmit error. 1b - An interrupt will be generated when a transmit error occurs.

53.3.5.1.15 FIFO Interrupt Enable Clear (FIFOINTENCLR)

Offset

Register	Offset
FIFOINTENCLR	E14h

Function

Use the FIFOINTENCLR register to clear interrupt enable bits in FIFOINTENSET. Writing 1 clears the corresponding bits in the FIFOINTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												RXLVL	TXLVL	RXERR	TXERR
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0

Fields

Field	Function
31-4	Reserved
—	Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[RXLVL] 0b - No effect 1b - Clear the interrupt
2 TXLVL	Transmit FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[TXLVL] 0b - No effect 1b - Clear the interrupt
1 RXERR	Receive Error Interrupt Enable Clear the interrupt 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Clear the interrupt
0 TXERR	Transmit Error Interrupt Enable Writing 1 clears FIFOINTENSET[TXERR] 0b - No effect 1b - Clear the interrupt

53.3.5.1.16 FIFO Interrupt Status (FIFOINTSTAT)

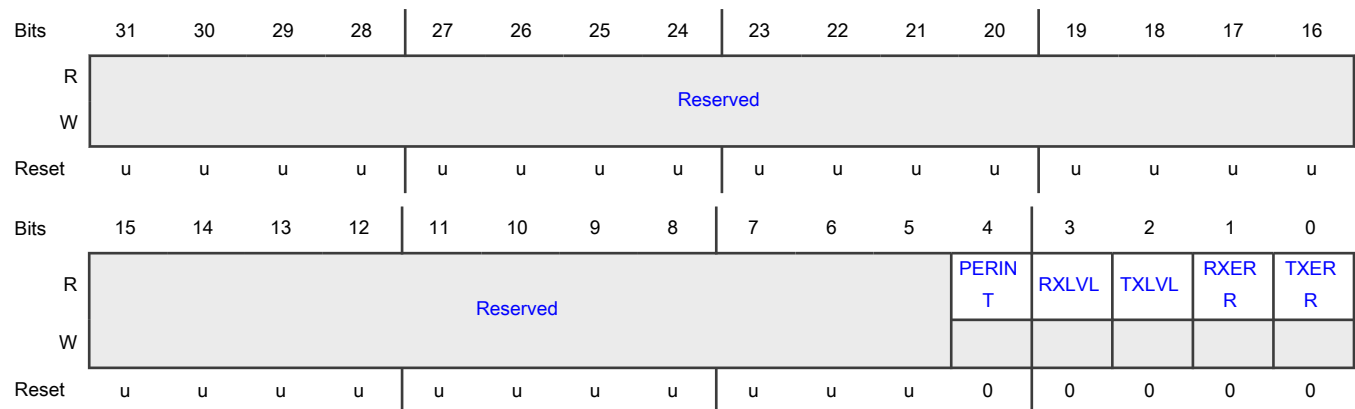
Offset

Register	Offset
FIFOINTSTAT	E18h

Function

The read-only FIFOINTSTAT register provides a view of the interrupt flags that are pending. This can simplify the software handling of interrupts.

Diagram



Fields

Field	Function
31-5 —	Reserved Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0b - Not pending

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0b - Not pending 1b - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0b - Not pending 1b - Pending
1 RXERR	RX FIFO Error Interrupt Status 0b - Not pending 1b - Pending
0 TXERR	TX FIFO Error Interrupt Status 0b - Not pending 1b - Pending

53.3.5.1.17 FIFO Write Data (FIFOWR)

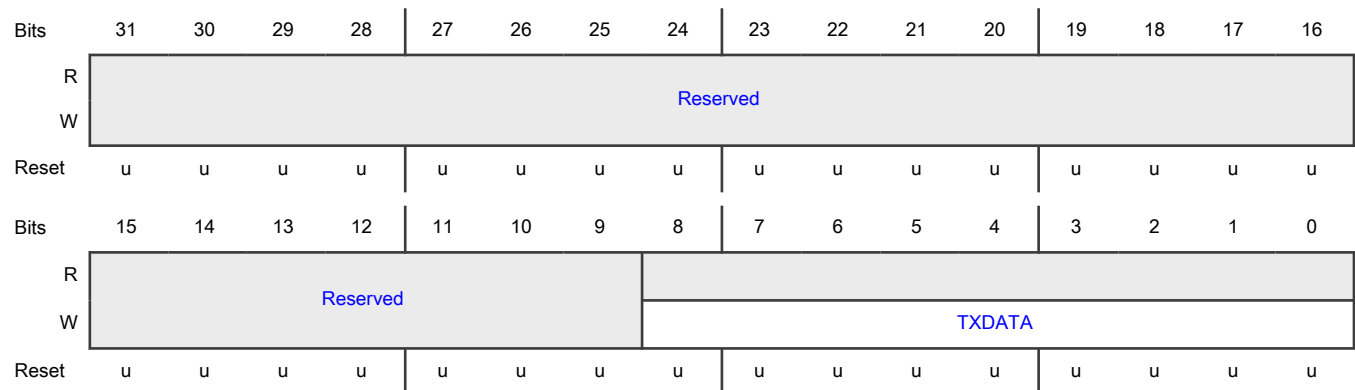
Offset

Register	Offset
FIFOWR	E20h

Function

The FIFOWR register is used to write values to be transmitted to the FIFO.

Diagram



Fields

Field	Function
31-9 —	Reserved Read value is undefined; only zero should be written.
8-0 TXDATA	Transmit data to the FIFO

53.3.5.1.18 FIFO Read Data (FIFORD)

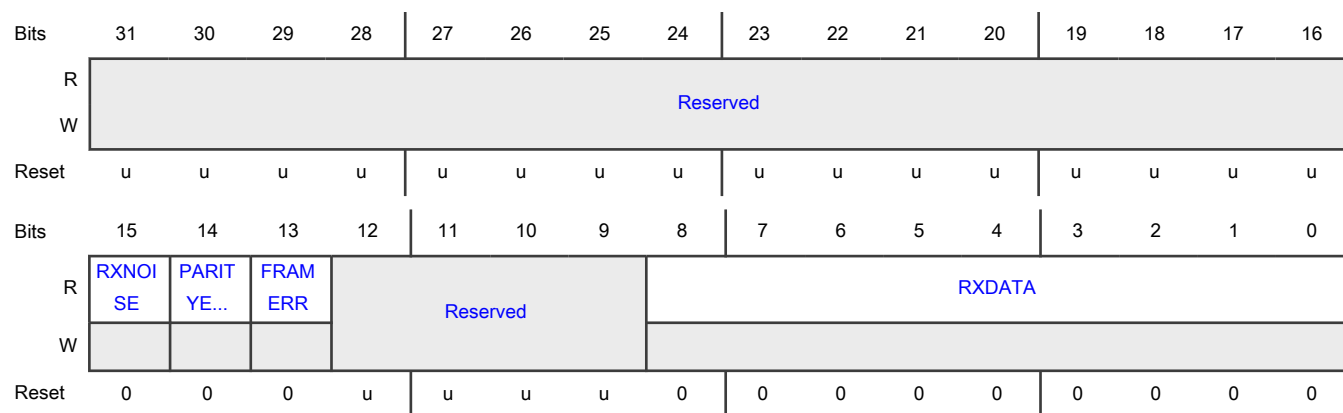
Offset

Register	Offset
FIFORD	E30h

Function

Use the FIFORD register to read values that have been received by the FIFO.

Diagram



Fields

Field	Function
31-16 —	Reserved The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag See description of STAT[RXNOISEINT].
14	Parity Error Status Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
PARITYERR	PARITYERR reflects the status for the data that is read from the FIFO. PARITYERR will be set when a parity error is detected in a received character.
13 FRAMERR	Framing Error Status Flag FRAMERR reflects the status for the data that is read from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	Reserved The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from the FIFO The number of bits used depends on the CFG[DATALEN] and CFG[PARITYSEL] settings.

53.3.5.1.19 FIFO Data Read with No FIFO Pop (FIFORDNOPOP)

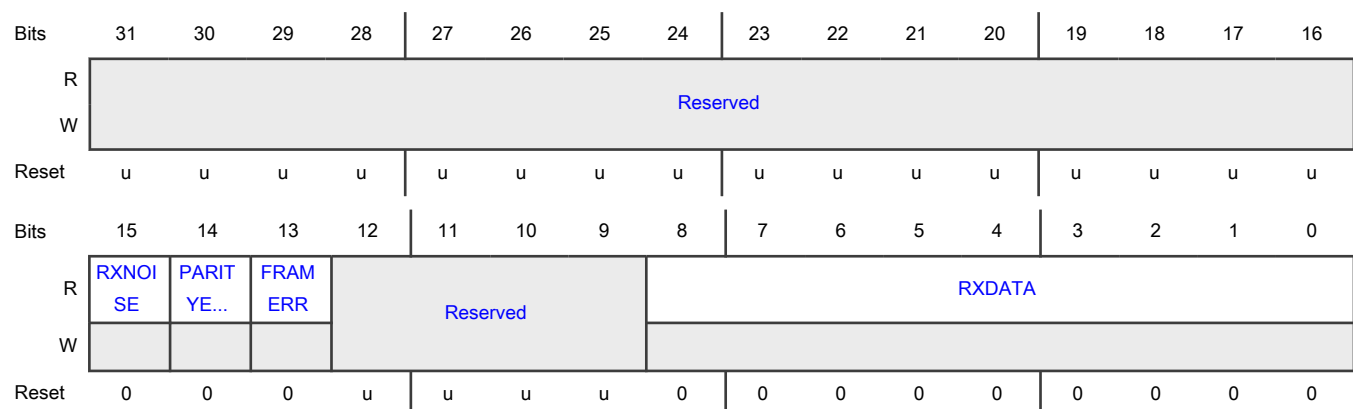
Offset

Register	Offset
FIFORDNOPOP	E40h

Function

The FIFORDNOPOP register acts in exactly the same way as the FIFORD register, except that FIFORDNOPOP supplies data from the top of the FIFO without popping the FIFO (thereby leaving the FIFO state unchanged). This can be used to allow the system software to observe incoming data without interfering with the peripheral driver.

Diagram



Fields

Field	Function
31-16 —	Reserved The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag See description of STAT[RXNOISEINT].
14 PARITYERR	Parity Error Status Flag PARITYERR reflects the status for the data that is read from the FIFO. PARITYERR will be set when a parity error is detected in a received character.
13 FRAMERR	Framing Error Status Flag FRAMERR reflects the status for the data that is read from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	Reserved The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from the FIFO The number of bits used depends on the CFG[DATALEN] and CFG[PARITYSEL] settings.

53.3.5.1.20 FIFO Size (FIFOSIZE)

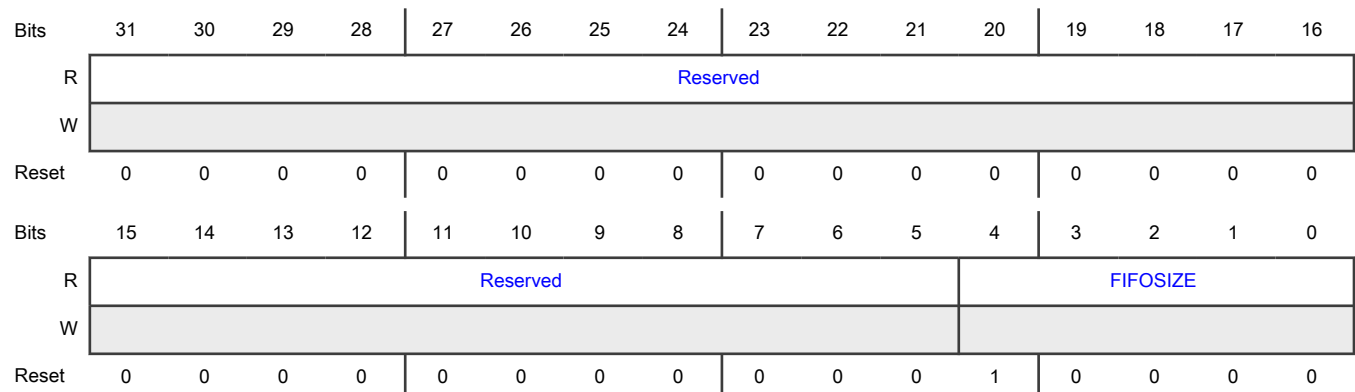
Offset

Register	Offset
FIFOSIZE	E48h

Function

Provides the FIFO size.

Diagram



Fields

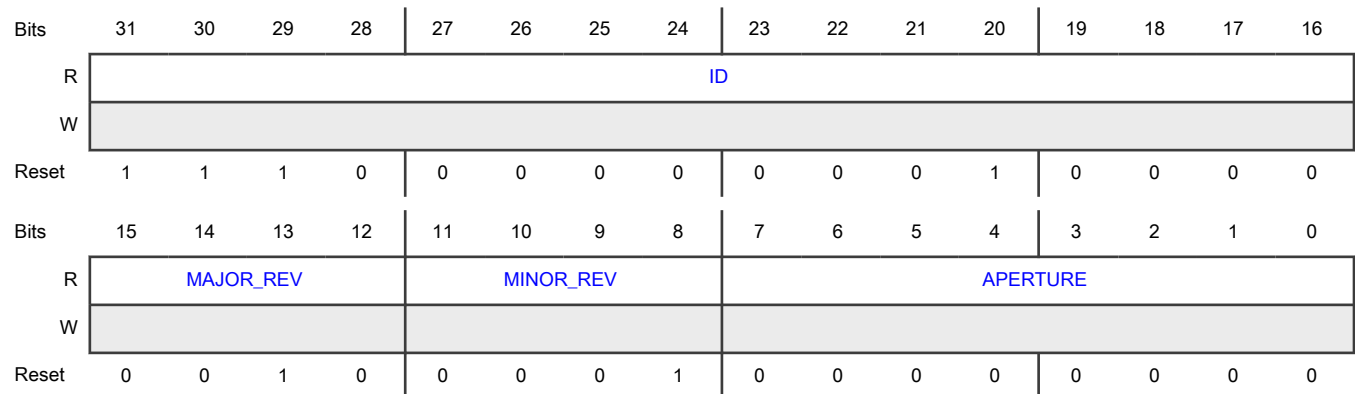
Field	Function
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 16 entries for this chip. This field is read only.

53.3.5.1.21 Peripheral Identification (ID)**Offset**

Register	Offset
ID	FFCh

Function

Identifies the type and revision of the module. A generic software driver can make use of this information to implement specific behavior, based on a module type or a specific revision of that module.

Diagram**Fields**

Field	Function
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8	Minor revision of module implementation

Table continues on the next page...

Table continued from the previous page...

Field	Function
MINOR_REV	
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

53.4 Serial Peripheral Interfaces (SPI)

53.4.1 Overview

The SPI is a Serial Peripheral Interface (SPI) module that supports an efficient interface to an SPI bus, either as a master and/or as a slave.

- The SPI is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The SPI can continue operating in stop modes, if an appropriate clock is available.
- The SPI supports DMA accesses and generates a DMA request.

The Serial Peripheral Interface bus (SPI) is a synchronous serial communication interface used in embedded systems, typically to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing to Secure Digital cards and LCD displays.

- SPI devices communicate in full duplex mode using a master-slave scheme with a single master. This enables communication in both directions to happen simultaneously.
- Multiple slave devices are selected using individual slave select (SS) lines.
- The master device originates the frame for reading and writing.
- SPI accesses are always synchronous to the external clock.
- Each SPI can only assert one chip select at a time, but technically there can be multiple SPI slaves sharing the same chip select (in the form of a larger shift register). This requires:
 - the SDO of the master to connect to the SDI of the first slave,
 - the SDO of first slave to connect to the SDI of the next slave,
 - and the SDO of last slave to connect to the SDI of the SPI master.

Note that some SPI modules use MISO/MOSI for the input/output data.

53.4.1.1 Block diagram

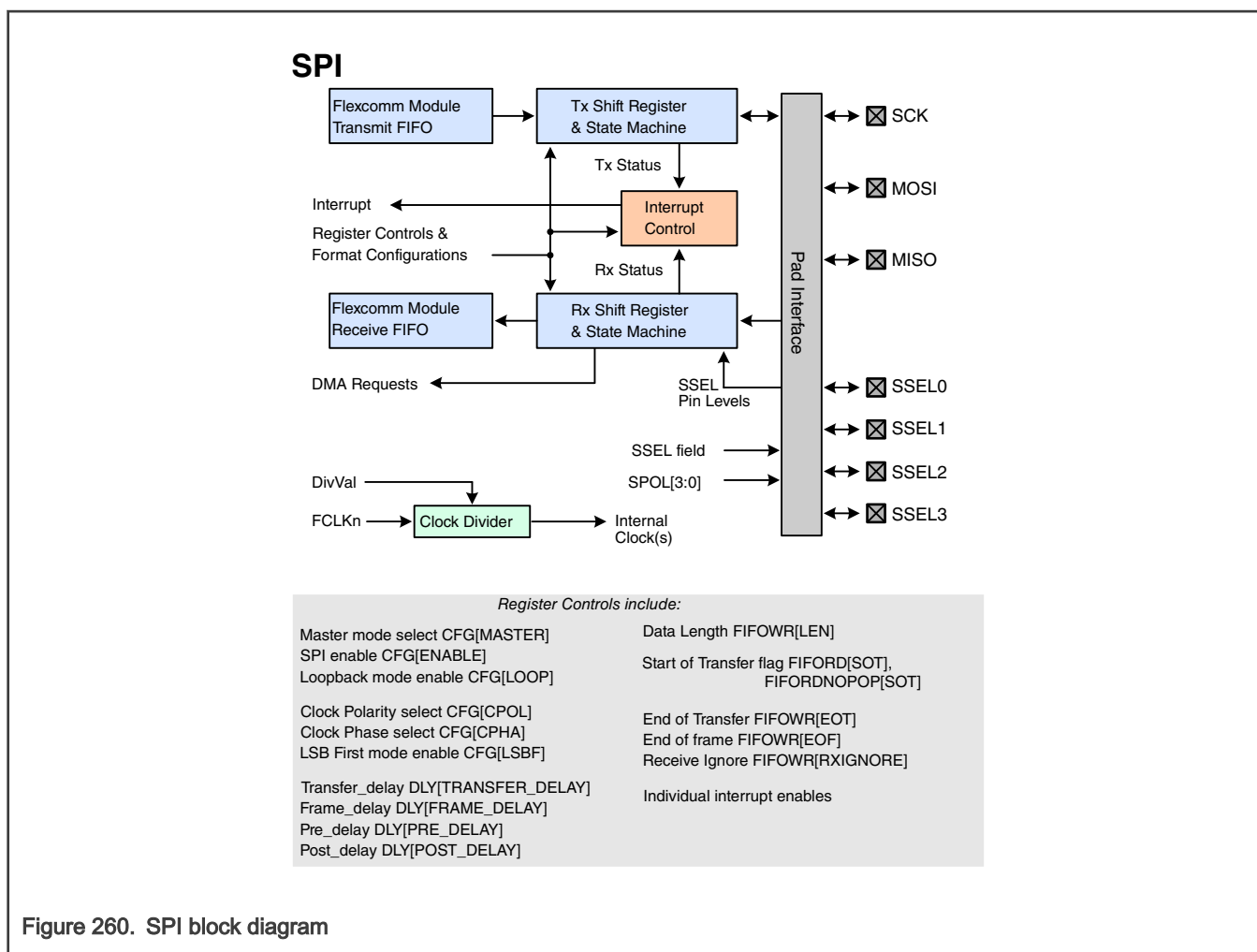


Figure 260. SPI block diagram

53.4.1.2 Features

- Master and slave operations
- Data transmits of 4 to 16 bits supported directly; larger frames are supported by software
- The SPI function supports separate transmit and receive FIFOs with 8 entries each
- Supports DMA transfers: SPIn transmit and receive functions can be operated with the system DMA controller
- Data can be transmitted to a slave without the need to read incoming data, which can be useful while setting up an SPI memory
- Up to 4 Slave Select input/outputs with selectable polarity and flexible usage

NOTE

Texas Instruments SSI and National Microwire modes are not supported.

53.4.2 Functional description

53.4.2.1 Operating modes

This section describes the operating modes of the SPI module.

53.4.2.1.1 Slave select

The SPI block provides for 4 Slave Select (SSEL n) inputs in slave mode or 4 Slave Select outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, then this signal inversion is performed as the signal leaves or enters the SPI block.

In Slave mode:

- Any asserted SSEL that is connected to a pin will activate the SPI.
- All 4 SSEL states and the received data are saved in the FIFO Read Data Register (FIFORD).

In Master mode:

- All SSELs that are connected to a pin will be output as defined in the SPI registers. The SSELs could potentially be decoded externally, to address more than 4 slave devices. Note that at least 1 SSEL is asserted when data is transferred in master mode.
- Slave Selects come from the TXSSEL n _N bits in the FIFO Write Data Register (FIFOWR).

53.4.2.1.2 DMA operation

A DMA request is provided for each SPI direction, and can be used instead of interrupts for transferring data (by configuring the DMA controller, FIFO registers, and INPUTMUX registers appropriately). The DMA controller provides an acknowledgement signal that clears the related request when it (the DMA controller) completes handling that request.

- The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.
- The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

53.4.2.1.2.1 DMA Master Mode End-Of-Transfer

When using polled or interrupt methods to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The End of Transfer bit (FIFOWR[EOT]) bit would be set just before the last data to be sent or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (the drive SSEL is inactive) can be generated in 4 ways:

- **Using DMA interrupt and a 2nd DMA transfer:**

To use only 8 or 16 bit wide DMA transfers for all the data, a 2nd DMA transfer can be used to terminate the transfer (drive SSEL inactive). The transfer would be started by configuring the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set the End of Transfer bit (FIFOWR[EOT]) and then configure DMA to send the last data.

- **Using DMA and SPI interrupts (or background SPI status polling):**

To use only one 8-bit or 16-bit wide DMA transfer for all the data, 2 interrupts will be required to properly terminate the transfer (drive SSEL inactive). The SPI Tx DMA completion interrupt function sets the Transmit FIFO Level Trigger Point (FIFOTRIG[TXLVL]) to 0, and also sets the Transmit FIFO Level Interrupt Enable interrupt enable (FIFOINTENSET[TXLVL]). To force termination after all data output has finished, the interrupt function handling the SPI TXLVL would set the the End Transfer bit (STAT[ENDTRANSFER]).

- **Using DMA linked descriptor:**

The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8-bit or 16-bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the End of Transfer bit (FIFOWR[EOT]).

NOTE

The DMA interrupt function cannot set the control end transfer bit (STAT[ENDTRANSFER]), because this may terminate the transfer while the FIFO still has data to send.

- **Using 32-bit wide DMA:**

Write both data and control bits to the FIFO Write Data Register (FIFOWR) for all data. The control bits for the last entry would include the setting of the End of Transfer bit (FIFOWR[EOT]). This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

53.4.2.1.3 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits, or 2 groups of 12 bits, among others. Frames of any size, including frames greater than 32 bits, can supported in the same way.

How to handle larger data widths depends partially on other SPI configuration options. For example, if it is intended for Slave Selects to be deasserted between frames, then the deassertion of Slave Selects must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for example, would require changing the value of the End of Frame bit (FIFOWR[EOF]) on alternate 12-bit frames.

53.4.2.1.4 Data stalls

Types of data stalls include:

- **Transmitter stall:**

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, if the next piece of data is not yet available, then the stall happens just before the final clock edge of data.

- **Receiver stall:**

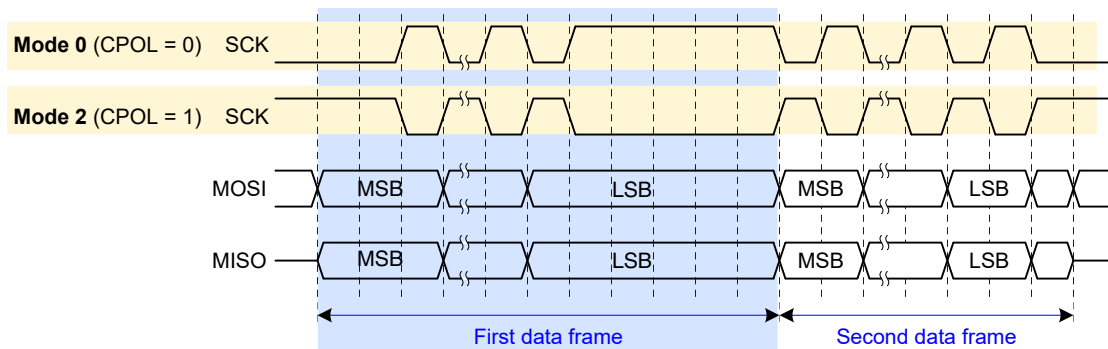
A stall for Master receive can happen when a FIFO overflow would otherwise occur (see RX FIFO Error Interrupt Status bit FIFOSTAT[RXERR]), if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This receiver stall happens 1 clock edge earlier than the transmitter stall.

- **Receiver stall:**

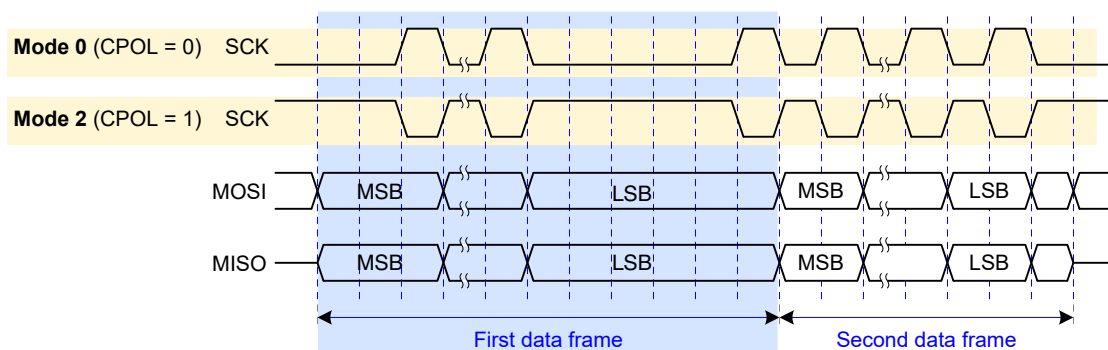
In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3, because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected by the Stalled status flag (STAT[STALLED]), which indicates the current SPI status. The transmitter will be stalled until the data is read from the receive FIFO. To avoid having to read the received data, use the Receive Ignore control bit (FIFOWR[RXIGNORE]=1).

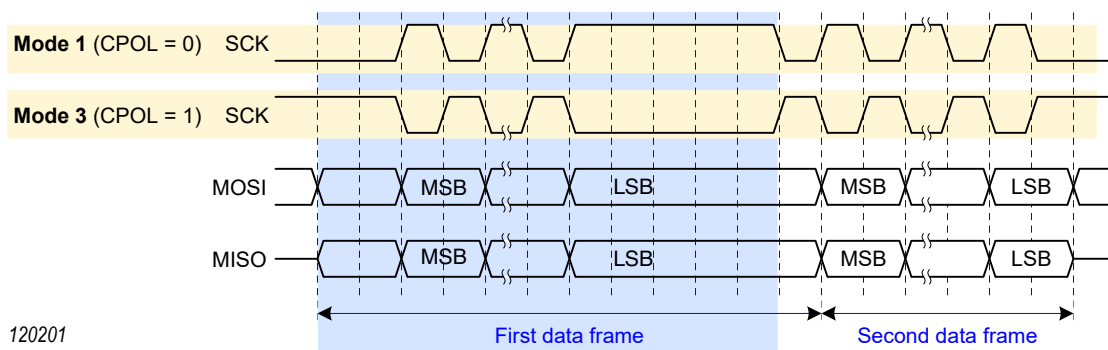
Transmitter stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall



Receiver stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall



Receiver stall: CPHA = 1, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall



120201

Figure 261. Data stall examples

53.4.2.2 Operations

This section describes the SPI module's operations.

53.4.2.2.1 Frame delays

Several types of delays can be specified for SPI frames:

- Pre-delay: a delay after SSEL is asserted before data clocking begins (DLY[PRE_DELAY])

- Post-delay: a delay at the end of a data frame before SSEL is deasserted (DLY[POST_DELAY])
- Frame delay: a delay between data frames when SSEL is not deasserted (DLY[FRAME_DELAY])
- Transfer delay: a minimum duration of SSEL in the deasserted state between transfers (DLY[TRANSFER_DELAY])

53.4.2.2.1.1 Pre-delay and Post-delay

The pre-delay value (DLY[PRE_DELAY]) controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The post-delay value (DLY[POST_DELAY]) controls the amount of time between the end of a data frame and the deassertion of the slave select line (SSEL).

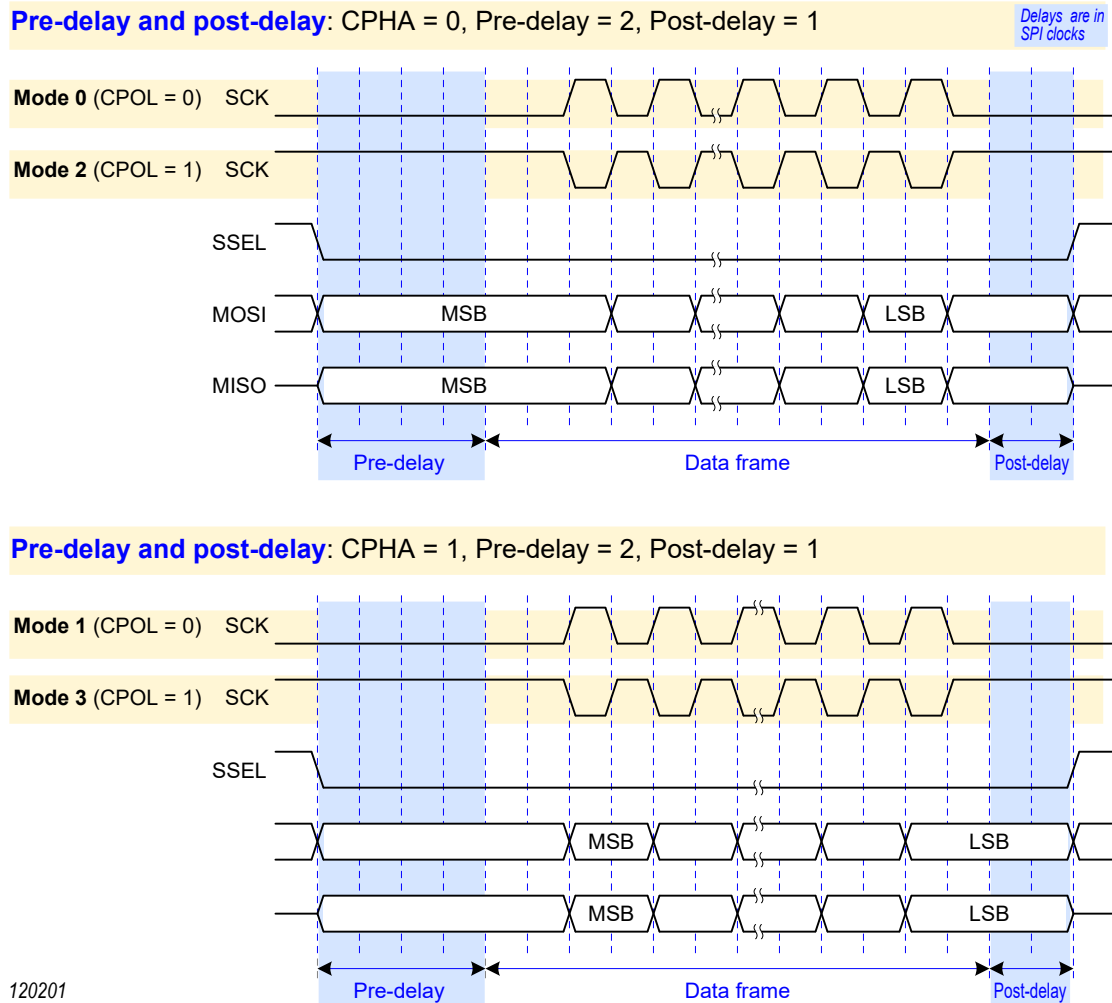


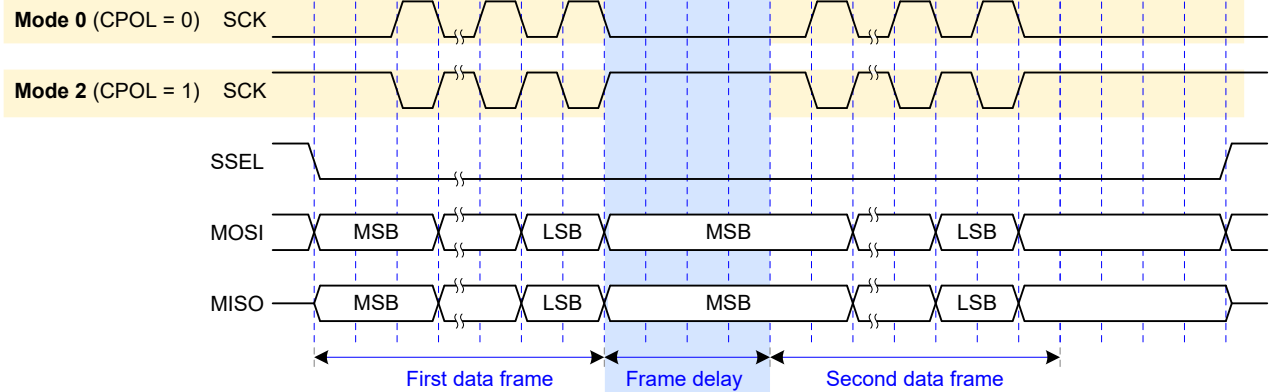
Figure 262. Pre-delay and post-delay

53.4.2.2.1.2 Frame delay

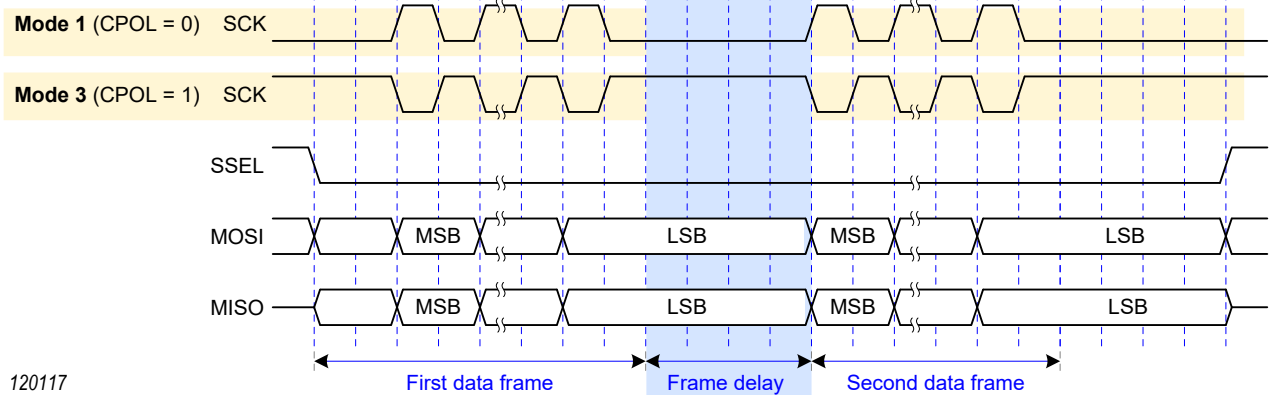
The frame delay value controls the amount of time at the end of each frame. The frame delay value is inserted when the End of frame bit (FIFOWR[EOF]) = 1. Note that frame boundaries occur only where specified, because frame lengths can be any size, involving multiple data writes.

Frame delay: CPHA = 0, Frame delay = 2, Pre-delay = 0, Post-delay = 0

Delays are in
SPI clocks



Frame delay: CPHA = 1, Frame delay = 2, Pre-delay = 0, Post-delay = 0

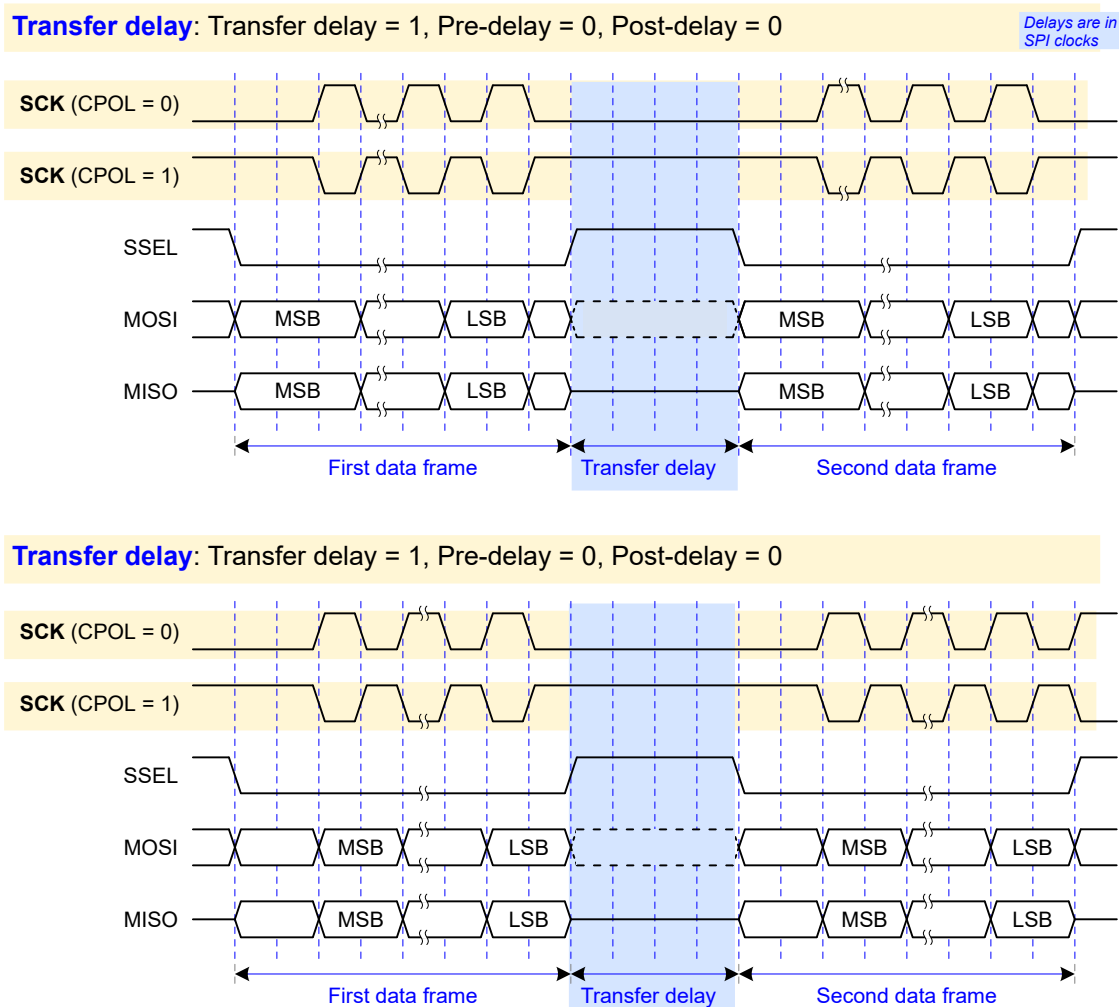


120117

Figure 263. Frame delay

53.4.2.2.1.3 Transfer delay

The transfer delay (DLY[TRANSFER_DELAY]) controls the minimum amount of time that a slave select line (SSEL0) is deasserted between transfers, because the End of Transfer bit = 1 (FIFOWR[EOT]). When the transfer delay = 0 (DLY[TRANSFER_DELAY]), the slave select line (SSEL) may be deasserted for a minimum of 1 SPI clock time.



120201

Figure 264. Transfer delay

53.4.2.3 Clocks

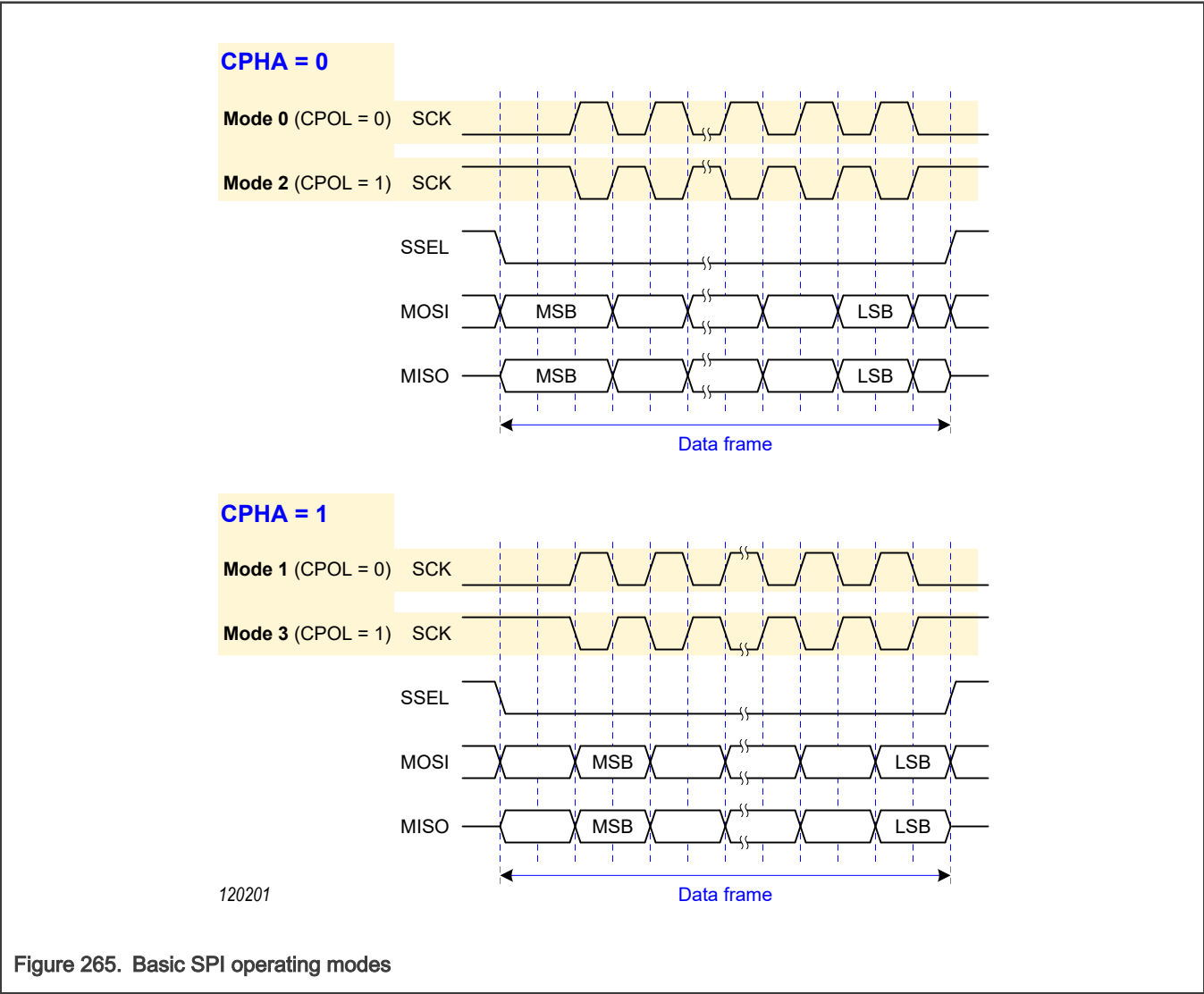
This section describes clocks and special clocking requirements of the SPI module.

53.4.2.3.1 Operating modes: clock and phase selection

SPI interfaces typically allow the clock phase and polarity to be configured. These clock configurations are sometimes referred to as numbered SPI modes.

Table 557. SPI mode summary

Clock Polarity Select CPOL	Clock Phase Select CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0, but with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1, but with SCK inverted.	high	falling	rising



53.4.2.3.2 Clocking and data rates

Before using the SPI, the clocking must be defined, which includes configuring the system clock and selecting the clock divider value (DIV[DIVVAL]).

53.4.2.3.2.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

- In master mode: the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.
- In slave mode: the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

The SPI clock divider is an integer divider.

- In master mode: the SPI can be set to run at the same speed as the selected FCLK, or at lower integer divide rates. The SPI rate will be = (FCLK of Flexcomm module) / DIVVAL.
- In slave mode: the clock is taken from the SCK input and the SPI clock divider is not used.

53.4.3 Signals

The SPI signals are movable Flexcomm module functions and are assigned to external pins via the SoC's IO controller.

Table 558. SPI signals

Signal	I/O	Description
SCK	I/O	Serial Clock for SPI on Flexcomm module • SCK is a clock signal used to synchronize the transfer of data. • SCK is driven by the master and received by the slave. SCK is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit. • When the SPI interface is used, the clock is programmable to be active-high or active-low. • SCK only switches during a data transfer.
MOSI	I/O	Master-Out Slave-In for SPI on Flexcomm module • The MOSI signal transfers serial data from the master to the slave. • When the SPI is a master, the SPI outputs serial data on this signal. • When the SPI is a slave, the SPI clocks in serial data from this signal. • MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MISO	I/O	Master In Slave Out for SPI on Flexcomm module • The MISO signal transfers serial data from the slave to the master. • When the SPI is a master, serial data is input from this signal. • When the SPI is a slave, serial data is output to this signal. • MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	Slave Select 0 for SPI on Flexcomm module

Table continues on the next page...

Table 558. SPI signals (continued)

Signal	I/O	Description
		- When the SPI interface is a master, the SPI will drive the SSEL signals to an active state before the start of serial data, and then release them (the SSEL signals) to an inactive state after the serial data has been sent. By default, this slave select signal is active low, but can be selected to operate as active high. - When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
SSEL1	I/O	Slave Select 1 for SPI on Flexcomm module
SSEL2	I/O	Slave Select 2 for SPI on Flexcomm module
SSEL3	I/O	Slave Select 3 for SPI on Flexcomm module

53.4.4 Initialization

To initially configure an SPI module:

- Reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the desired Flexcomm module function.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
 - Enable the clock to the SPI register interface.
 - Enable or disable the related Flexcomm interrupts in the SoC.
 - Configure the required Flexcomm pin functions in the SoC.
 - Configure the Flexcomm clock (FCLK) and SPI data rate.

NOTE

The Flexcomm clock frequency should not be greater than 48 MHz.

- Set the Receive Ignore control bit (FIFOWR[RXIGNORE] bit) to only transmit data and not read the incoming data. Otherwise, the transmit will halt when the FIFORD buffer is full.
- Configure the SPI function to wake up the part from low power modes.

53.4.4.1 Configure the SPI for wake-up

- In sleep mode:**, any signal that triggers an SPI interrupt can wake up the device, if that the interrupt is enabled in the INTENSE register and in the SoC interrupt controller. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master mode or slave mode.
- In deep-sleep mode:** the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, then the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the SoC interrupt controller.

53.4.4.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode.
- Enable the SPI interrupt in the SoC interrupt controller.

- Any enabled SPI interrupt will now wake up the device from sleep mode.

53.4.4.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. The SCK function must be connected to a pin and the pin must be connected to the SPI master.
- Enable the SPI interrupt.
- Enable the SPI interrupt in the SoC interrupt controller.
- Enable desired SPI interrupts. For examples, wake-up events can be:
 - A change in the state of the SSEL pins
 - Data available to be received
 - Receive FIFO overflow

Wake-up for DMA only

The device can optionally be woken up only enough to perform the needed DMA operations before returning to deep-sleep mode, without the CPU waking up at all. To do this:

- Set up the appropriate SPI function to use DMA, and set the related WAKE bit (FIFOCFG[WAKETX] for the transmit function; FIFOCFG[WAKERX] for the receive function).
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related SPI interrupt in the SoC interrupt controller.
- Enable the DMA interrupt in the SoC interrupt controller.
- Enable HWWAKE[FCWAKE] and HWWAKE[WAKEDMA].

53.4.5 Memory map and register definition

This section includes the SPI module memory map and detailed descriptions of all registers.

NOTE

Except for the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm module support only word writes. Byte and halfword writes are not supported in the SPI function for those registers. The FIFOWR register also supports byte and halfword (data only) writes, to allow writing FIFO data without affecting the SPI control fields above bit 15.

53.4.5.1 Serial Peripheral Interfaces (SPI) register descriptions

53.4.5.1.1 SPI memory map

FLEXCOMM0.SPI base address: 4010_6000h

FLEXCOMM1.SPI base address: 4010_7000h

FLEXCOMM2.SPI base address: 4010_8000h

FLEXCOMM3.SPI base address: 4010_9000h

FLEXCOMM4.SPI base address: 4012_2000h

FLEXCOMM5.SPI base address: 4012_3000h

FLEXCOMM6.SPI base address: 4012_4000h

FLEXCOMM7.SPI base address: 4012_5000h

FLEXCOMM8.SPI base address: 4020_9000h

FLEXCOMM9.SPI base address: 4020_A000h

FLEXCOMM10.SPI base address: 4020_B000h

FLEXCOMM11.SPI base address: 4020_C000h

FLEXCOMM12.SPI base address: 4020_D000h

FLEXCOMM13.SPI base address: 4020_E000h

FLEXCOMM14.SPI base address: 4012_6000h

FLEXCOMM16.SPI base address: 4012_8000h

Offset	Register	Width (In bits)	Access	Reset value
400h	Configuration Register (CFG)	32	RW	0000_0000h
404h	Delay Register (DLY)	32	RW	0000_0000h
408h	Status Register (STAT)	32	RW	0000_0100h
40Ch	Interrupt Enable Register (INTENSET)	32	RW	0000_0000h
410h	Interrupt Enable Clear Register (INTENCLR)	32	RW	See section
424h	Clock Divider Register (DIV)	32	RW	0000_0000h
428h	Interrupt Status Register (INTSTAT)	32	R	0000_0000h
E00h	FIFO Configuration Register (FIFOCFG)	32	RW	See section
E04h	FIFO Status Register (FIFOSTAT)	32	RW	0000_0030h
E08h	FIFO Trigger Register (FIFOTRIG)	32	RW	0000_0000h
E10h	FIFO Interrupt Enable Register (FIFOINTENSET)	32	RW	0000_0000h
E14h	FIFO Interrupt Enable Clear Register (FIFOINTENCLR)	32	RW	0000_0000h
E18h	FIFO Interrupt Status Register (FIFOINTSTAT)	32	R	0000_0000h
E20h	FIFO Write Data Register (FIFOWR)	32	RW	See section
E30h	FIFO Read Data Register (FIFORD)	32	R	See section
E40h	FIFO Data Read with no FIFO Pop Register (FIFORDNOPOP)	32	R	See section
E48h	FIFO Size Register (FIFOSIZE)	32	R	See section
FFCh	Peripheral Identification Register (ID)	32	R	E020_1200h

53.4.5.1.2 Configuration Register (CFG)

Offset

Register	Offset
CFG	400h

Function

The CFG register is used to initially configure the SPI, and typically the SPI configuration is not changed during normal operations. A setup sequence is recommended for initial SPI setup (after the SPI function has been selected), and for when changes must be made to settings in the CFG register after the SPI interface has been in use.

SPI Setup Sequence: To initialize the SPI interface or to change the existing SPI configuration when the SPI interface is operating:

- Disable the FIFO, by clearing the FIFOCFG[ENABLETX] and FIFOCFG[ENABLERX] bits
- Set up the SPI interface in the CFG register, leaving CFG[ENABLE] = 0
- Enable the FIFO, by setting the FIFOCFG[ENABLETX] and/or FIFOCFG[ENABLERX] bits
- Enable the SPI, by setting the CFG[ENABLE] bit

Also see the Master Idle Status flag (STAT[MSTIDLE]).

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				SPOL	SPOL	SPOL	SPOL	LOOP	Reserv	CPOL	CPHA	LSBF	MAST	Reserv	ENAB
W					3	2	1	0		ed				ER	ed	LE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-12	Reserved
—	Read value is undefined; only zero should be written.
11 SPOL3	SSEL3 Polarity Select 0b - Low. The SSEL3 pin is active low. 1b - High. The SSEL3 pin is active high.
10 SPOL2	SSEL2 Polarity Select 0b - Low. The SSEL2 pin is active low. 1b - High. The SSEL2 pin is active high.
9 SPOL1	SSEL1 Polarity Select 0b - Low. The SSEL1 pin is active low. 1b - High. The SSEL1 pin is active high.
8 SPOLO	SSEL0 Polarity Select

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Low. The SSEL0 pin is active low. 1b - High. The SSEL0 pin is active high.
7 LOOP	Loopback Mode Enable Loopback mode applies only to Master mode, and has transmit and receive data connected together, to enable simple software testing. 0b - Disabled 1b - Enabled
6 —	Reserved Read value is undefined; only zero should be written.
5 CPOL	Clock Polarity Select 0b - Low. The rest state of the clock (between transfers) is low. 1b - High. The rest state of the clock (between transfers) is high.
4 CPHA	Clock Phase Select 0b - Change. Change. The SPI captures serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state); data is changed on the following edge. 1b - Capture. Capture. The SPI changes serial data on the 1st clock data is captured on the following edge.
3 LSBF	LSB First Mode Enable 0b - Standard. Data is transmitted and received in standard MSB-first order. 1b - Reverse. Data is transmitted and received in reverse order (LSB first).
2 MASTER	Master Mode Select 0b - Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs; MISO is an output. 1b - Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs; MISO is an input.
1 —	Reserved Read value is undefined; only zero should be written.
0 ENABLE	SPI Enable 0b - Disabled. The SPI is disabled and the internal state machine and counters are reset. 1b - Enabled. The SPI is enabled for operation.

53.4.5.1.3 Delay Register (DLY)

Offset

Register	Offset
DLY	404h

Function

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in the number of SPI clocks. Timing details are shown in the PRE_DELAY, POST_DELAY, FRAME_DELAY, and TRANSFER_DELAY fields.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRANSFER_DELAY				FRAME_DELAY				POST_DELAY				PRE_DELAY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved Read value is undefined; only zero should be written.
15-12 TRANSFER_DELAY	Transfer Delay Controls the minimum amount of time that the SSEL is deasserted between transfers. 0000b - The minimum time that SSEL is deasserted is 1 SPI clock time (zero-added time) 0001b - The minimum time that SSEL is deasserted is 2 SPI clock times 0010b - The minimum time that SSEL is deasserted is 3 SPI clock times 1111b - The minimum time that SSEL is deasserted is 16 SPI clock times
11-8 FRAME_DELAY	Frame Delay - If the End of Frame bit (FIFOWR[EOF]) flag is set, then FRAME_DELAY controls the minimum amount of time between the current frame and the next frame. - If the End of Transfer bit (FIFOWR[EOT]) is set, then FRAME_DELAY controls the minimum amount of time between the current frame and SSEL deassertion.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - No additional time is inserted 0001b - 1 SPI clock time is inserted 0010b - 2 SPI clock times are inserted 1111b - 15 SPI clock times are inserted
7-4 POST_DELAY	Post-Delay Controls the amount of time between the end of a data transfer and SSEL deassertion. 0000b - No additional time is inserted 0001b - 1 SPI clock time is inserted 0010b - 2 SPI clock times are inserted 1111b - 15 SPI clock times are inserted
3-0 PRE_DELAY	Pre-Delay Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always 1 SPI clock time between SSEL assertion and the 1st clock edge, and this is not considered part of the pre-delay time. 0000b - No additional time is inserted 0001b - 1 SPI clock time is inserted 0010b - 2 SPI clock times are inserted 1111b - 15 SPI clock times are inserted

53.4.5.1.4 Status Register (STAT)

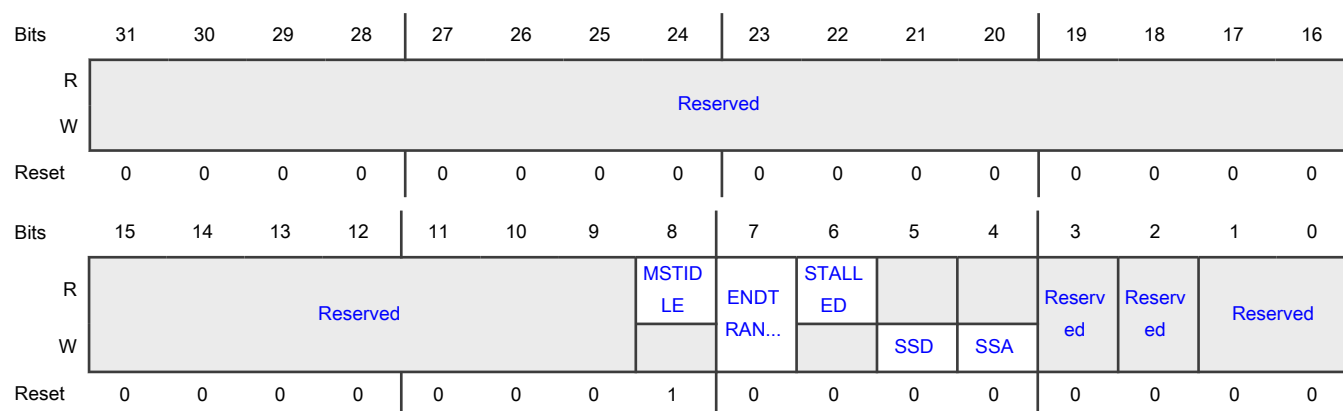
Offset

Register	Offset
STAT	408h

Function

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits in the STAT register.

Diagram



Fields

Field	Function
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Status Flag MSTIDLE bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.
7 ENDTRANSFER	End Transfer Control Software can set ENDTRANSFER bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the End of Transfer bit (FIFOWR[EOT]) had been set before to the last transmission. This capability is included to support cases where it is not known when transmit data is written and if it will be the end of a transfer. - The ENDTRANSFER bit is cleared when the transmitter becomes idle as the transfer ends. - Forcing an end of transfer using the ENDTRANSFER bit causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.
6 STALLED	Stalled Status Flag Indicates whether the SPI is currently in a stall condition.
5 SSD	Slave Select Deassert The Slave Select Deassert flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. SSD flag is cleared by software.
4 SSA	Slave Select Assert The Slave Select Assert flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. SSA flag is cleared by software.
3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined; only zero should be written.
2	Reserved
—	Read value is undefined; only zero should be written.
1-0	Reserved
—	Read value is undefined; only zero should be written.

53.4.5.1.5 Interrupt Enable Register (INTENSET)

Offset

Register	Offset
INTENSET	40Ch

Function

Contains the complete set of interrupt enables for the SPI. Use the INTENSET register to enable various SPI interrupt sources. Enable bits in INTENSET are mapped to locations that correspond to the flags in the STAT register. Writing a 1 to any implemented bit position causes that interrupt to be set.

Use the INTENCLR register to clear the interrupt enable bits in the INTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
R	Reserved																	
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R	Reserved								MSTID LE...		Reserved		SSDE N	SSAE N	Reserv ed	Reserv ed	Reserved	
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Fields

Field	Function
31-9	Reserved
—	Read value is undefined; only zero should be written.
8	Master Idle Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
MSTIDLEEN	0b - No interrupt will be generated when the SPI master function is idle. 1b - An interrupt will be generated when the SPI master function is fully idle.
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSDEN	Slave Select Deassert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is deasserted. 0b - Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted. 1b - Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.
4 SSAEN	Slave Select Assert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is asserted. 0b - Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted. 1b - Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

53.4.5.1.6 Interrupt Enable Clear Register (INTENCLR)

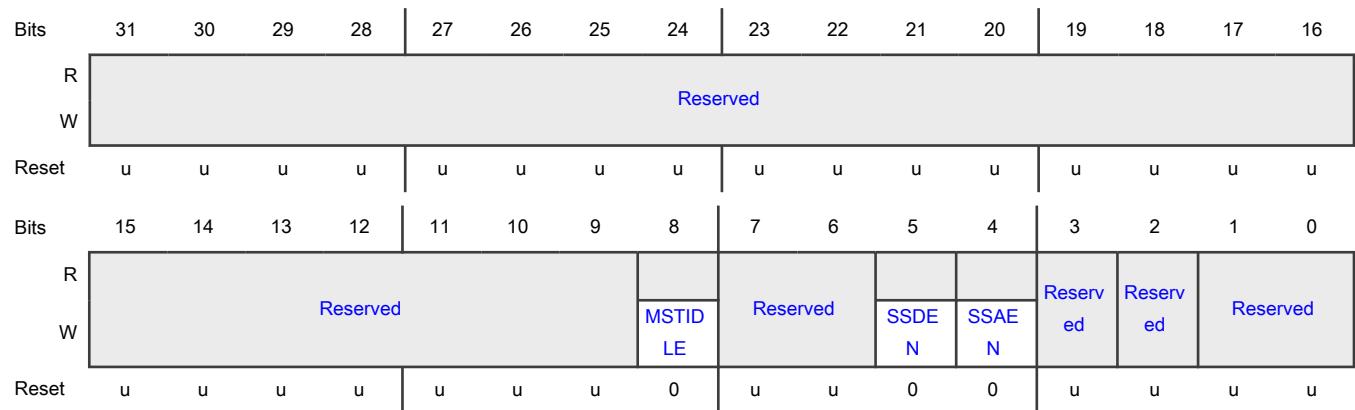
Offset

Register	Offset
INTENCLR	410h

Function

Use the INTENCLR register to clear interrupt enable bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit in the INTENSET register to be cleared.

Diagram



Fields

Field	Function
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Interrupt Enable Writing 1 clears the Master Idle Interrupt Enable bit (INTENSET[MSTIDLEEN]). 0b - No effect 1b - Clear the Master Idle Interrupt Enable bit (INTENSET[MSTIDLE])
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSDEN	Slave Select Deassert Interrupt Enable Writing 1 clears the Slave Select Deassert Interrupt Enable bit (INTENSET[SSDEN]). 0b - No effect 1b - Clear the Slave Select Deassert Interrupt Enable bit (INTENSET[SSDEN])
4 SSAEN	Slave Select Assert Interrupt Enable Writing 1 clears the Slave Select Assert Interrupt Enable bit (INTENSET[SSAEN]). 0b - No effect 1b - Clear the Slave Select Assert Interrupt Enable bit (INTENSET[SSAEN])
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

53.4.5.1.7 Clock Divider Register (DIV)

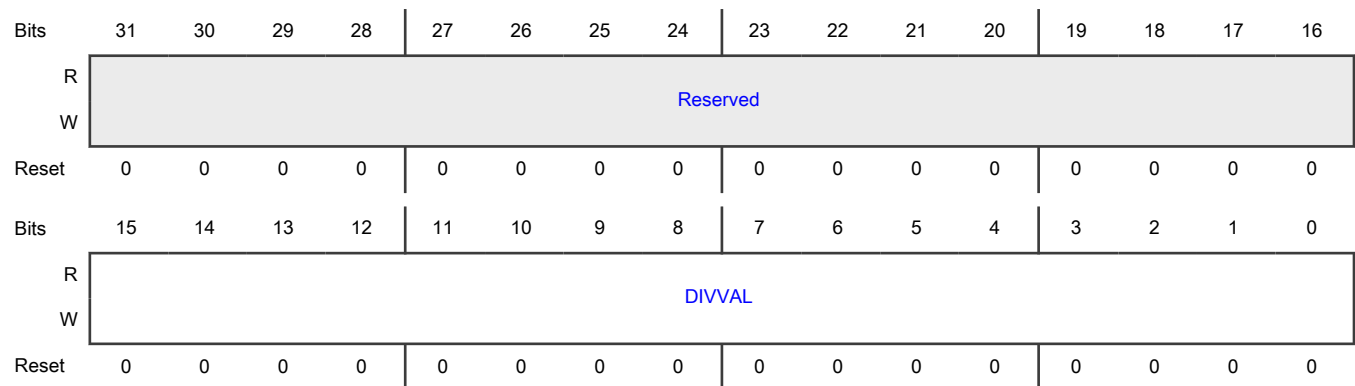
Offset

Register	Offset
DIV	424h

Function

The DIV register specifies the clock divider value used to generate the SPI clock in SPI master mode.

Diagram



Fields

Field	Function
31-16	Reserved
—	Read value is undefined; only zero should be written.
15-0 DIVVAL	Rate Divider Value Specifies how the Flexcomm clock (FCLK) is divided, to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in FCLK/1, the value 1 results in FCLK/2, up to the maximum possible divide value of 0xFFFF, which results in FCLK/65536.

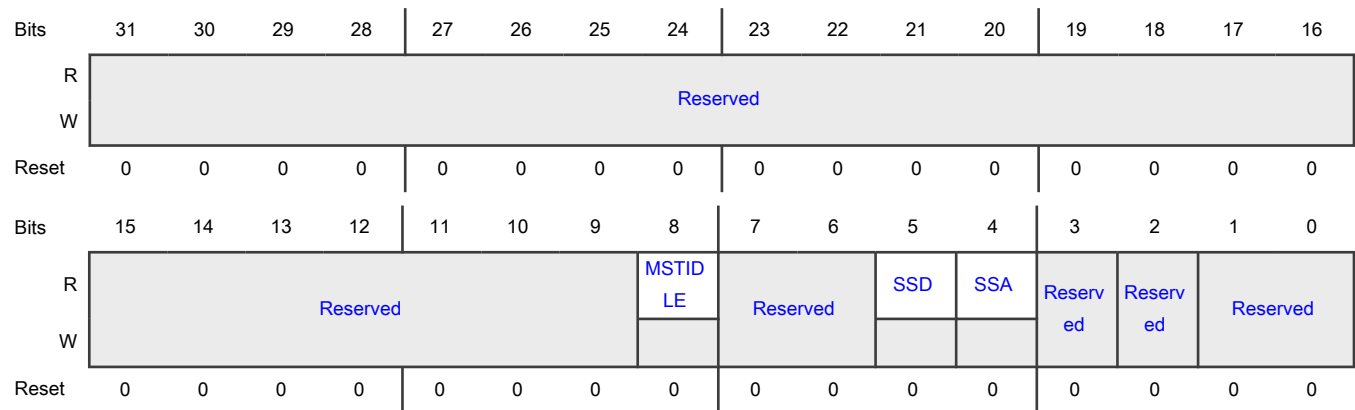
53.4.5.1.8 Interrupt Status Register (INTSTAT)

Offset

Register	Offset
INTSTAT	428h

Function

The read-only INTSTAT register provides a view of the interrupt flags that are currently enabled. This can simplify the software handling of interrupts.

Diagram**Fields**

Field	Function
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Status Flag Interrupt 0b - Disabled 1b - Enabled
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSD	Slave Select Deassert Interrupt 0b - Disabled 1b - Enabled
4 SSA	Slave Select Assert Interrupt 0b - Disabled 1b - Enabled
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

53.4.5.1.9 FIFO Configuration Register (FIFOCFG)

Offset

Register	Offset
FIFOCFG	E00h

Function

This register configures how the SPI uses the FIFO. Before configuring the FIFO, a peripheral function within the Flexcomm module must be selected.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved													POPD	EMPT	EMPT
W														BG	YRX	YTX
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WAKE	WAKE	DMAR	DMAT	Reserved							SIZE	Reserved		ENAB	ENAB
W	RX	TX	X	X											LERX	LETX
Reset	0	0	0	0	0	0	0	0	0	0	u	u	0	0	u	u

Fields

Field	Function
31-19	Reserved
—	Read value is undefined; only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0b - Debug reads of the FIFO do not pop the FIFO 1b - A debug read will cause the FIFO to pop
17 EMPTYRX	Empty Command for the Receive FIFO When a 1 is written to EMPTYRX bit, the RX FIFO is emptied. 0b - No effect 1b - The RX FIFO is emptied
16 EMPTYTX	Empty Command for the Transmit FIFO When a 1 is written to EMPTYTX bit, the TX FIFO is emptied. 0b - No effect 1b - The TX FIFO is emptied

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and then goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See the Hardware Wake-up control register. 0b - Only enabled interrupts will wake up the device form reduced power modes. 1b - A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14 WAKETX	Wake-up for Transmit FIFO Level WAKETX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and then goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See the Hardware Wake-up control register. 0b - Only enabled interrupts will wake up the device form reduced power modes 1b - A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Configuration for Receive 0b - DMA is not used for the receive function. 1b - Issues a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Configuration for Transmit 0b - DMA is not used for the transmit function 1b - Issues DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined; only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. 00b - FIFO is configured as 16 entries of 8 bits. 01b - FIFO is configured as 8 entries of 16 bits. 10b - Not used 11b - Not used
3-2 —	Reserved Read value is undefined; only zero should be written.
1	Enable the Receive FIFO

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENABLERX	0b - The receive FIFO is not enabled 1b - The receive FIFO is enabled
0 ENABLETX	Enable the Transmit FIFO 0b - The transmit FIFO is not enabled 1b - The transmit FIFO is enabled

53.4.5.1.10 FIFO Status Register (FIFOSTAT)

Offset

Register	Offset
FIFOSTAT	E04h

Function

Provides status information for the FIFO, and also indicates if a peripheral has issued an interrupt.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RXLVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				TXLVL				RXFU LL	RXNO TEM...	TXNO TFU...	TXEM PTY	PERIN T	Reserv ed	RXER R	TXER R
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Fields

Field	Function
31-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level RXLVL=0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. RXLVL="other values" tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags will be 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-13 —	Reserved Read value is undefined; only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level TXLVL=0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. TXLVL="other values" tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags will be 0.
7 RXFULL	Receive FIFO is Full 0b - The receive FIFO is not full 1b - The receive FIFO is full. To prevent the peripheral from causing an overflow, data should be read out.
6 RXNOTEMPTY	Receive FIFO is Not Empty 0b - When 0, the receive FIFO is empty 1b - When 1, the receive FIFO is not empty, so data can be read
5 TXNOTFULL	Transmit FIFO is Not Full 0b - The transmit FIFO is full and another write would cause it to overflow 1b - The transmit FIFO is not full, so more data can be written
4 TXEMPTY	Transmit FIFO Empty 0b - The transmit FIFO is not empty 1b - The transmit FIFO is empty, although the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt 0b - The peripheral function has not asserted an interrupt 1b - Indicates that the peripheral function has asserted an interrupt. More information can be found by reading the peripheral's status register (STAT).
2 —	Reserved Read value is undefined; only zero should be written.
1 RXERR	RX FIFO Error Cleared by writing a 1 to itself. 0b - A receive FIFO overflow has not occurred 1b - A receive FIFO overflow has occurred, caused by software or DMA not emptying the FIFO fast enough
0 TXERR	TX FIFO Error Cleared by writing a 1 to itself.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - A transmit FIFO error has not occurred. 1b - A transmit FIFO error has occurred. This error could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed.

53.4.5.1.11 FIFO Trigger Register (FIFOTRIG)

Offset

Register	Offset
FIFOTRIG	E08h

Function

Allows selecting when FIFO-level related interrupts occur, and includes FIFO trigger settings for interrupt and DMA requests.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RXLVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				TXLVL				Reserved				RXLVL ENA		TXLVL ENA	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20	Reserved
—	Read value is undefined; only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. See the Hardware Wake-up control register. NOTE FIFOCFG[WAKERX] allows the device to be woken from reduced power modes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000b - Trigger when the RX FIFO has received 1 entry (is no longer empty) 0001b - Trigger when the RX FIFO has received 2 entries 1111b - Trigger when the RX FIFO has received 16 entries (has become full)
15-12 —	Reserved Read value is undefined; only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. See the Hardware Wake-up control register. NOTE FIFOCFG[WAKETX] allows the device to be woken from reduced power modes. 0000b - Trigger when the TX FIFO becomes empty 0001b - Trigger when the TX FIFO level decreases to 1 entry 1111b - Trigger when the TX FIFO level decreases to 15 entries (is no longer full)
7-2 —	Reserved Read value is undefined; only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable RXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if DMARX in FIFOCFG is set. 0b - Receive FIFO level does not generate a FIFO level trigger 1b - An trigger will be generated if the receive FIFO level reaches the value specified by the FIFOTRIG[RXLVL] field.
0 TXLVLENA	Transmit FIFO Level Trigger Enable TXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if DMATX in FIFOCFG is set. 0b - Transmit FIFO level does not generate a FIFO level trigger 1b - An trigger will be generated if the transmit FIFO level reaches the value specified by the FIFOTRIG[TXLVL] field.

53.4.5.1.12 FIFO Interrupt Enable Register (FIFOINTENSET)

Offset

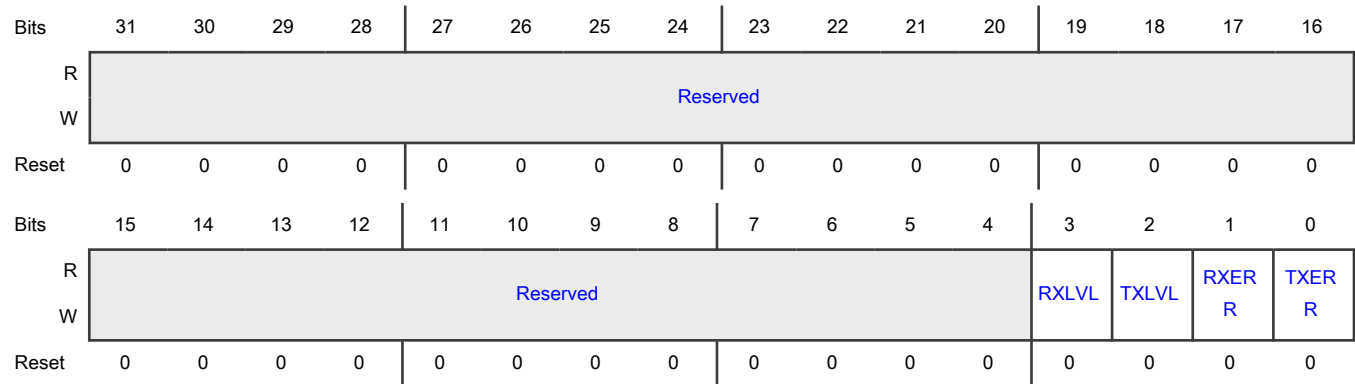
Register	Offset
FIFOINTENSET	E10h

Function

FIFOINTENSET includes the complete set of FIFO interrupt enables. Use the FIFO Interrupt Enable Register to enable various interrupt sources. Writing ones to implemented bits in FIFOINTENSET register forces those bits to be set.

Use the FIFOINTENCLR register to clear the interrupt enable bits in this register.

Diagram



Fields

Field	Function
31-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0b - No interrupt will be generated based on the RX FIFO level 1b - If FIFOTRIG[RXLVLENA]=1, then an interrupt will be generated when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL]
2 TXLVL	Transmit FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0b - No interrupt will be generated based on the TX FIFO level 1b - If FIFOTRIG[TXLVLENA]=1, then an interrupt will be generated when the TX FIFO level decreases to the level specified by FIFOTRIG[TXLVL]
1 RXERR	Receive Error Interrupt Enable Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag 0b - No interrupt will be generated for a receive error 1b - An interrupt will be generated when a receive error occurs
0	TX Error Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXERR	Determines whether an interrupt occurs after a transmit error occurs, based on the FIFOSTAT[TXERR flag] 0b - No interrupt will be generated for a transmit error 1b - An interrupt will be generated when a transmit error occurs

53.4.5.1.13 FIFO Interrupt Enable Clear Register (FIFOINTENCLR)

Offset

Register	Offset
FIFOINTENCLR	E14h

Function

Use the FIFOINTENCLR register to clear interrupt enable bits in FIFOINTENSET. Writing 1 clears the corresponding bits in the FIFOINTENSET register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												RXLVL	TXLVL	RXERR	TXERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-4	Reserved
—	Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[RXLVL] 0b - No effect 1b - Clear the Receive FIFO Level Interrupt Enable bit FIFOINTENSET[RXLVL]

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 TXLVL	Transmit FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[TXLVL] 0b - No effect 1b - Clear the Transmit FIFO Level Interrupt Enable bit FIFOINTENSET[TXLVL]
1 RXERR	Receive Error Interrupt Enable Writing 1 clears FIFOINTENSET[RXERR] 0b - No effect 1b - Clear the Receive Error Interrupt Enable bit FIFOINTENSET[RXERR]
0 TXERR	TX Error Interrupt Enable Writing 1 clears FIFOINTENSET[TXERR] 0b - No effect 1b - Clear the TX Error Interrupt Enable bit FIFOINTENSET[TXERR]

53.4.5.1.14 FIFO Interrupt Status Register (FIFOINTSTAT)

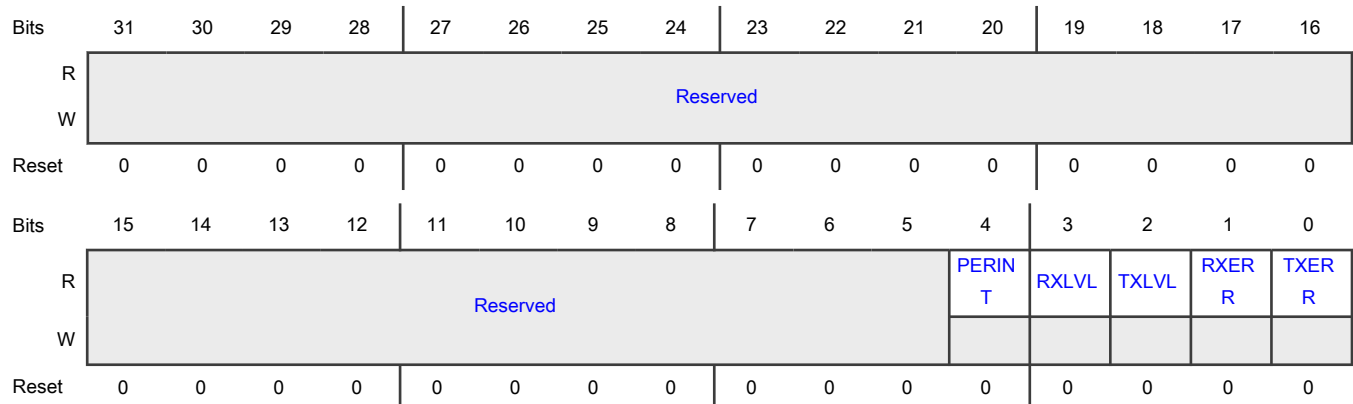
Offset

Register	Offset
FIFOINTSTAT	E18h

Function

The read-only FIFOINTSTAT register provides a view of the interrupt flags that are both pending and currently enabled. This can simplify the software handling of interrupts.

Diagram



Fields

Field	Function
31-5 —	Reserved Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0b - Not pending 1b - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0b - Not pending 1b - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0b - Not pending 1b - Pending
1 RXERR	RX FIFO Error Interrupt Status 0b - Not pending 1b - Pending
0 TXERR	TX FIFO Error Interrupt Status 0b - Not pending 1b - Pending

53.4.5.1.15 FIFO Write Data Register (FIFOWR)**Offset**

Register	Offset
FIFOWR	E20h

Function

The FIFOWR register is used to write values to be transmitted to the FIFO. FIFOWR is a write only register; do not read-modify-write the FIFOWR register.

FIFOWR allows you to alter some SPI controls while sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically. These added fields are described in bits 16 - 27.

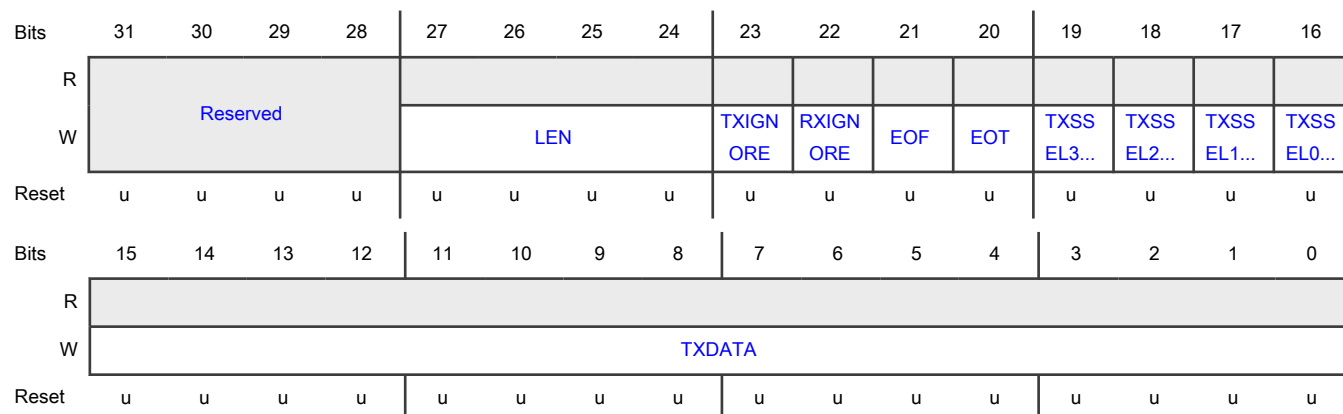
Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified.

- Halfword writes to just the control bits (offset 0xE22) doesn't push anything into the FIFO.
- A zero written to the upper halfword will not modify the control settings.
- Non-zero writes to it will modify all the control bits.

Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO.

- Word writes with the upper halfword of zero, byte writes or halfword writes to FIFOWR will push the data and the current control bits, into the FIFO.
- Word writes with a non-zero upper halfword will modify the control bits before pushing them (the control bits) onto the stack.

Diagram



Fields

Field	Function
31-28	Reserved
—	Read value is undefined; only zero should be written.
27-24 LEN	Data Length Specifies the data length in bits, from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. Except for the reserved values (0x0 - 0x2), for LEN=n, the data length is n+1 bits. 0000b - Reserved 0001b - Reserved 0010b - Reserved 0011b - Data transfer is 4 bits in length 0100b - Data transfer is 5 bits in length 1111b - Data transfer is 16 bits in length
23 TXIGNORE	Transmit Ignore TXIGNORE allows data to be received using the SPI in slave mode without the need to transmit data that is not needed. NOTE TXIGNORE can only be set by writing to the upper 16 bits of FIFOWR, that is, a halfword write to offset 0xE22.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Write transmit data. Write transmit data. Transmit data must be written for each data exchange between master and slave. In slave mode, an underrun error occurs if transmit data is not provided before needed in a data frame. 1b - Ignore transmit data. Ignore transmit data. Data can be received without transmitting data (after FIFOWR has been initialized to set TXIGNORE). No transmitter flags are generated. When configured with TXIGNORE = 1, the slave sets the data to always 0.
22 RXIGNORE	Receive Ignore RXIGNORE allows data to be transmitted using the SPI, without the need to read unneeded data from the receiver. Setting RXIGNORE bit simplifies the transmit process and can be used with DMA. 0b - Read received data. Received data must be read, to allow transmission to proceed. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received. 1b - Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.
21 EOF	End of Frame Between frames, a delay may be inserted, as defined by the Frame_delay value in the DLY register. The end of a frame may not be particularly meaningful if the Frame_delay value = 0. This EOF control can be used as part of the support for frame lengths greater than 16 bits. 0b - Data not EOF. This piece of data transmitted is not treated as the end of a frame. 1b - Data EOF. This piece of data is treated as the end of a frame, causing the Frame_delay time to be inserted before subsequent data is transmitted.
20 EOT	End of Transfer The asserted SSEL will be deasserted at the end of a transfer, and remain to at least the time specified by the Transfer_delay value in the DLY register. 0b - SSEL is not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. 1b - SSEL is deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.
19 TXSSEL3_N	Transmit Slave Select 3 TXSSEL3_N field asserts SSEL3 in master mode. The output on the pin is active LOW by default. 0b - SSEL3 is asserted 1b - SSEL3 is not asserted
18 TXSSEL2_N	Transmit Slave Select 2 TXSSEL2_N field asserts SSEL2 in master mode. The output on the pin is active LOW by default. 0b - SSEL2 is asserted 1b - SSEL2 is not asserted

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 TXSSEL1_N	Transmit Slave Select 1 TXSSEL1_N field asserts SSEL1 in master mode. The output on the pin is active LOW by default. 0b - SSEL1 is asserted 1b - SSEL1 is not asserted
16 TXSSEL0_N	Transmit Slave Select 0 TXSSEL0_N field asserts SSEL0 in master mode. The output on the pin is active LOW by default. 0b - SSEL0 is asserted 1b - SSEL0 is not asserted
15-0 TXDATA	Transmit Data to the FIFO

53.4.5.1.16 FIFO Read Data Register (FIFORD)

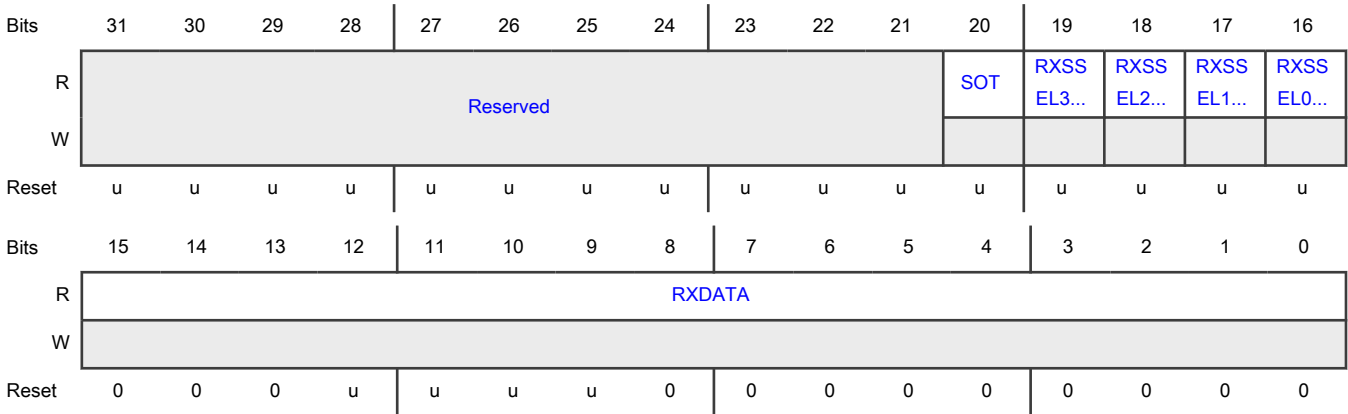
Offset

Register	Offset
FIFORD	E30h

Function

Use the FIFORD register to read values that have been received by the FIFO.

Diagram



Fields

Field	Function
31-21 —	Reserved The value read from a reserved bit is not defined.
20 SOT	Start of Transfer Flag 0b - This is not the 1st data after the SSELs went from deasserted to asserted 1b - This is the 1st data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the 1st piece of data in cases where the transfer length is greater than 16 bits.
19 RXSSEL3_N	Slave Select 3 for Receive RXSSEL3_N field allows the state of the SSEL3 pin to be saved, along with the data received at that pin. The value indicates the SSEL3 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0b - Slave Select 3 is active 1b - Slave Select 3 is not active
18 RXSSEL2_N	Slave Select 2 for Receive RXSSEL2_N field allows the state of the SSEL2 pin to be saved, along with the data received at that pin. The value indicates the SSEL2 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0b - Slave Select 2 is active 1b - Slave Select 2 is not active
17 RXSSEL1_N	Slave Select 1 for Receive RXSSEL1_N field allows the state of the SSEL1 pin to be saved, along with the data received at that pin. The value indicates the SSEL1 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0b - Slave Select 1 is active 1b - Slave Select 1 is not active
16 RXSSEL0_N	Slave Select 0 for Receive RXSSEL0_N field allows the state of the SSEL0 pin to be saved, along with the data received at that pin. The value indicates the SSEL0 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0b - Slave Select 0 is active 1b - Slave Select 0 is not active
15-0 RXDATA	Received Data from the FIFO

53.4.5.1.17 FIFO Data Read with no FIFO Pop Register (FIFORDNOPOP)

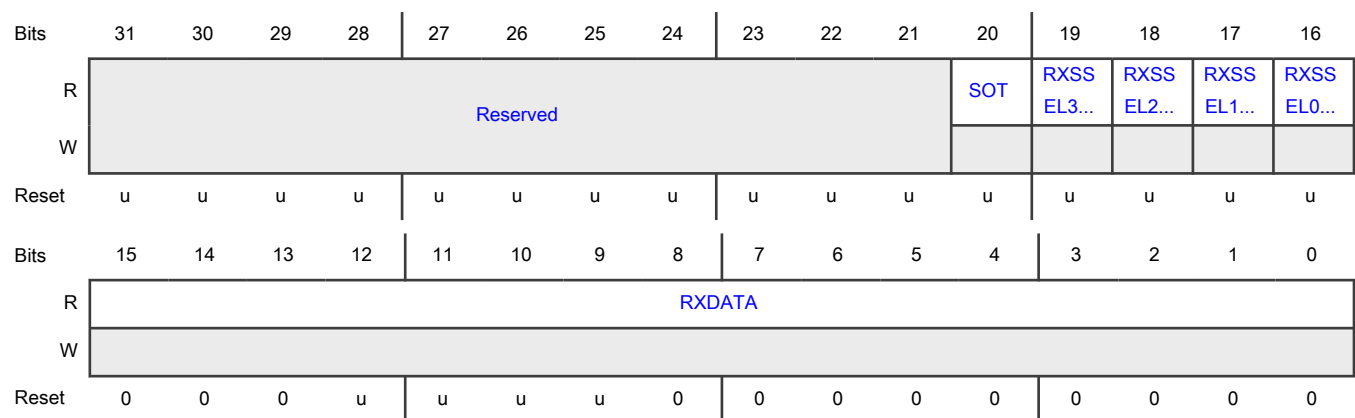
Offset

Register	Offset
FIFORDNOPOP	E40h

Function

The FIFORDNOPOP register acts in exactly the same way as the FIFORD register, except that FIFORDNOPOP supplies data from the top of the FIFO without popping the FIFO (i.e., thereby leaving the FIFO state unchanged). This can be used to allow the system software to observe incoming data without interfering with the peripheral driver.

Diagram



Fields

Field	Function
31-21 —	Reserved The value read from a reserved bit is not defined.
20 SOT	Start of Transfer Flag 0b - Not active 1b - Active
19 RXSSEL3_N	Slave Select 3 for Receive 0b - Not selected 1b - Selected
18 RXSSEL2_N	Slave Select 2 for Receive 0b - Not selected 1b - Selected
17	Slave Select 1 for Receive

Table continues on the next page...

Table continued from the previous page...

Field	Function
RXSSEL1_N	0b - Not selected 1b - Selected
16 RXSSEL0_N	Slave Select 0 for Receive 0b - Not selected 1b - Selected
15-0 RXDATA	Received Data from the FIFO

53.4.5.1.18 FIFO Size Register (FIFOSIZE)

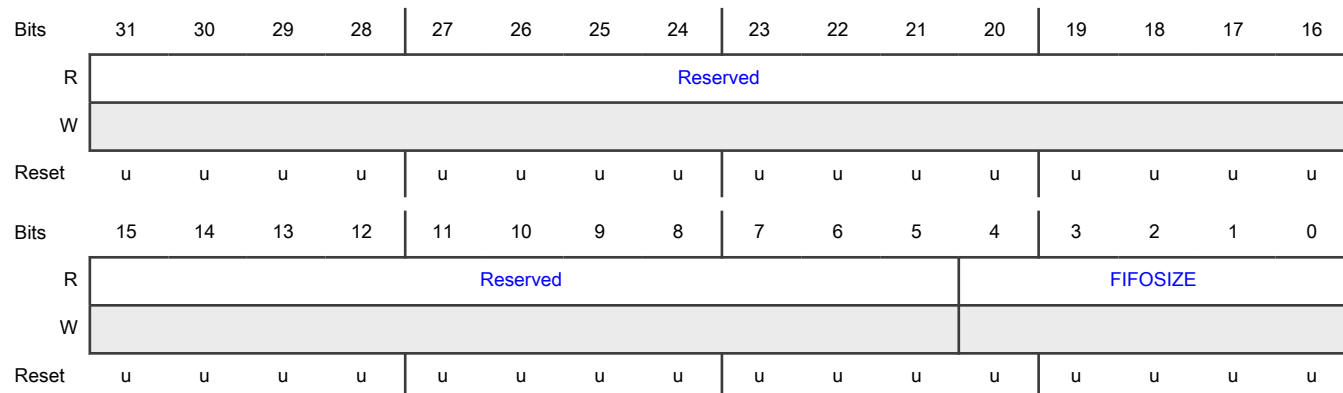
Offset

Register	Offset
FIFOSIZE	E48h

Function

Provides the FIFO size.

Diagram



Fields

Field	Function
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 8 entries for this chip. This field is read only.

53.4.5.1.19 Peripheral Identification Register (ID)

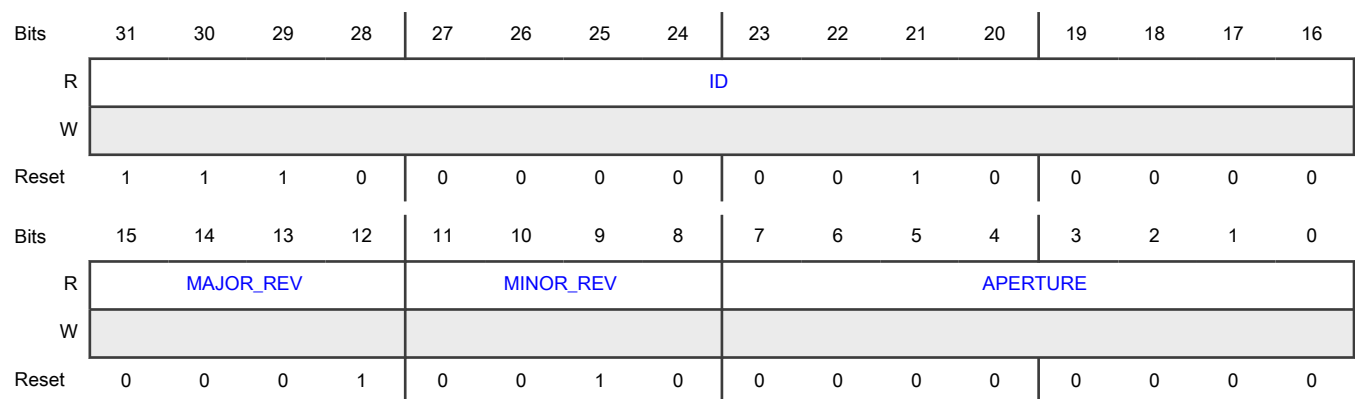
Offset

Register	Offset
ID	FFCh

Function

Identifies the type and revision of the SPI module. A generic software driver can make use of this information to implement specific behavior, based on a module type or a specific revision of that module. This value appears in the shared Flexcomm peripheral ID register when the SPI is selected.

Diagram



Fields

Field	Function
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8 MINOR_REV	Minor revision of module implementation
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

53.5 Inter-integrated Circuit (I2C)

53.5.1 Overview

The Inter-integrated Circuit (I2C) module enables multiple “slave” digital modules to communicate with one or more “master” modules, for short distance communications within a single device.

53.5.1.1 Block diagram

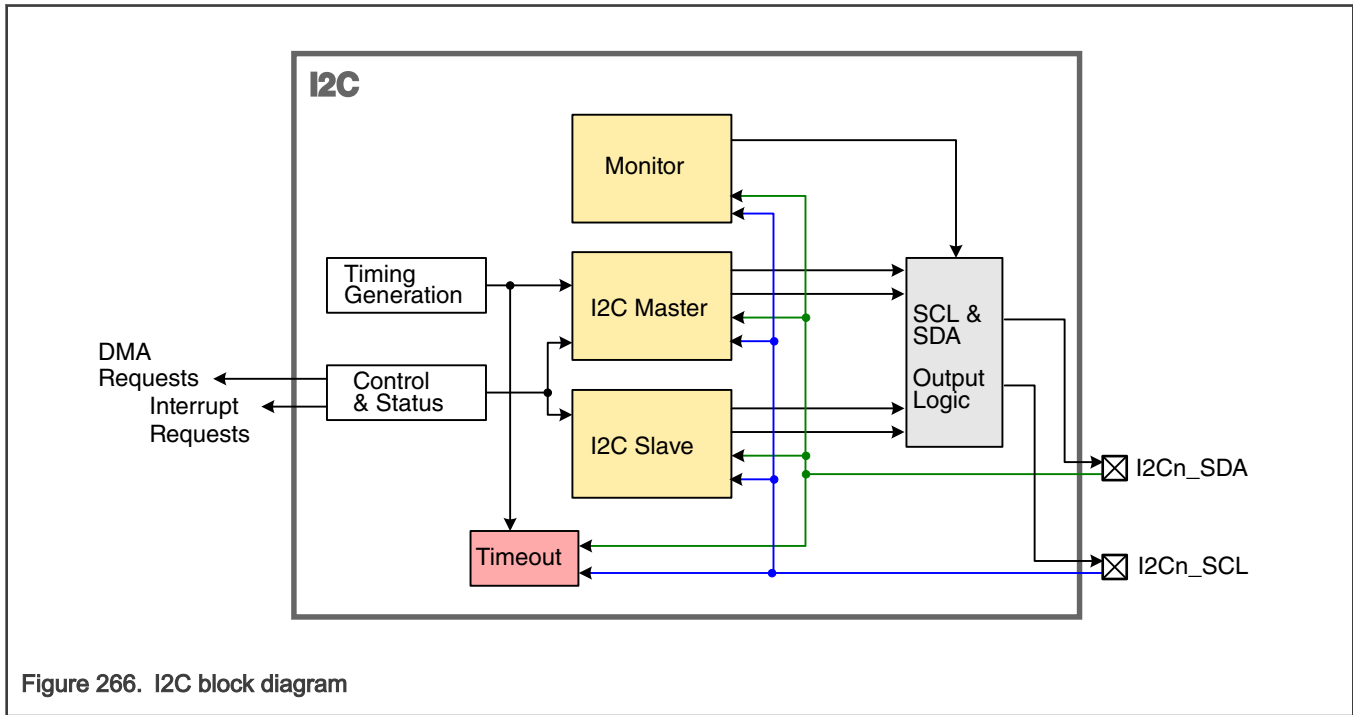


Figure 266. I2C block diagram

53.5.1.2 Features

I2C bus functions are available on the device as a selectable function in each Flexcomm Interface peripheral. The I2C module includes:

- Independent Master, Slave, and Monitor functions
- Bus speeds supported:
 - Standard mode, up to 100 kbits/s
 - Fast-mode, up to 400 kbits/s
 - Fast-mode Plus, up to 1 Mbits/s (on pins that include specific I2C support)
 - High speed mode, 3.4 Mbits/s as a Slave only (on pins that include specific I2C support)
- Supports both Multi-master and Multi-master with Slave functions
- Multiple I2C slave addresses supported in hardware
- To respond to multiple I2C-bus addresses, one slave address can be selectively qualified with a bit mask or an address range
- 10-bit addressing supported with software assist
- Supports System Management Bus (SMBus)
- Separate DMA requests for Master, Slave, and Monitor functions

- No chip clocks are required to receive and compare an address as a Slave, so this event (a Slave address match) can wake up the device from deep sleep mode
- Automatic modes optionally allow less software overhead for some use cases

53.5.2 Functional description

The following sections describe functional details of the I2C module.

53.5.2.1 Operating modes

This section describes all functional operation modes of the I2C module.

53.5.2.1.1 I2C transmit/receive in master mode

The Flexcomm interface is configured to be an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave. The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol (by writing to the MSTCTL register).

If specialized I2C pins are used, then the pins should be configured as required for the I2C bus mode that will be used (SM, FM, FM+, HS). If specialized I2C or standard pins are used, then the pins should be configured correctly.

To configure the I2C bit rate:

- Select a clock source for the Flexcomm interface that will provide the desired I2C bus rate, and divide the clock as needed.
- To further divide the source clock (if needed) using the CLKDIV register.
- Set the SCL high and low times to complete the bus rate setup.

Master write to slave:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a master: set the CFG[MSTEN] bit to 1.
- Write data to the slave:
 1. Write the slave address with the RW bit set to 0 to the Master data register (MSTDAT).
 2. Start the transmission by setting the MSTCTL[MSTSTART] bit to 1. The following happens:
 - The pending status is cleared and the I2C bus is busy.
 - The I2C master sends the start bit and address with the RW bit to the slave.
 3. Wait for the pending status to be set (MSTPENDING = 1), by polling the STAT register.
 4. Write 8 bits of data to the MSTDAT register.
 5. Continue with the transmission of data by setting the MSTCTL[MSTCONT] bit to 1. The following happens:
 - The pending status is cleared and the I2C bus is busy.
 - The I2C master sends the data bits to the slave address.
- Wait for the pending status to be set (MSTPENDING = 1), by polling the STAT register.
- Stop the transmission, by setting the MSTCTL[MSTSTOP] bit to 1.

Master write to slave: code example

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

```

I2C->MSTDAT = (0x23 << 1) | 0;      // address and 0 for R/W bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART;   // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd;                  // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP;     // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

```

Master read from slave:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a master: set the CFG[MSTEN] bit to 1.
- Read data from the slave:
 1. Write the slave address (with the RW bit set to 1) to the Master data register (MSTDAT).
 2. Start the transmission, by setting the MSTCTL[MSTSTART] bit to 1. The following events happen next:
 - The pending status is cleared and the I2C bus is busy.
 - The I2C master sends the start bit and address with the RW bit to the slave.
 - The slave sends 8 bit of data.
 3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
 4. Read 8 bits of data from the MSTDAT register.
 5. Stop the transmission by setting the MSTCTL[MSTSTOP] bit to 1.

Master read from slave: code example

```

// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;

while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1;      // address and 1 for R/W bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART;   // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT;                  // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP;     // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

```

53.5.2.1.2 I²C receive/transmit in slave mode

The Flexcomm interface is configured to be an I2C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data

to be sent to the master are available in the SLVDAT register. After sending and receiving data, the application should continue to the next step of the transmission protocol by writing to the SLVCTL register.

The SCL and SDA functions must be enabled on its pins. The pins should be configured as required for the I2C bus mode that will be used (SM, FM, FM+, HS).

Slave read from master:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a slave with address x:
 - Set the CFG[SLVEN] bit to 1.
 - Write the slave address x to the address 0 match register.
- Read data from the master:
 1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
 2. Acknowledge (“ack”) the address by setting SLVCTL[SLVCONTINUE] = 1.
 3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
 4. Read 8 bits of data from the SLVDAT register.
 5. Acknowledge (“ack”) the data by setting SLVCTL[SLVCONTINUE] = 1.

Slave read from master: code example

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1;          // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;    // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT;                      // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;    // ack data
```

Slave write to master:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a slave with address x:
 - Set the CFG[SLVEN] bit to 1.
 - Write the slave address x to the address 0 match register.
- Write data to the master:
 1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
 2. ACK the address by setting SLVCTL[SLVCONTINUE] = 1.
 3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
 4. Write 8 bits of data to SLVDAT register.
 5. Continue the transaction by setting SLVCTL[SLVCONTINUE] = 1.

Slave write to master: code example

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1;          // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;    // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd;                    // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;    // continue transaction
```

53.5.2.1.3 Configure the I2C for wake-up

In sleep mode, any activity on the I2C bus that triggers an I2C interrupt can wake up the device, if the interrupt is enabled in the INTENSET register and in the interrupt controller. As long as the Flexcomm interface clock remains active in sleep mode, the I2C can wake up the part independently of whether the I2C interface is configured in master or slave mode.

In deep-sleep mode, the I2C clock and all peripheral clocks are turned off. However, if the I2C is configured in slave mode and an external master on the I2C bus provides the clock signal, then the I2C interface can create an interrupt asynchronously. This asynchronous interrupt can then wake up the core, if that interrupt is enabled in the interrupt controller and in the I2C interface INTENCLR register.

Wake-up from sleep mode:

- Enable the I2C interrupt in the interrupt controller.
- Enable the I2C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported. Examples of wake-up events:
 - Master pending
 - Change to idle state
 - Start/stop error
 - Slave pending
 - Address match (in slave mode)
 - Data available/ready

Wake-up from deep-sleep mode:

- Enable the I2C interrupt .
- Enable the I2C interrupt to create the interrupt signal asynchronously while the core and the peripheral are not clocked.
- Configure the I2C in slave mode.
- Enable the I2C interrupt in the INTENCLR register, which configures the interrupt as a wake-up event. Examples of events are:
 - Slave de-select
 - Slave pending (wait for read, write, or ACK)
 - Address match
 - Data available/ready for the Monitor function

53.5.2.2 Operations

This section describes the I2C module's operations.

53.5.2.2.1 Bus timing

AHB bus access limitations

- The internal bus interface to the I2C registers contained in the Flexcomm interface supports only word writes.
- Byte and halfword writes are not supported in the I2C function.

Bus rates and timing considerations

Due to the nature of the I2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C bus, the clock can be stretched by any slave device, extended by software overhead time, and using other methods. In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL, as long as that master is participating in I2C traffic (i.e., when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For example, a bus master allows SCL to go high. It (the bus master) then monitors the line to make sure that the line actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line. Rate calculations give a base frequency that represents the fastest that the I2C bus can operate if nothing slows it (the I2C bus) down.

Rate calculations: Master timing

SCL high time (in Flexcomm interface function clocks) = I2C clock divider x SCL high multiplier

SCL low time (in Flexcomm interface function clocks) = I2C clock divider x SCL low multiplier

Nominal SCL rate = Flexcomm Interface function clock rate / (SCL high time + SCL low time)

NOTE

- DIVVAL must be ≥ 1 .
- For 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. The next table shows the recommended settings for 400 kHz clock rate.

Table 559. Settings for 400 kHz clock rate

Input clock to I2C (MHz)	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLLOW for MSTTIME register
96	59	0	0
48	29	0	0
48	23	0	1
30	14	0	1
24	14	0	0
24	11	0	1
12	5	0	1

Rate calculations: Slave timing

Most aspects of slave operation are controlled by the SCL signal received from the I2C bus master. However, if the slave stretches SCL to allow more time for a software response, it (the slave) must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting 1 clock time of CLKDIV at that point. If CLKDIV is already configured for master operation, then that is sufficient. If only the slave function is used, then CLKDIV should be configured such that 1 clock time is greater than the $t_{SU,DAT}$ value noted in the I2C bus specification for the I2C mode that is being used.

53.5.2.2.2 DMA

DMA with the I2C is performed only for data transfers; DMA cannot handle control of the I2C bus. After DMA is transferring data, I2C acknowledges are handled implicitly. While DMA is transferring data, no CPU intervention is required.

Data transfers can be handled by DMA:

- for Master mode: after an address is sent and acknowledged by a slave.
- for Slave mode: after software has acknowledged an address.

In either mode (Master or Slave), software is always involved in the address part of a message. In master and slave modes, data received and transmit data can be transferred by the DMA. The DMA supports 3 types of DMA requests:

- Data transfer in master mode
- Data transfer in slave mode
- Monitor mode

NOTE

DMA may be used in connection with Automatic Operation, to minimize software overhead time for I2C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop, and then an interrupt will be generated. A Repeated Start sensed on the I2C bus will similarly cause DMA to stop and then an interrupt to be generated. The Monitor function may be used with DMA if a channel is available. See “DMA with I2C monitor mode” for how DMA channels are used with the Monitor function.

DMA as a Master transmitter: basic sequence

- Software sets up DMA to transmit a message.
- Software causes a slave address with a write command to be sent, and then checks to see if the address was acknowledged.
- Software turns on DMA mode in the I2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, like when a slave sends a NACK before the end of the transfer.

DMA as a Master receiver: basic sequence

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and then checks that the address was acknowledged.
- Software starts DMA.
- DMA finishes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

DMA as a Slave transmitter: basic sequence

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA finishes.

DMA as a Slave receiver: basic sequence

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA finishes.
- Software sets SLVNACK before receiving the final data byte.
- Software receives the final data byte.

53.5.2.2.3 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations, and these automatic operation modes are intended to be used primarily with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. The next table shows how these control bits may be used. These cases apply when an address that matches SLVADR0 (and is qualified by SLVQUAL0) is received.

Table 560. Automatic operation cases

AUTONACK bit	AUTOACK bit	Received R/W bit matches AUTOMATCHREAD	SLVPENDING interrupt generated?	ACK/NACK on I2C-bus	Description
0	0	X	Yes	None	Normal, non-automatic operation
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
X	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	X	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

53.5.2.2.4 Time-outs on I2C bus

A time-out feature on an I2C interface can be used to detect a “stuck” I2C bus and potentially do something to fix the condition. Two different types of time-out are supported: event timeout (STAT[EVENTTIMEOUT]) and SCL timeout (STAT[SCLTIMEOUT]). Both time-out types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

- **EVENTTIMEOUT flag:** In the STAT register, the EVENTTIMEOUT flag is asserted when the time between bus events is longer than the time configured in the TIMEOUT register. These bus events include Start, Stop, and all changes on the I2C clock (SCL). The EVENTTIMEOUT time-out can be used to monitor an I2C bus within a system, as part of a scheme to keep the bus running after problems occur.
- **SCLTIMEOUT flag:** In the STAT register, the SCLTIMEOUT flag is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to the SMBus time-out parameter TTIMEOUT. In this situation, a slave could reset its own I2C interface, in case it (the slave) is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the I2C bus will be released, unless a current master is causing the problem. Refer to the SMBus specification for more details.

Both types of I2C bus time-outs are generated only when the I2C bus is considered busy, for example, when there has been a Start condition more recently than a Stop condition.

53.5.2.2.5 10-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a specific range of standard 7-bit addresses.

- For a master writing to a slave, the I2C frame simply continues with data after the 2 address bytes.
- For a master reading from a slave, the master needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it (the slave) has been addressed by the previous write operation and the slave must stay selected for the subsequent read with the correct partial I2C address.

For the Master function: the I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function: the first part of the address is automatically matched in the same way as 7-bit addressing. The slave address qualifier feature can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one 10-bit address. In the case of Slave Receiver mode, data is received in the normal way after software matches the first data byte to the remaining portion of the 10-bit address. The Slave should record the fact that it (the slave) has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode: the slave responds to the initial address in the same way as for Slave Receiver mode, and checks that it (the slave) has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, then software must verify that the first part of the 10-bit address is a complete match to the previous address, before the software acknowledges the address.

53.5.2.3 Clocks

This section describes clocks and special clocking requirements of the I2C module.

- The Master function of the I2C always requires a peripheral clock to be running.
- The Slave function actually can operate without any internal clocking when the slave is not currently addressed.

This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

53.5.3 Initialization

I2C initialization

To initialize the I2C and related clocks:

- Reset the Flexcomm interface that is about to use the I2C function.

- Select the desired Flexcomm interface function by writing to the PSELID register of the related Flexcomm Interface. Note that any previous selection that has been made will be cleared if the Flexcomm interface itself is reset via the PRESETCTRL1 register.
- Configure the I2C for the desired functions:
 - Enable the clock to the Flexcomm register interface.
 - Enable or disable the related Flexcomm interface interrupts.
 - Configure the related Flexcomm Interface pin functions.
 - Configure the I2C clock and data rate, for both master and slave modes. The Flexcomm interface clock frequency should not be above 48 MHz.

NOTE

Although the I2C function is incorporated into the Flexcomm interface, the I2C function does not use the Flexcomm interface FIFO.

53.5.4 Memory Map and register definition

This section includes the I2C module memory map and detailed descriptions of all registers.

53.5.4.1 I2C Bus Interface register descriptions

53.5.4.1.1 I2C memory map

FLEXCOMM0.I2C base address: 4010_6000h

FLEXCOMM1.I2C base address: 4010_7000h

FLEXCOMM2.I2C base address: 4010_8000h

FLEXCOMM3.I2C base address: 4010_9000h

FLEXCOMM4.I2C base address: 4012_2000h

FLEXCOMM5.I2C base address: 4012_3000h

FLEXCOMM6.I2C base address: 4012_4000h

FLEXCOMM7.I2C base address: 4012_5000h

FLEXCOMM8.I2C base address: 4020_9000h

FLEXCOMM9.I2C base address: 4020_A000h

FLEXCOMM10.I2C base address: 4020_B000h

FLEXCOMM11.I2C base address: 4020_C000h

FLEXCOMM12.I2C base address: 4020_D000h

FLEXCOMM13.I2C base address: 4020_E000h

FLEXCOMM15.I2C base address: 4012_7000h

Offset	Register	Width (In bits)	Access	Reset value
800h	Configuration Register (CFG)	32	RW	See section
804h	Status Register (STAT)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
808h	Interrupt Enable Set Register (INTENSET)	32	RW	See section
80Ch	Interrupt Enable Clear Register (INTENCLR)	32	RW	See section
810h	Time-out Register (TIMEOUT)	32	RW	See section
814h	Clock Divider Register (CLKDIV)	32	RW	See section
818h	Interrupt Status Register (INTSTAT)	32	R	See section
820h	Master Control Register (MSTCTL)	32	RW	See section
824h	Master Timing Register (MSTTIME)	32	RW	See section
828h	Master Data Register (MSTDAT)	32	RW	See section
840h	Slave Control Register (SLVCTL)	32	RW	See section
844h	Slave Data Register (SLVDAT)	32	RW	See section
848h - 854h	Slave Address Register (SLVADR0 - SLVADR3)	32	RW	See section
858h	Slave Qualification for Address 0 Register (SLVQUAL0)	32	RW	See section
880h	Monitor Receiver Data Register (MONRXDAT)	32	R	See section
FFCh	Peripheral Identification Register (ID)	32	R	E030_1300h

53.5.4.1.2 Configuration Register (CFG)

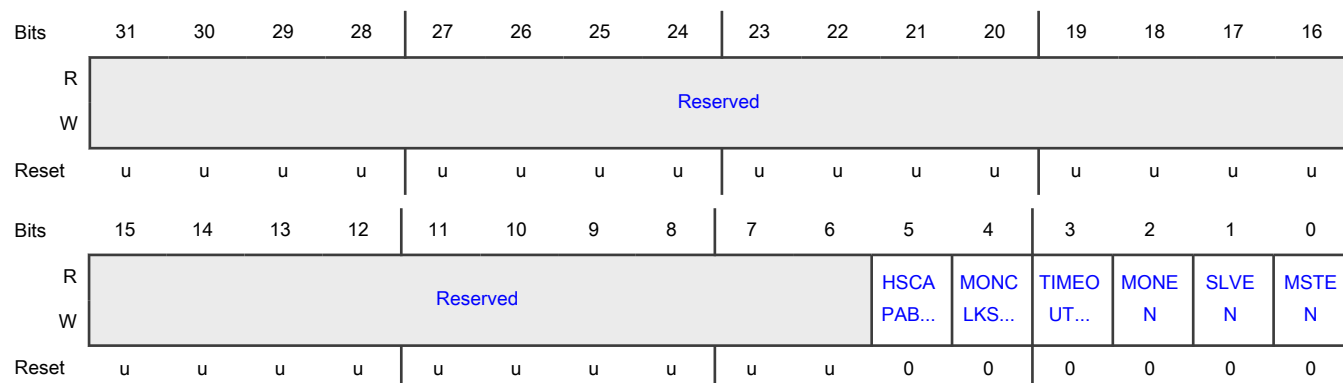
Offset

Register	Offset
CFG	800h

Function

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Diagram



Fields

Field	Function
31-6 —	Reserved Read value is undefined; only zero should be written.
5 HSCAPABLE	High Speed mode Capable enable Enabling High Speed mode applies to all functions (Master, Slave, Monitor), because High Speed mode alters the way that the I2C pins drive and filter, as well as the timing for certain I2C signalling. The I2C interface will support Standard mode, Fast mode, and Fast mode Plus, if and only if those pins' hardware support these modes. Any changes to be made to the pin controls, like changing the drive strength or filtering, must be made by software using the IOCON register associated with each I2C pin. 0b - Fast mode Plus enable 1b - High Speed mode enable
4 MONCLKSTR	Monitor function Clock Stretching 0b - Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it (the data) is overwritten. This mode can be used when non-invasive monitoring is critical. 1b - Enabled. The Monitor function will perform clock stretching, to ensure that the software or DMA can read all incoming data supplied by the Monitor function.
3 TIMEOUTEN	I2C bus Time-out Enable 0b - Disabled. The time-out function is disabled. When disabled, the time-out function is internally reset. 1b - Enabled. The time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if those flags are enabled. Typically, only one time-out flag will be used in a system.
2 MONEN	Monitor Enable 0b - Disabled. The I2C Monitor function is disabled. When disabled, the Monitor function configuration settings are not changed, but the Monitor function is internally reset. 1b - Enabled. The I2C Monitor function is enabled.
1 SLVEN	Slave Enable 0b - Disabled. The I2C slave function is disabled. When disabled, the Slave configuration settings are not changed, but the Slave function is internally reset. 1b - Enabled. The I2C slave function is enabled.
0 MSTEN	Master Enable 0b - Disabled. The I2C Master function is disabled. When disabled, the Master configuration settings are not changed, but the Master function is internally reset. 1b - Enabled. The I2C Master function is enabled.

53.5.4.1.3 Status Register (STAT)

Offset

Register	Offset
STAT	804h

Function

Status register for Master, Slave, and Monitor functions. The STAT register provides status flags and state information about all of the functions of the I2C interface. Access to bits in this register varies. RO = Read-only, W1C = write 1 to clear.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				SCLTIMEOUT		EVENTTIMEOUT		Reserved				MONITOR	MONITOR	MONITOR	MONITOR
W																
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVD ESEL	SLVSE L	SLVIDX		SLVNOTS...	SLVSTATE		SLVPE ND...	Reserved	MSTST TST...	Reserved	MSTARB...	MSTSTATE		MSTP END...	
W																
Reset	0	0	0	0	1	0	0	0	u	0	u	0	0	0	0	1

Fields

Field	Function
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUT	SCL Time-out Interrupt flag Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. Can be cleared by writing a 1 to itself (SCLTIMEOUT). 0b - No time-out. SCL low time has not caused a time-out. 1b - Time-out. SCL low time has caused a time-out.
24 EVENTTIMEOUT	Event Time-out Interrupt flag Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. - No time-out is created when the I2C bus is idle. - Can be cleared by writing a 1 to itself (EVENTTIMEOUT). 0b - No time-out. I2C bus events have not caused a time-out.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLE	Monitor Idle flag MONIDLE flag is set when the Monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. • MONIDLE flag will cause an interrupt when set (if enabled via the INTENSET register). • Can be cleared by writing a 1 to itself (MONIDLE). 0b - Not idle. The I2C bus is not idle, or MONIDLE flag has been cleared by software. 1b - Idle. The I2C bus has gone idle at least once, since the last time MONIDLE flag was cleared by software.
18 MONACTIVE	Monitor Active flag Indicates when the Monitor function considers the I2C bus to be active. Active is defined here as when a Master is on the bus, meaning that a bus Start has occurred more recently than a bus Stop. 0b - Inactive. The Monitor function considers the I2C bus to be inactive. 1b - Active. The Monitor function considers the I2C bus to be active.
17 MONOV	Monitor Overflow flag 0b - No overrun. Monitor data has not overrun. 1b - Overrun. A Monitor data overrun has occurred. An overrun can only happen when Monitor clock stretching not enabled via the CFG[MONCLKSTR] bit. Writing 1 to MONOV bit clears the MONOV flag.
16 MONRDY	Monitor Ready MONRDY flag is cleared when the MONRXDAT register is read. 0b - No data. The Monitor function does not currently have data available. 1b - Data waiting. The Monitor function has data waiting to be read.
15 SLVDESEL	Slave Deselected flag • SLVDESEL flag will cause an interrupt when set (if enabled via INTENSET). • SLVDESEL flag can be cleared by writing a 1 to itself (SLVDESEL). 0b - Not deselected. The Slave function has not become deselected. This does not mean that the Slave is currently selected. That information is in the SLVSEL flag. 1b - Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See SLVSEL for details about when that event occurs.
14	Slave selected flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
SLVSEL	SLVSEL is set after an address match • when software tells the Slave function to acknowledge the address, • or when the address has been automatically acknowledged. SLVSEL is cleared • when another address cycle presents an address that does not match an enabled address on the Slave function, • or when slave software decides to NACK a matched address, • or when there is a Stop detected on the bus, • or when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data. 0b - Not selected. The Slave function is not currently selected. 1b - Selected. The Slave function is currently selected.
13-12 SLVIDX	Slave address match Index T SLVIDX field is valid when the I2C slave function has been selected, by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and also provides an identification of the address that was matched. It is possible that more than one address could have been matched, but only one match can be reported here. 00b - Address 0. Slave address 0 was matched. 01b - Address 1. Slave address 1 was matched. 10b - Address 2. Slave address 2 was matched. 11b - Address 3. Slave address 3 was matched.
11 SLVNOTSTR	Slave Not Stretching Indicates when the slave function is stretching the I2C clock. Clock stretching is needed to gracefully invoke Deep Sleep modes during slave operations. This read-only flag reflects the slave function status in real time. 0b - Stretching. The slave function is currently stretching the I2C bus clock. Deep-Sleepmode cannot be entered at this time. 1b - Not stretching. The slave function is not currently stretching the I2C bus clock. Deep-sleep mode can be entered at this time.
10-9 SLVSTATE	Slave State Each value of the SLVSTATE field indicates a specific required service for the Slave function. Note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes. 00b - Slave address. Address plus R/W received. At least one of the 4 slave addresses has been matched by hardware. 01b - Slave receive. Received data is available (in Slave Receiver mode).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	10b - Slave transmit. Data can be transmitted (in Slave Transmitter mode). 11b - Reserved
8 SLVPENDING	Slave Pending Indicates that the Slave function is waiting to continue communication on the I2C bus and needs software service. • The SLVPENDING flag will cause an interrupt when set (and if the interrupt is enabled via INTENSET register). • The SLVPENDING flag is not set when the DMA is handling an event (if SLVCTL[SLVDMA] bit is set). • The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to SLVCTL[SLVCONTINUE]. The point in time when SlvPending is set depends on whether the I2C interface is in HSCAPABLE mode. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to HS I2C specification requirements, slave addresses must also be detected automatically, because the address must be acknowledged before the clock can be stretched. 0b - In progress. The Slave function does not currently need software service. 1b - Pending. The Slave function needs software service. Information about what is needed is in the Slave state field (SLVSTATE).
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPER R	Master Start/Stop Error flag • Cleared by software writing a 1 to this bit (MSTSTSTPERR) • Cleared automatically when a 1 is written to MSTCONTINUE 0b - No Start/Stop Error has occurred. 1b - The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when Start or Stop is not allowed by the I2C specification. The Master interface has stopped driving the bus and gone into an idle state; no action is required. A request for a Start could be made, or software could attempt to make sure that the bus has not stalled.
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSS	Master Arbitration Loss flag • Cleared by software writing a 1 to this bit (MSTARBLOSS) • Cleared automatically when a 1 is written to MSTCONTINUE 0b - No Arbitration Loss has occurred

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and has gone into an idle state. Software can respond by doing nothing, or by sending a Start (to attempt to gain control of the bus when the bus next becomes idle).
3-1 MSTSTATE	Master State code The master state code indicates the master state when the MSTPENDING bit is set, which means that the master is pending or is in the idle state. Each value of the MSTSTATE field indicates a specific required service for the Master function. All other MSTSTATE values are reserved. 000b - Idle. The Master function is available to be used for a new transaction. 001b - Receive ready. Received data is available (in Master Receiver mode). Address plus Read was previously sent and Acknowledged by a slave. 010b - Transmit ready. Data can be transmitted (in Master Transmitter mode). Address plus Write was previously sent and Acknowledged by a slave. 011b - NACK Address. Slave NACKed address. 100b - NACK Data. Slave NACKed transmitted data.
0 MSTPENDING	Master Pending Indicates that the Master is either waiting to continue communications on the I2C bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service (if any) that the master expects. - The MSTPENDING flag will cause an interrupt when set (if enabled via the INTENSET register). - The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). - If the master is in the idle state, and if no communication is needed, then mask this interrupt (MSTPENDING). 0b - In progress. Communication is in progress and the Master function is busy and cannot currently accept a command. 1b - Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, then the master is waiting to receive or transmit data, or is waiting for the NACK bit.

53.5.4.1.4 Interrupt Enable Set Register (INTENSET)

Offset

Register	Offset
INTENSET	808h

Function

The INTENSET register controls which I2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						SCLTIMEOUTEN	EVENTTIMEOUTEN	Reserved				MONIDLEEN	Reserved	MONOVEN	MONROVER
W																
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVDSEEN	Reserved				SLVNOTSENT	Reserved		SLVPELND	Reserved	MSTSTST	Reserved	MSTARBL	Reserved		MSTPEND
W																
Reset	0	u	u	u	0	u	u	0	u	0	u	0	u	u	u	0

Fields

Field	Function
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUTEN	SCL Time-out interrupt Enable 0b - Disabled. The SCL time-out interrupt is disabled. 1b - Enabled. The SCL time-out interrupt is enabled.
24 EVENTTIMEOUTEN	Event Time-out interrupt Enable 0b - Disabled. The Event time-out interrupt is disabled. 1b - Enabled. The Event time-out interrupt is enabled.
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLEEN	Monitor Idle interrupt Enable 0b - Disabled. The MonIdle interrupt is disabled. 1b - Enabled. The MonIdle interrupt is enabled.
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOVEN	Monitor Overrun interrupt Enable 0b - Disabled. The MonOv interrupt is disabled. 1b - Enabled. The MonOv interrupt is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 MONRDYEN	Monitor data Ready interrupt Enable 0b - Disabled. The MonRdy interrupt is disabled. 1b - Enabled. The MonRdy interrupt is enabled.
15 SLVDESELEN	Slave Deselect interrupt Enable 0b - Disabled. The SlvDeSel interrupt is disabled. 1b - Enabled. The SlvDeSel interrupt is enabled.
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTRE N	Slave Not Stretching interrupt Enable 0b - Disabled. The SlvNotStr interrupt is disabled. 1b - Enabled. The SlvNotStr interrupt is enabled.
10-9 —	Reserved Read value is undefined; only zero should be written.
8 SLVPENDINGE N	Slave Pending interrupt Enable 0b - Disabled. The SlvPending interrupt is disabled. 1b - Enabled. The SlvPending interrupt is enabled.
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPER REN	Master Start/Stop Error interrupt Enable 0b - Disabled. The MstStStpErr interrupt is disabled. 1b - Enabled. The MstStStpErr interrupt is enabled.
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSS EN	Master Arbitration Loss interrupt Enable 0b - Disabled. The MstArbLoss interrupt is disabled. 1b - Enabled. The MstArbLoss interrupt is enabled.
3-1 —	Reserved Read value is undefined; only zero should be written.
0 MSTPENDING EN	Master Pending interrupt Enable 0b - Disabled. The MstPending interrupt is disabled. 1b - Enabled. The MstPending interrupt is enabled.

53.5.4.1.5 Interrupt Enable Clear Register (INTENCLR)

Offset

Register	Offset
INTENCLR	80Ch

Function

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register. Bits that do not correspond to defined bits in INTENSET are reserved and only zeros should be written to them.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserved				MONI DLE...	Reserv ed	MONO VCLR	MONR DYC...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVD ESE...	Reserved				SLVN OTS...	Reserved		Reserv ed	MSTS TST...	Reserv ed	MSTA RBL...	Reserved			MSTP END...
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUTCLR	SCL time-out interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
24 EVENTTIMEOUTCLR	Event time-out interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
23-20 —	Reserved Read value is undefined; only zero should be written.
19	Monitor Idle interrupt clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
MONIDLECLR	0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOVCLR	Monitor Overrun interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
16 MONRDYCLR	Monitor data Ready interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
15 SLVDESELCLR	Slave Deselect interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTRCLR	Slave Not Stretching interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
10-9 —	Reserved Read value is undefined; only zero should be written.
8 SLVPENDINGCLR	Slave Pending interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPERCLR	Master Start/Stop Error interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
5 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 MSTARBLOSS CLR	Master Arbitration Loss interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register
3-1 —	Reserved Read value is undefined; only zero should be written.
0 MSTPENDING CLR	Master Pending interrupt clear 0b - No effect on interrupt 1b - Clears the interrupt bit in INTENSET register

53.5.4.1.6 Time-out Register (TIMEOUT)

Offset

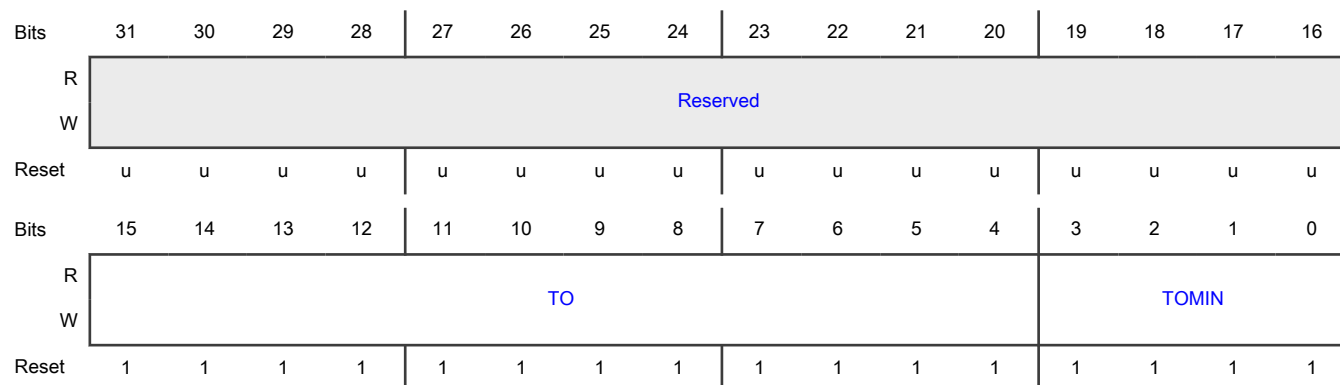
Register	Offset
TIMEOUT	810h

Function

The TIMEOUT register allows setting an upper limit to certain I2C-bus times, informing by status flag and/or interrupt when those times are exceeded. Two time-outs are generated, and software can elect to use either of the timeouts.

- EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The STAT[EVENTTIMEOUT] status flag is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt (if enabled to do so by the INTENSET[EVENTTIMEOUTEN] bit).
- SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The STAT[SCLTIMEOUT] status flag is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt (if enabled to do so by the INTENSET[SCLTIMEOUTEN] bit). The SCLTIMEOUT can be used with the SMBus.

Diagram



Fields

Field	Function
31-16 —	Reserved Read value is undefined; only zero should be written.
15-4 TO	Time-out time value Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this Time-out value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0000_0000_0000b - A time-out will occur after 16 counts of the I2C function clock. 0000_0000_0001b - A time-out will occur after 32 counts of the I2C function clock. 1111_1111_1111b - A time-out will occur after 65,536 counts of the I2C function clock.
3-0 TOMIN	Time-out time value, the bottom 4 bits The bottom 4 bits are hard-wired to 0xF, which gives a minimum time-out of 16 I2C function clocks, and also a time-out resolution of 16 I2C function clocks.

53.5.4.1.7 Clock Divider Register (CLKDIV)

Offset

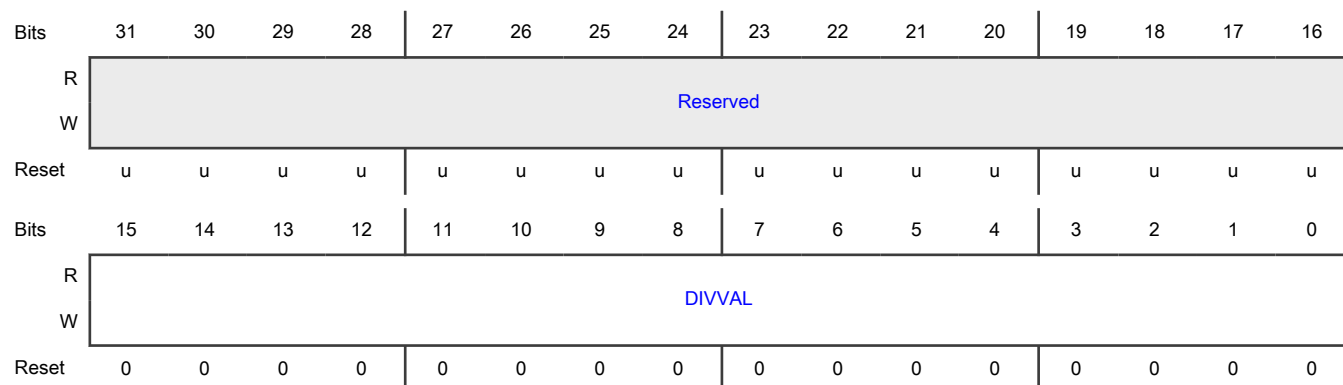
Register	Offset
CLKDIV	814h

Function

Clock pre-divider for the entire I2C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.

The CLKDIV register divides down the Flexcomm clock (FCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C-bus specification, some of which are user-configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

Diagram



Fields

Field	Function
31-16 —	Reserved Read value is undefined; only zero should be written.
15-0 DIVVAL	Divider Value Controls how the Flexcomm clock (FCLK) is used by the I2C functions that need an internal clock (to operate). 0000_0000_0000_0000b - FCLK is used directly by the I2C. 0000_0000_0000_0001b - FCLK is divided by 2 before being used by the I2C. 0000_0000_0000_0010b - FCLK is divided by 3 before being used by the I2C. 1111_1111_1111_1111b - FCLK is divided by 65,536 before being used by the I2C.

53.5.4.1.8 Interrupt Status Register (INTSTAT)

Offset

Register	Offset
INTSTAT	818h

Function

The INTSTAT register provides register provides a view of the Master, Slave, and Monitor interrupt flags that are currently enabled. This register can simplify the software handling of interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				SCLTI ME...		EVEN TTI...		Reserved				MONI DLE	Reserv ed	MONO V	MONR DY
W																
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVD ESEL	Reserved				SLVN OTS...	Reserved		SLVPE ND...	Reserv ed	MSTS TST...	Reserv ed	MSTA RBL...	Reserved		MSTP END...
W																
Reset	0	u	u	u	1	u	u	0	u	0	u	0	u	u	u	1

Fields

Field	Function
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUT	SCL Time-out Interrupt flag 0b - Not active 1b - Active
24 EVENTTIMEOUT	Event Time-out Interrupt flag 0b - Not active 1b - Active
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLE	Monitor Idle flag 0b - Not active 1b - Active
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOV	Monitor Overflow flag 0b - Not active 1b - Active
16 MONRDY	Monitor Ready 0b - Not active 1b - Active
15 SLVDESEL	Slave Deselected flag 0b - Not active 1b - Active
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTR	Slave Not Stretching status 0b - Not active 1b - Active
10-9 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 SLVPENDING	Slave Pending 0b - Not active 1b - Active
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPER R	Master Start/Stop Error flag 0b - Not active 1b - Active
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSS	Master Arbitration Loss flag 0b - Not active 1b - Active
3-1 —	Reserved
0 MSTPENDING	Master Pending 0b - Not active 1b - Active

53.5.4.1.9 Master Control Register (MSTCTL)

Offset

Register	Offset
MSTCTL	820h

Function

The MSTCTL register contains bits that control various functions of the I2C Master interface. Only write to the MSTCTL register when the master is pending (STAT[MSTPENDING] = 1). Software should always write a complete value to MSTCTL, and not OR new control bits into the register, as is possible in other registers such as CFG. This is because MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I2C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes; in this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as when setting either the MSTSTOP or MSTSTART control bit.

NOTE

When in the idle or slave NACKed states, set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Diagram**Fields**

Field	Function
31-4	Reserved
—	Read value is undefined; only zero should be written.
3 MSTDMA	Master DMA enable Data operations of the I2C can be performed with DMA. - Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. - Address acknowledgement must also be done by software, except when the I2C is configured to be High Speed mode capable (CFG[HSCAPABLE]) and address acknowledgement is handled entirely by hardware, or when Automatic Operation is enabled. - When a DMA data transfer is complete, MSTDMA must be cleared before beginning the next operation (typically a Start or Stop). 0b - Disable. No DMA requests are generated for master operation. 1b - Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.
2 MSTSTOP	Master Stop control(write-only) 0b - No effect 1b - Stop. A Stop will be generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (in Master Receiver mode).
1 MSTSTART	Master Start control(write-only) 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Start. A Start will be generated on the I2C bus at the next allowed time.
0 MSTCONTINUE	Master Continue(write-only) 0b - No effect 1b - Continue. Informs the Master function to continue to the next operation. This action must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.

53.5.4.1.10 Master Timing Register (MSTTIME)

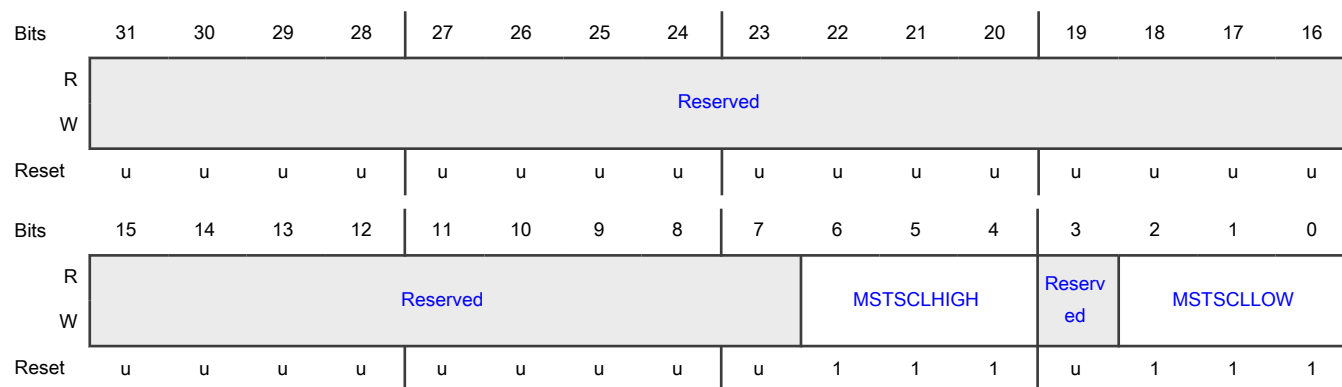
Offset

Register	Offset
MSTTIME	824h

Function

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

Diagram



Fields

Field	Function
31-7 —	Reserved Read value is undefined; only zero should be written.
6-4 MSTSCLHIGH	Master SCL High time Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I2C bus

Table continues on the next page...

Table continued from the previous page...

Field	Function
	specification. I2C bus specification parameters $t_{SU,STO}$ and $t_{HD,STA}$ have the same values and are also controlled by MSTSCLHIGH. 000b - 2 clocks. Minimum SCL high time is 2 clocks of the I2C clock pre-divider. 001b - 3 clocks. Minimum SCL high time is 3 clocks of the I2C clock pre-divider . 010b - 4 clocks. Minimum SCL high time is 4 clocks of the I2C clock pre-divider. 011b - 5 clocks. Minimum SCL high time is 5 clocks of the I2C clock pre-divider. 100b - 6 clocks. Minimum SCL high time is 6 clocks of the I2C clock pre-divider. 101b - 7 clocks. Minimum SCL high time is 7 clocks of the I2C clock pre-divider. 110b - 8 clocks. Minimum SCL high time is 8 clocks of the I2C clock pre-divider. 111b - 9 clocks. Minimum SCL high time is 9 clocks of the I2C clock pre-divider.
3 —	Reserved
2-0 MSTSCLLOW	Master SCL Low time Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I2C bus specification. I2C bus specification parameters t_{BUF} and $t_{SU,STA}$ have the same values and are also controlled by MSTSCLLOW. 000b - 2 clocks. Minimum SCL low time is 2 clocks of the I2C clock pre-divider. 001b - 3 clocks. Minimum SCL low time is 3 clocks of the I2C clock pre-divider. 010b - 4 clocks. Minimum SCL low time is 4 clocks of the I2C clock pre-divider. 011b - 5 clocks. Minimum SCL low time is 5 clocks of the I2C clock pre-divider. 100b - 6 clocks. Minimum SCL low time is 6 clocks of the I2C clock pre-divider. 101b - 7 clocks. Minimum SCL low time is 7 clocks of the I2C clock pre-divider. 110b - 8 clocks. Minimum SCL low time is 8 clocks of the I2C clock pre-divider. 111b - 9 clocks. Minimum SCL low time is 9 clocks of the I2C clock pre-divider.

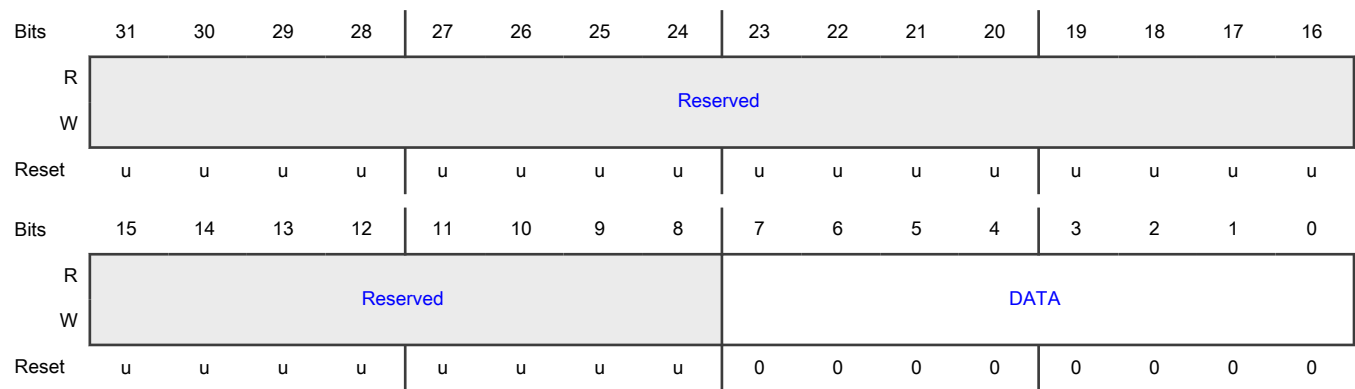
53.5.4.1.11 Master Data Register (MSTDAT)

Offset

Register	Offset
MSTDAT	828h

Function

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Diagram**Fields**

Field	Function
31-8	Reserved
—	Read value is undefined; only zero should be written.
7-0	Master function data register
DATA	• Read: read the most recently received data for the Master function. • Write: transmit data using the Master function.

53.5.4.1.12 Slave Control Register (SLVCTL)**Offset**

Register	Offset
SLVCTL	840h

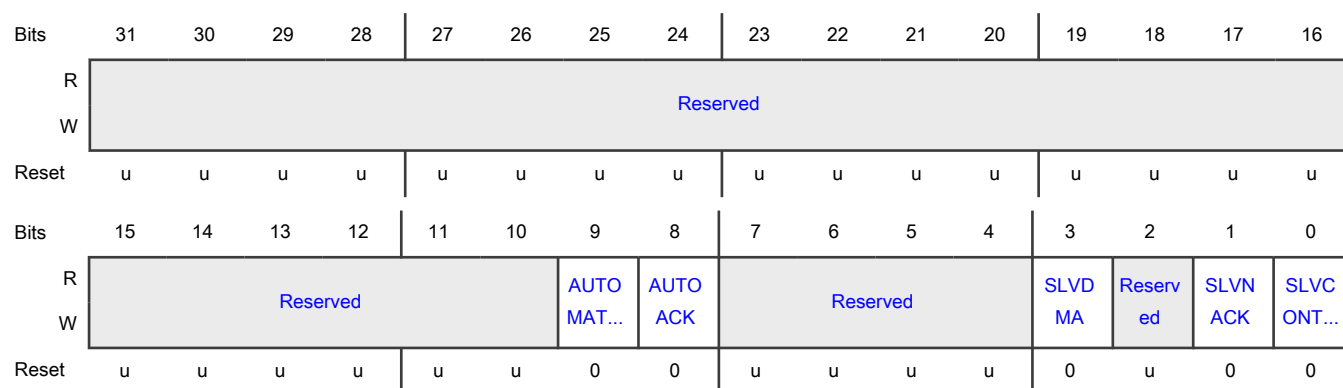
Function

The SLVCTL register contains bits that control various functions of the I2C Slave interface. Only write to the SLVCTL register when the slave is pending (STAT[SLVPENDING] = 1).

NOTE

When in the slave address state (slave state 0), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Diagram



Fields

Field	Function
31-10 —	Reserved Read value is undefined; only zero should be written.
9 AUTOMATCHREAD	Automatic Match Read When AUTOACK is set, AUTOMATCHREAD bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Because DMA needs to be configured to match the transfer direction, the direction needs to be specified. AUTOMATCHREAD bit allows a direction to be chosen for the next operation. 0b - In Automatic Mode, the expected next operation is an I2C write. 1b - In Automatic Mode, the expected next operation is an I2C read.
8 AUTOACK	Automatic Acknowledge - When AUTOACK bit is set, it will cause an I2C header that matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. - If AUTOACK bit is clear and a header matches SLVADR0, then the behavior is controlled by SLVADR0[AUTONACK], allowing NACK or interrupt. 0b - Normal, non-automatic operation. If AUTONACK = 0, then a SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, then received addresses are NACKed (ignored). 1b - A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, then the behavior will depend on the SLVADR0[AUTONACK] bit: if AUTONACK is set, then it will be Nacked; if AUTONACK is clear, then a SlvPending interrupt is generated.
7-4 —	Reserved Read value is undefined; only zero should be written.
3	Slave DMA enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
SLVDMA	0b - Disabled. No DMA requests are issued for Slave mode operation. 1b - Enabled. DMA requests are issued for I2C slave data transmission and reception.
2 —	Reserved Read value is undefined; only zero should be written.
1 SLVNACK	Slave NACK 0b - No effect 1b - NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (in Slave Receiver mode).
0 SLVCONTINUE	Slave Continue 0b - No effect 1b - Continue. Informs the Slave function to continue to the next operation, by clearing the STAT[SLVPENDING] flag. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.

53.5.4.1.13 Slave Data Register (SLVDAT)

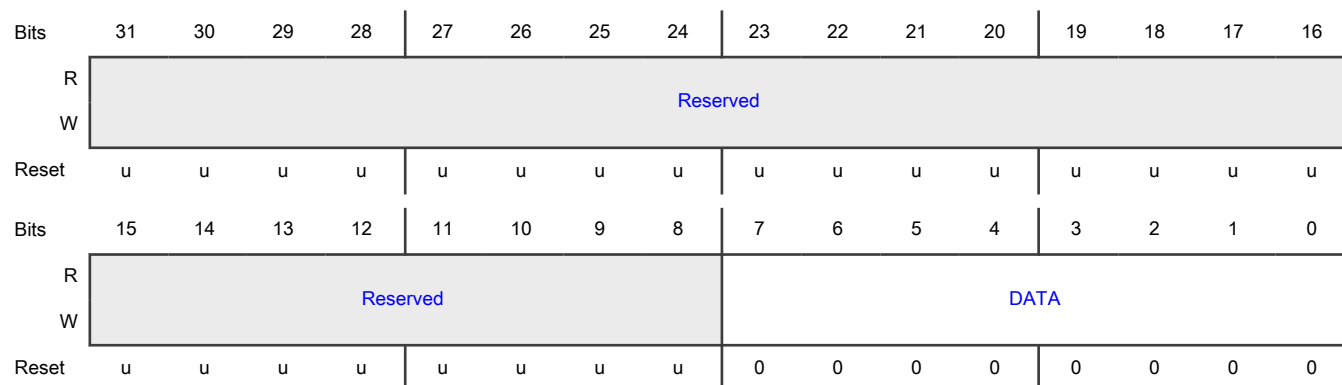
Offset

Register	Offset
SLVDAT	844h

Function

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Diagram



Fields

Field	Function
31-8 —	Reserved Read value is undefined; only zero should be written.
7-0 DATA	Slave function data register - Read: read the most recently received data for the Slave function. - Write: transmit data using the Slave function.

53.5.4.1.14 Slave Address Register (SLVADR0 - SLVADR3)

Offset

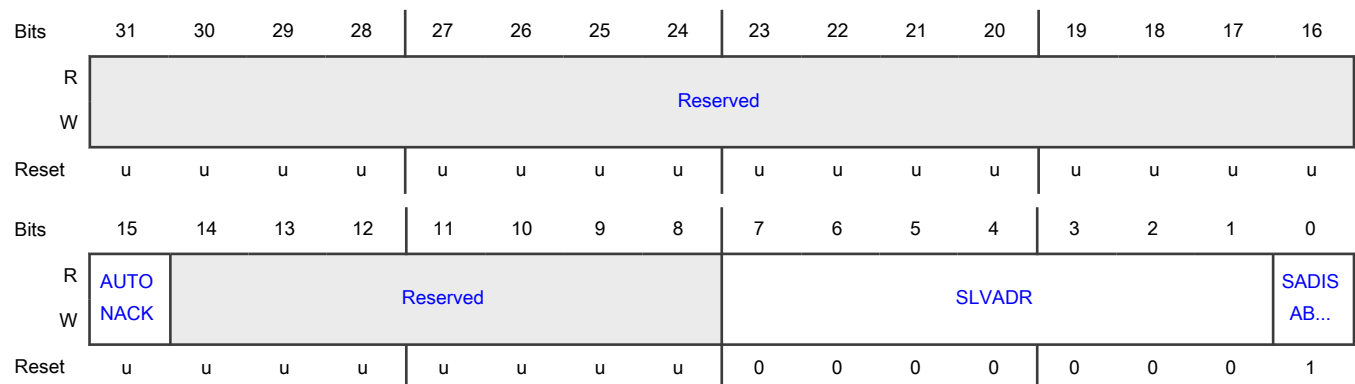
Register	Offset
SLVADR0	848h
SLVADR1	84Ch
SLVADR2	850h
SLVADR3	854h

Function

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware. The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register.

The slave address registers (SLVADR1 - SLVADR3) provide 3 additional addresses that can be automatically recognized by the I2C slave hardware. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Diagram



Fields

Field	Function
31-16 —	Reserved Read value is undefined; only zero should be written.
15 AUTONACK	Automatic NACK operation Used in conjunction with AUTOACK and AUTOMATCHREAD, AUTONACK allows software to ignore I2C traffic while handling previous I2C data or other operations. 0b - Normal operation, matching I2C addresses are not ignored. 1b - Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, and the address matches SLVADRn, and AUTOMATCHREAD matches the direction.
14-8 —	Reserved Read value is undefined; only zero should be written.
7-1 SLVADR	Slave Address. The 7-bit slave address that is compared to received addresses, if enabled.
0 SADISABLE	Slave Address n Disable 0b - Enabled. Slave Address n is enabled. 1b - Ignored. Slave Address n is ignored.

53.5.4.1.15 Slave Qualification for Address 0 Register (SLVQUAL0)

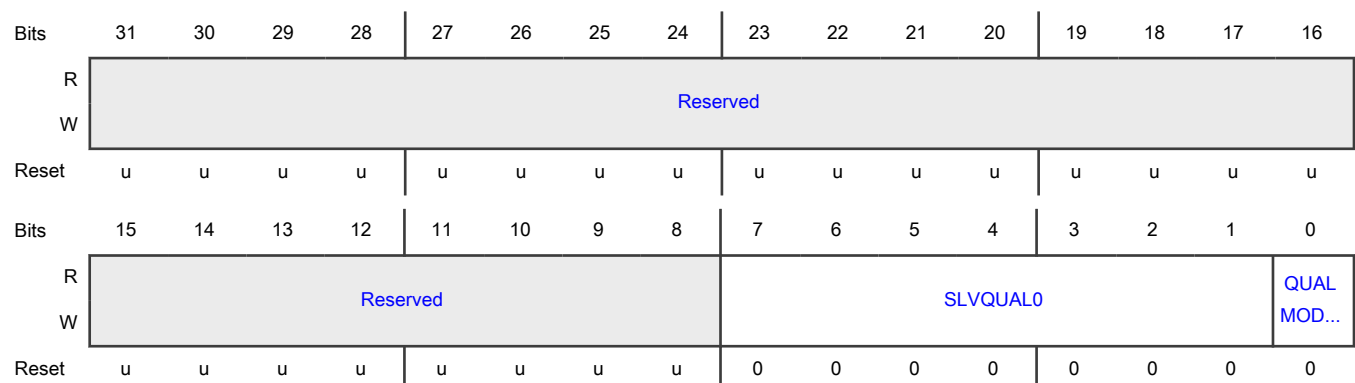
Offset

Register	Offset
SLVQUAL0	858h

Function

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Diagram



Fields

Field	Function
31-8 —	Reserved Read value is undefined; only zero should be written.
7-1 SLVQUAL0	Slave address Qualifier for address 0 SLVQUAL0=0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. - If QUALMODE0 = 0, then any bit in SLVQUAL0 field that is set to 1 will cause an automatic match of the corresponding bit of the received address when it (the address) is compared to the SLVADR0 register. - If QUALMODE0 = 1, then an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when $SLVADR0[7:1] \leq \text{received address} \leq SLVQUAL0[7:1]$).
0 QUALMODE0	Qualify mode for slave address 0 0b - Mask. The SLVQUAL0 field is used as a logical mask for matching address 0. 1b - Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.

53.5.4.1.16 Monitor Receiver Data Register (MONRXDAT)

Offset

Register	Offset
MONRXDAT	880h

Function

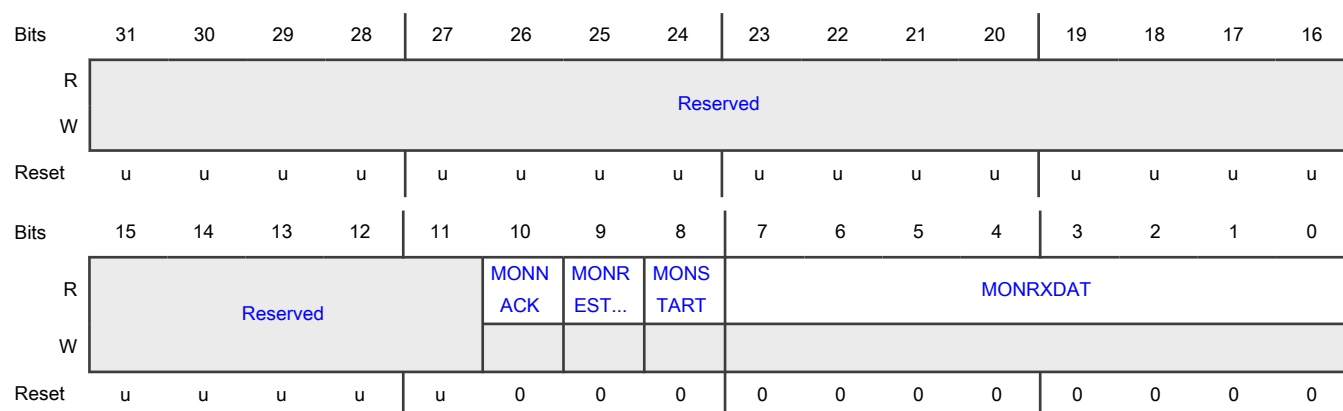
The read-only MONRXDAT register provides information about events on the I2C-bus, primarily to facilitate debugging of the I2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the CFG[MONEN] bit. Monitor mode can be configured to stretch the I2C clock if data is not read from the MONRXDAT register in time to prevent it, via the CFG[MONCLKSTR] bit. This can help ensure that nothing is missed, but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). To improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it (the monitor data) is available until the end of the next piece of information from the I2C-bus.

NOTE

Details of clock stretching are different in HS mode.

Diagram



Fields

Field	Function
31-11 —	Reserved Read value is undefined; only zero should be written.
10 MONNACK	Monitor Received NACK 0b - Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver. 1b - Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.
9 MONRESTART	Monitor Received Repeated Start 0b - No repeated start detected. The Monitor function has not detected a Repeated Start event on the I2C bus. 1b - Repeated start detected. The Monitor function has detected a Repeated Start event on the I2C bus.
8 MONSTART	Monitor Received Start 0b - No start detected. The Monitor function has not detected a Start event on the I2C bus. 1b - Start detected. The Monitor function has detected a Start event on the I2C bus.
7-0 MONRXDAT	Monitor function Receiver Data This reflects every data byte that passes on the I2C pins.

53.5.4.1.17 Peripheral Identification Register (ID)

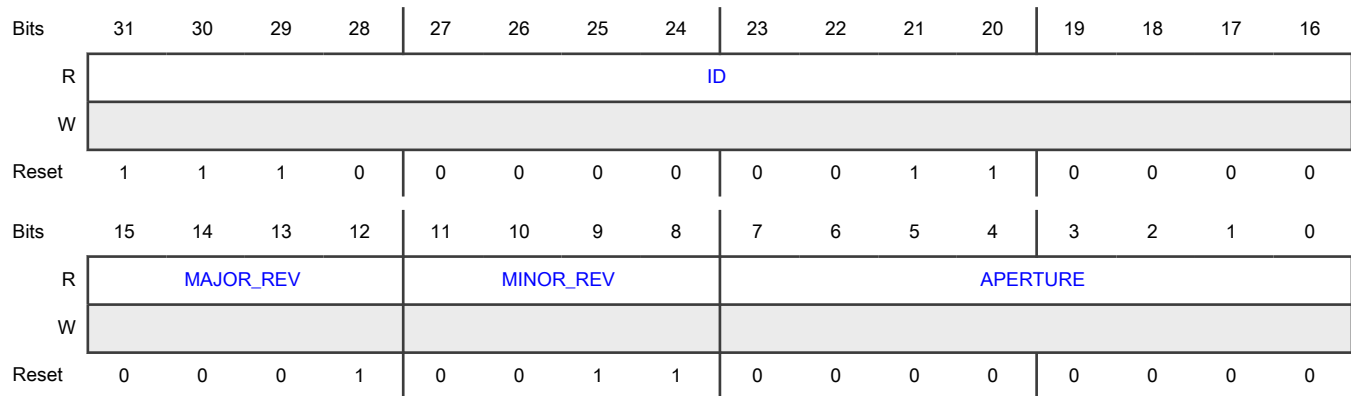
Offset

Register	Offset
ID	FFCh

Function

The ID register identifies the type and revision of the I2C module. A generic software driver can make use of this information register, to implement module type or revision-specific behavior.

Diagram



Fields

Field	Function
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8 MINOR_REV	Minor revision of module implementation
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

53.6 Inter-IC Sound (I2S)

53.6.1 Overview

The I²S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I²S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer.

The I²S function within one Flexcomm module provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm module share one set of I²S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration, and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, this is handled outside of the I²S block in the system level clocking scheme.

53.6.1.1 Block diagram

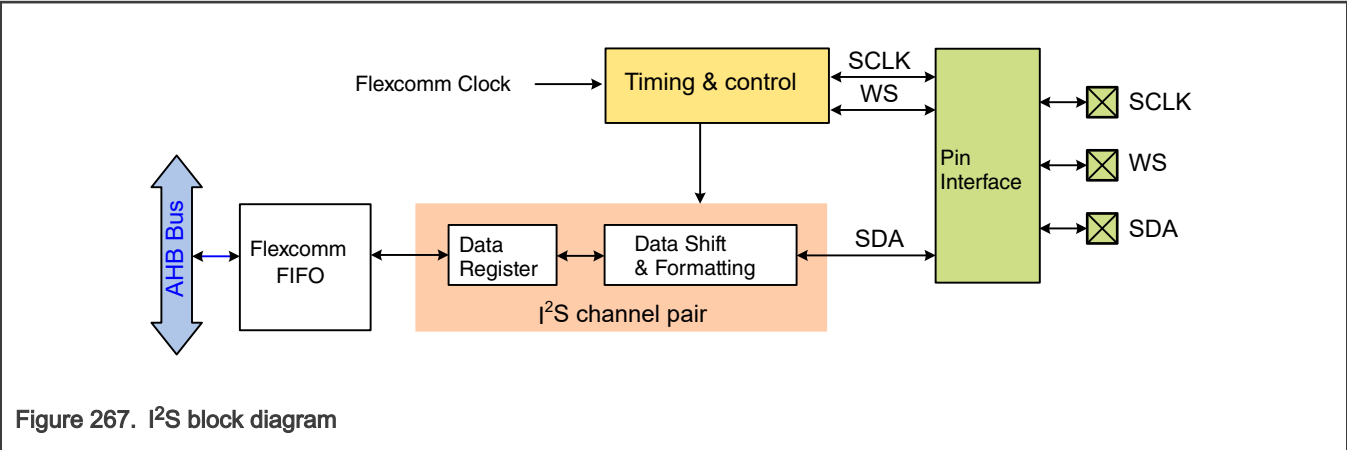


Figure 267. I²S block diagram

53.6.1.2 Features

- A Flexcomm module may implement one or more I²S channel pairs, the first of which could be a master or a slave, and the rest of which would be slaves.
 - All channel pairs are configured together for either transmit or receive and other shared attributes.
 - The number of channel pairs is defined for each Flexcomm module, and may be from 0 to 4.
- Configurable data size for all channels within one Flexcomm module, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- Signals configuration:
 - All channel pairs within one Flexcomm module share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
 - SLK and WS can be selectively inverted.
- Data for all I²S traffic within one Flexcomm module uses the Flexcomm module FIFO. The FIFO depth is 8 entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.
- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.
- Sampling frequencies depend on the specific device configuration and applications constraints (e.g. system clock frequency, PLL availability, etc.) but generally supports standard audio data rates.

53.6.1.3 Terminology

Table 561. List of some terminology used in I²S

Term	Description
Channel	Essentially, one piece of information on a single SDA line. In classic I²S, there is a single set of stereo data, which is 2 channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I²S: left and right. On a microcontroller, this is typically what is implemented in a single instance of an I²S module.

Table continues on the next page...

Table 561. List of some terminology used in I²S (continued)

Term	Description
Classic I ² S	This term refers to the original I ² S bus specification from Philips Semiconductors. That specification defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The many variations of I ² S that may be found have descended from this original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. DSP mode may be used in conjunction with TDM mode.
MCLK	Master Clock. In some I ² S systems, this is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial Clock. Sometimes referred to as BCK. This is a bit clock for data on the SDA line.
SDA	Serial Data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I ² S stream, typically each with the same slot length. For classic I ² S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In MONO mode, each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots to put more channels of data into a single stream. TDM mode may be used in conjunction with DSP mode or I ² S mode.
WS	Word Select. Sometimes called LRCLK. Distinguishes left versus right data in most single stereo formats. Used as a frame delimiter in DSP and TDM modes.

53.6.2 Functional description

53.6.2.1 Formats and modes

The format of data frames and WS is determined by several fields in the [CFG1 register](#) and the [CFG2 register](#). CFG1 and CFG2 together control the formatting of the data and the format of the frame which contains the data.

53.6.2.1.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I²S module. These fields plus the position of data for each channel pair, as determined by the CFG2[POSITION] field, define the main features of the frame.

- The CFG1[MODE] field defines the overall character of the frame.
- The CFG2[FRAMELEN] field defines the length of the data frame this I²S participates in. This field is minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- The CFG1[DATALEN] field defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by the CFG1[DATALEN] field.

The CFG1[DATALEN] field is also used in these ways:

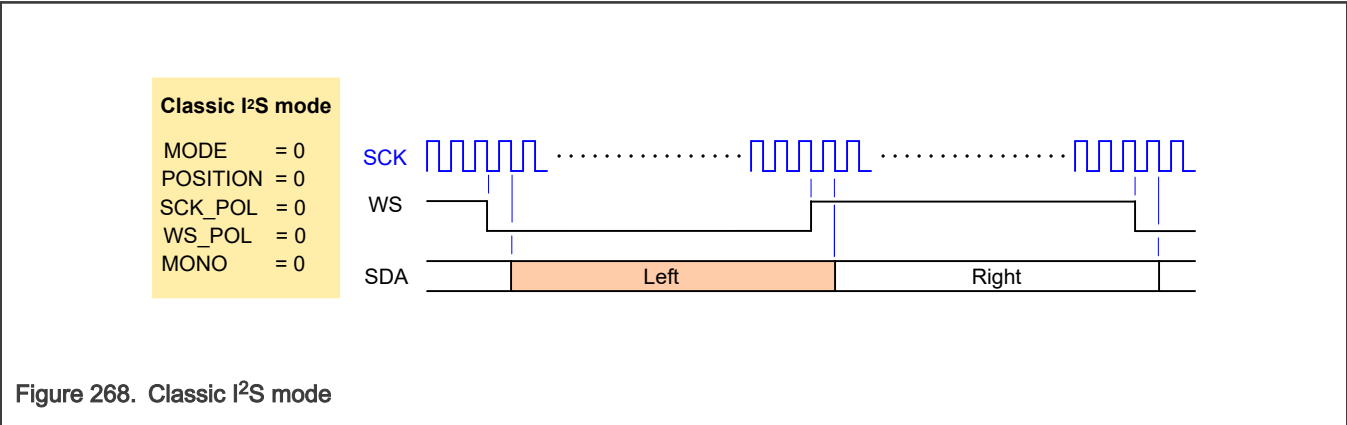
- Determines the size of data transfers between the FIFO and the I²S serializer/deserializer

- Determines the position of Right data following Left data within the frame, when the CFG1[MODE] field = 0x1, 0x2, or 0x3 (i.e. not 0x0).
- Determines the duration of the WS pulse, when the CFG1[MODE] field = 0x3.

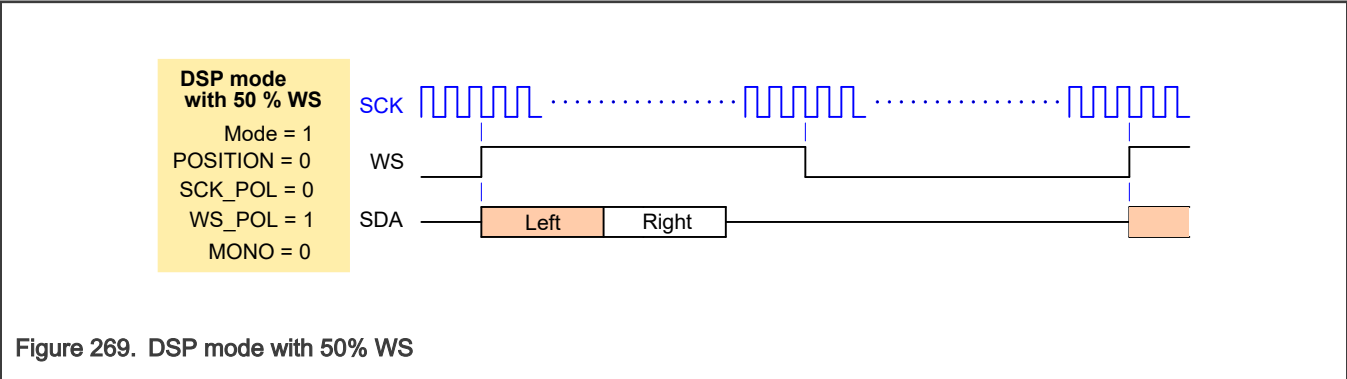
53.6.2.1.2 Example frame configurations

The following figures show a sampling of frame slot formats. These are not all of the possible formats, but shows the various frame formatting concepts. Note that slot identifications are examples only. Data positions are flexible and there are no predefined slots for the hardware.

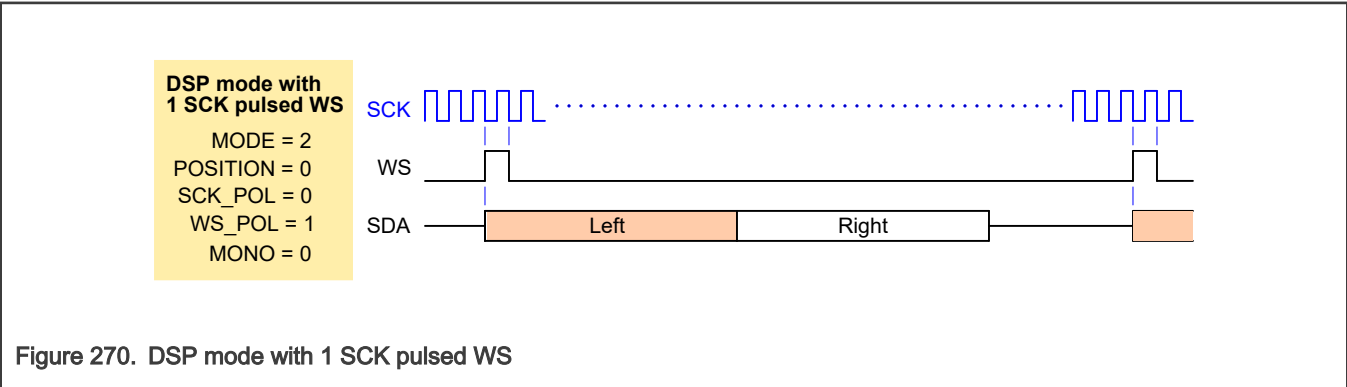
The first example shows the frame format for the Classic I²S mode. The data region (SDA signal) begins one clock after the leading WS edge for the frame.



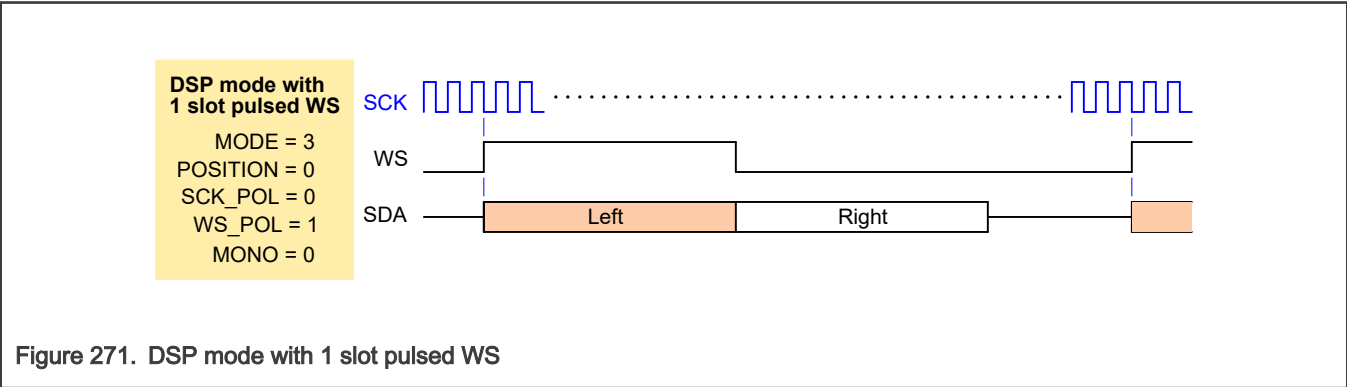
This frame format shows the DSP mode with 50% WS duty cycle and inverted WS (Word Select) polarity.



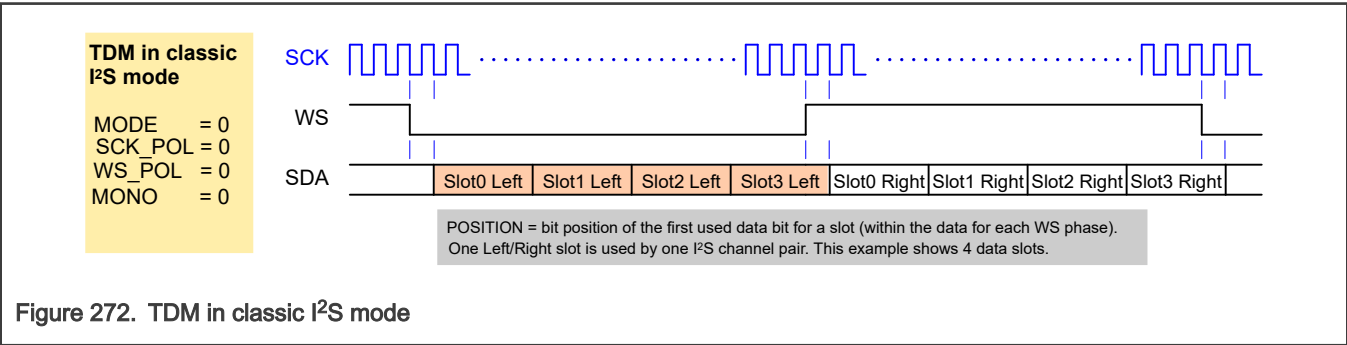
This frame format shows a DSP mode with 1 SCK pulsed WS. This is a DSP mode where WS (Word Select) polarity is inverted and the frame format has a one clock long pulse at the beginning of each data frame.



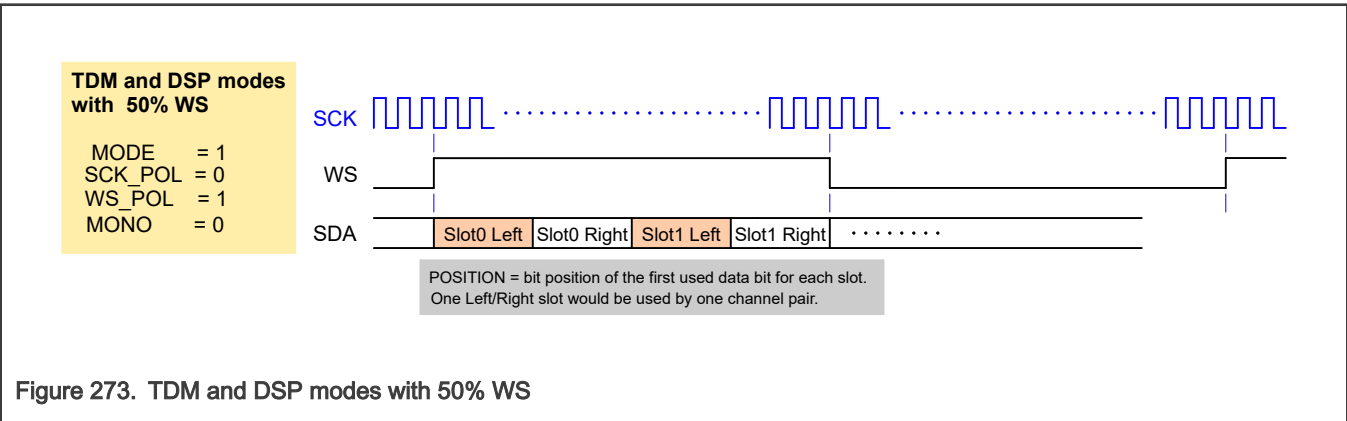
This frame format shows a DSP mode with 1 slot pulsed WS. This is a DSP mode where WS has a one data slot long pulse at the beginning of each data frame.



This frame format shows TDM in classic I²S mode.



This example shows the frame format for TDM and DSP modes with 50% WS. This is a TDM and DSP mode where WS has a 50% duty cycle and WS (Word Select) polarity is inverted.



This frame format shows TDM and DSP modes with 1 SCK pulsed WS. This is a TDM and DSP mode where WS (Word Select) polarity is inverted and the format has a one clock long pulse at the beginning of each data frame.

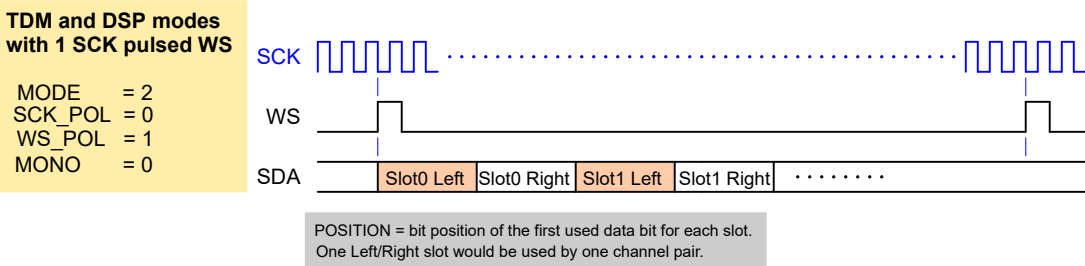


Figure 274. TDM and DSP modes with 1 SCK pulsed WS

This example shows the frame format for TDM and DSP modes with 1 slot pulsed WS. This is a TDM and DSP mode where WS has a one data slot long pulse at the beginning of each data frame. The frame format shows WS (Word Select) polarity is inverted.

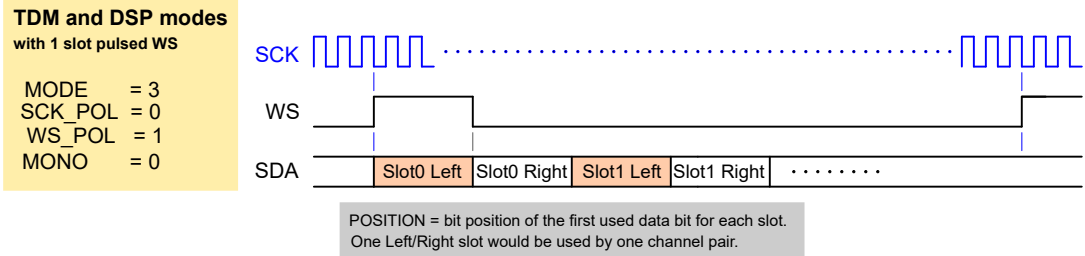


Figure 275. TDM and DSP modes with 1 slot pulsed WS

This example shows a I²S mode with mono or single channel selected.

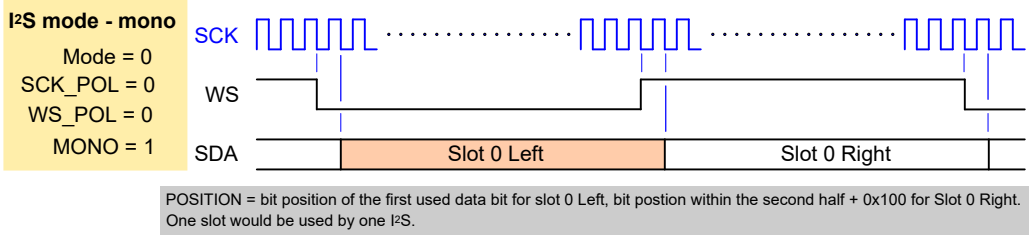


Figure 276. I²S mode, mono

This frame format shows DSP mode with mono or single channel selected. The frame format shows WS (Word Select) polarity is inverted.

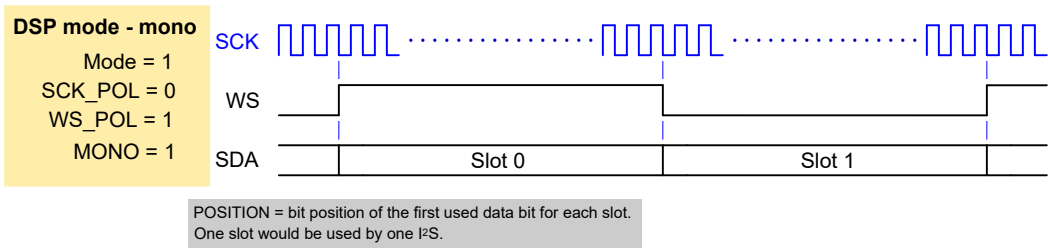


Figure 277. DSP mode, mono

This example shows TDM and DSP modes with mono or single channel and WS pulsed for one SCK time. This is a TDM and DSP mode where WS (Word Select) polarity is inverted and the format has a one clock long pulse at the beginning of each data frame.

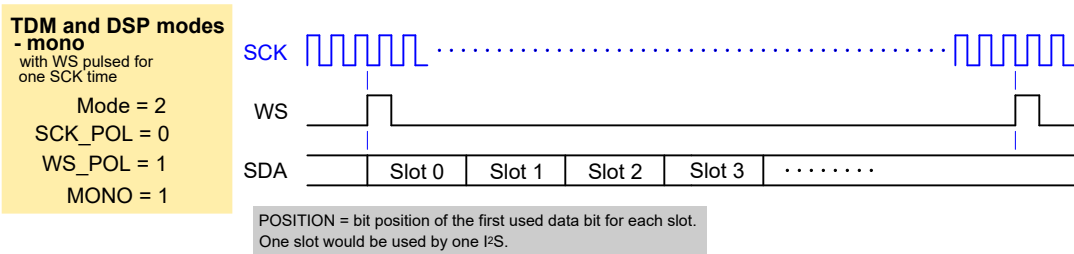


Figure 278. TDM and DSP modes, mono, with WS pulsed for one SCK time

53.6.2.1.3 I²S signal polarities

This diagram shows SCK (Serial Clock) and WS (Word Select) polarities and how they relate to data positions at the start of a frame.

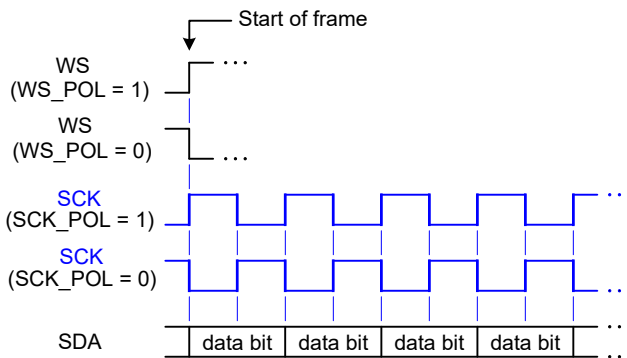


Figure 279. SCK and WS polarities

53.6.2.2 Data rates

53.6.2.2.1 Rate support

The actual I²S clock rates and sample rates that can be supported depend on the clocking. As a slave, the module receives SCK from a master. In that case, there is an upper limit to how fast the module can operate and a limit to how much data can be transferred across clock domains and handled by the CPU.

In general, the I²S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more.

53.6.2.2.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I²S is generally an integer multiple of:

- Frame/sample rate x number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I²S function divider can be used to produce the desired frequency.

Example 1

This I²S channel pair transfers stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup:

- Sample rate is 96 kHz.
- The frame is configured for two 32-bit data slots (32-bit stereo).
- The function clock divider output rate would be $96,000 \times (2 \times 32) = 6.144 \text{ MHz}$.

The value in the DIV register would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I²S channel pair supplies one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup:

- Sample rate is 50 kHz.
- The frame is configured for four 16-bit data slots.
- The function clock divider output rate would be $50,000 \times (4 \times 16) = 3.2 \text{ MHz}$.

The value in the DIV register would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

53.6.2.3 FIFO buffer configurations and usage

The Flexcomm module supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I²S CFG1[DATALEN] field, and other configuration bits as follows:

- If CFG1[DATALEN] specifies a number of data bits from 4 to 16:
 - The FIFO will be configured as 32 bits wide and 8 entries deep.
 - Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the CFG1[RIGHTLOW] configuration bit.
 - If a channel pair is configured with CFG1[ONECHANNEL] = 1, then only one value is transferred, nominally the left.
- If CFG1[DATALEN] specifies a number of data bits from 17 to 24:

- The FIFO will be configured as 48 bits wide and 8 entries deep.
- Data transfer between the bus and the FIFO depends the FIFOCFG[PACK48] configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with the FIFOWR or FIFORD register. When DMA is not enabled, transfers will alternate between the FIFOWR or FIFORD and FIFOWR48H or FIFORD48H registers, depending on the data direction selected for the I²S function. In all cases, the 2 transfers constitute a pair of left and right values. The order of left and right data is selectable via the CFG1[RIGHTLOW] configuration bit.
- If FIFOCFG[PACK48] = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder needed to complete the paired data as defined.
- If a channel pair is configured with CFG1[ONECHANNEL] = 1, then only the left value is transferred using the FIFOWR or FIFORD register.
- If CFG1[DATALEN] specifies a number of data bits from 25 to 32:
 - The FIFO will be configured as 32 bits wide and 8 entries deep.
 - Each data transfer between the bus and the FIFO is a single value, starting with left, then right.
 - If a channel pair is configured with CFG1[ONECHANNEL] = 1, then only one value is transferred.

53.6.2.4 DMA

The Flexcomm module can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA.

To enable DMA data transfers: .

- Configure I²S clocking
- Configure the channel
- Configure the DMA. To enable the DMA requests, set the bit for the related Flexcomm module instance in the INPUTMUX DMA controller request enable register. For some Flexcomm module instances, a DMA device selection must also be made in the DMA controller channel mux select register.

The I²S bus must be running to generate DMA requests.

DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm module I²S may be found in the [FIFO Configuration and Enable \(FIFOCFG\)](#) register [DMATX, DMARX, WAKETX, WAKERX, and PACK48 bits] and in the [FIFO Trigger Settings \(FIFOTRIG\)](#) register [TXLVLENA, RXLVLENA bits and fields TXLVL and RXLVL].

53.6.2.5 Clocking and power considerations

The master function of the I²S requires the Flexcomm clock (FCLK) to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

53.6.3 Signals

This table describes the module signals.

NOTE

When the I²S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOCON selection, in order for the I²S to operate.

Table 562. I²S Signals

Signal	I/O	Description
SCK	I/O	Serial Clock for I²S. Clock signal used to synchronize the transfer of data on the SDA signal. It is normally driven by the master and received by one or more slaves. NOTE When the primary I²S channel pair of a Flexcomm module is configured as a master, such that SCK is an output, it must actually be connected to a pin for the I²S to work properly.
WS	I/O	Word Select for I²S. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. NOTE When the primary I²S channel pair of a Flexcomm module is configured as a master, such that WS is an output, it must actually be connected to a pin for the I²S to work properly.
SDA	I/O	Serial Data for a single data stream used by one or more I ² S channel pairs of I ² S. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm module to locally generate SCK and/or WS. This clock is not created inside the I ² S block. If MCLK is supported as an input to the device, it can be routed to the I ² S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I ² S block and used to operate its functions.

53.6.4 Initialization

To initialize the I²S module:

1. Enable the clocks to the I²S registers depending on whether they function as master or slave.
2. Reset the Flexcomm module for the I²S module function.
3. Enable the I²S signals.
4. Enable I²S channel pair interrupts
5. If using DMA, enable the I²S channel pair master and slave functions to work with the system DMA controller.

See the chip-specific I²S information for configuration details.

53.6.4.1 Configure the I²S for wake-up

In Sleep mode, any I²S interrupt can wake up the part, provided that the interrupt is enabled. As long as the I²S clock is configured to be active in Sleep mode, the I²S can wake up the part independently of whether the I²S block is configured in master or slave mode.

In Deep-sleep mode, I²S clocks are normally turned off. However, if the I²S module is configured to use the MCLK input or as a slave receiving an external SCK, the I²S can generate an interrupt and wake up the device.

53.6.4.1.1 Wake-up from Sleep mode

- Configure the I²S for the desired mode.
- Enable the I²S interrupts on the chip.
- Any enabled I²S interrupt wakes up the part from Sleep mode.

53.6.4.1.2 Wake-up from Deep-sleep mode

- Configure the I²S for the desired mode. The selected function clock of the I²S (whether internal or externally sourced) must be running.
- Enable the I²S interrupts on the chip.
- Any enabled I²S interrupt wakes up the part from Deep-sleep mode.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to Deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Configure the I²S for the desired mode. The selected function clock of the I²S (whether internal or externally sourced) must be running.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related I²S interrupt on the chip.
- Enable the DMA interrupt in the chip.
- Enable the wake-up controls on the chip.

53.6.5 Memory map and register definition

This section includes the I²S module memory map and detailed descriptions of all registers.

NOTE

The bus interface to the I²S registers contained in the Flexcomm module support only word writes. Byte and halfword writes are not supported in conjunction with the I²S function.

53.6.5.1 I²S Interface register descriptions

53.6.5.1.1 I²S memory map

FLEXCOMM0.I²S base address: 4010_6000h
FLEXCOMM1.I²S base address: 4010_7000h
FLEXCOMM2.I²S base address: 4010_8000h
FLEXCOMM3.I²S base address: 4010_9000h
FLEXCOMM4.I²S base address: 4012_2000h
FLEXCOMM5.I²S base address: 4012_3000h
FLEXCOMM6.I²S base address: 4012_4000h
FLEXCOMM7.I²S base address: 4012_5000h
FLEXCOMM8.I²S base address: 4020_9000h
FLEXCOMM9.I²S base address: 4020_A000h
FLEXCOMM10.I²S base address: 4020_B000h
FLEXCOMM11.I²S base address: 4020_C000h
FLEXCOMM12.I²S base address: 4020_D000h
FLEXCOMM13.I²S base address: 4020_E000h

Offset	Register	Width (In bits)	Access	Reset value
C00h	Configuration Register 1 for the Primary Channel Pair (CFG1)	32	RW	See section
C04h	Configuration Register 2 for the Primary Channel Pair (CFG2)	32	RW	See section
C08h	Status Register for the Primary Channel Pair (STAT)	32	RW	See section
C1Ch	Clock Divider (DIV)	32	RW	See section
C20h	Configuration Register 1 for Channel Pair 1 (P1CFG1)	32	RW	See section
C24h	Configuration Register 2 for Channel Pair 1 (P2CFG1)	32	RW	See section
C28h	Status Register for Channel Pair 1 (PSTAT1)	32	R	See section
C40h	Configuration Register 1 for Channel Pair 2 (P1CFG2)	32	RW	See section
C44h	Configuration Register 2 for Channel Pair 2 (P2CFG2)	32	RW	See section
C48h	Status Register for Channel Pair 2 (PSTAT2)	32	R	See section
C60h	Configuration Register 1 for Channel Pair 3 (P1CFG3)	32	RW	See section
C64h	Configuration Register 2 for Channel Pair 3 (P2CFG3)	32	RW	See section
C68h	Status Register for Channel Pair 3 (PSTAT3)	32	R	See section
E00h	FIFO Configuration and Enable (FIFOCFG)	32	RW	See section
E04h	FIFO Status (FIFOSTAT)	32	RW	See section
E08h	FIFO Trigger Settings (FIFOTRIG)	32	RW	See section
E10h	FIFO Interrupt Enable Set and Read (FIFOINTENSET)	32	RW	See section
E14h	FIFO Interrupt Enable Clear and Read (FIFOINTENCLR)	32	RW	See section
E18h	FIFO Interrupt Status (FIFOINTSTAT)	32	R	See section
E20h	FIFO Write Data (FIFOWR)	32	W	See section
E24h	FIFO Write Data for Upper Data Bits (FIFOWR48H)	32	RW	See section
E30h	FIFO Read Data (FIFORD)	32	R	See section
E34h	FIFO Read Data for Upper Data Bits (FIFORD48H)	32	R	See section
E40h	FIFO Data Read with No FIFO Pop (FIFORDNOPOP)	32	R	See section
E44h	FIFO Data Read for Upper Data Bits with No FIFO Pop (FIFORD48HNOPOP)	32	R	See section
E48h	FIFO Size Register (FIFOSIZE)	32	R	See section
FFCh	I2S Module Identification (ID)	32	R	E090_1000h

53.6.5.1.2 Configuration Register 1 for the Primary Channel Pair (CFG1)

Offset

Register	Offset
CFG1	C00h

Function

The CFG1 register contains mode settings, most of which apply to all I²S channel pairs within one Flexcomm module. A few settings apply only to the primary channel pair, as noted.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												DATALEN			
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		WS_ POL	SCK_ POL	PDMD ATA	ONEC HAN...	LEFTJ UST	RIGHT LOW	MODE		MSTSLVCFG		PAIRCOUNT		DATA PAU...	MAINE NA...
W																
Reset	u	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21	Reserved
—	Read value is undefined, only zero should be written.
20-16 DATALEN	Data Length Data length minus 1 when encoded defines the number of data bits to be transmitted or received for all I²S channel pairs in this Flexcomm module. Note that data is only driven to or received from the SDA signal for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I²S: - Determines the size of data transfers between the FIFO and the I²S serializer/deserializer. See FIFO buffer configurations and usage - In CFG1[MODE] for settings 0x1, 0x2, and 0x3, DATALEN determines the location of right data following left data in the frame. - In CFG1[MODE] = 0x3 (where the WS signal has a one data slot long pulse at the beginning of each data frame), DATALEN determines the duration of the WS pulse. Values: 0x00 to 0x02 are not supported Settings 0x03 - 0x1F represent 4 bits to 32 bits length.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0_0011b - Data is 4 bits in length. 0_0100b - Data is 5 bits in length. 0_0111b - Data is 8 bits in length. 1_1110b - Data is 31 bits in length. 1_1111b - Data is 32 bits in length.
15-14 —	Reserved Read value is undefined, only zero should be written.
13 WS_POL	WS Polarity 0b - Not inverted. Data frames begin at a falling edge of the WS signal (standard for classic I²S). 1b - Inverted. The WS signal is inverted, resulting in a data frame beginning at a rising edge of the WS signal (standard for most 'non-classic' variations of I²S).
12 SCK_POL	SCK Polarity 0b - Falling edge. Data is launched on the SCK signal falling edges and sampled on the SCK signal rising edges (standard for I²S). 1b - Rising edge. Data is launched on the SCK signal rising edges and sampled on the SCK signal falling edges.
11 PDMDATA	PDM Data Selection PDMDATA controls the data source for I²S transmit, and cannot be set in Rx mode. PDMDATA only has an effect if the device includes a DMIC subsystem. See chip-specific information. 0b - Normal Operation. Data is transferred to or from the Flexcomm FIFO. 1b - DMIC subsystem. The data source is the DMIC subsystem. When PDMDATA = 1, only the primary channel pair can be used in this Flexcomm module. If ONECHANNEL = 1, only the PDM left data is used. the WS signal rate must match the Fs (sample rate) of the DMIC decimator. A rate mismatch will at some point cause the I²S to overrun or underrun.
10 ONECHANNEL	Single Channel Mode Applies to both transmit and receive. ONECHANNEL applies only to the first I²S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers. NOTE In mode 0 only, the right side of the frame begins at POSITION = 0x400 (the number is calculated from Max Frame length (2048/2). This is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side (e.g. 0x108 would place data starting at the 8th clock after the middle of the frame). In other modes, data for the single channel of data is placed at the clock defined by POSITION. 0b - Dual channel. I²S data for this channel pair is treated as left and right channels.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Single channel. I ² S data for this channel pair is treated as a single channel, functionally the left channel for this pair.
9 LEFTJUST	Left-Justify Data 0b - Right-justified. Data is transferred between the FIFO and the I²S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. This corresponds to right-justified data in the stream on the data bus. 1b - Left-justified. Data is transferred between the FIFO and the I²S serializer/deserializer left justified, i.e., starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. This corresponds to left-justified data in the stream on the data bus.
8 RIGHTLOW	Right Channel Low Right channel data is in the low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits or if PDMDATA = 1. Note that if ONECHANNEL = 1, then the one channel to be used is the left channel. CFG2[POSITION] can still place that data in the frame where right channel data is normally located. NOTE If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed. 0b - Right high. The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel. 1b - Right low. The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.
7-6 MODE	Mode Selects the basic I²S operating mode. Other configurations modify this setting to obtain all supported cases. See Formats and modes for examples. NOTE For a 50% WS duty cycle, FRAMELEN must define an even number of I²S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock occurs on the right. 00b - Classic Mode. The I²S mode where in each enabled channel pair the WS signal has a 50% duty cycle, with one piece of left channel data occurring during the first phase, and one piece of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS signal edge for the frame. See note for Mode with 50% duty cycle. 0. 01b - DSP mode WS 50% duty cycle. DSP mode where the WS signal has a 50% duty cycle. See note for Mode with 50% duty cycle. 0. 10b - DSP mode WS 1 clock. DSP mode where the WS signal has a one clock long pulse at the beginning of each data frame. 11b - DSP mode WS 1 data. DSP mode where the WS signal has a one data slot long pulse at the beginning of each data frame.
5-4	Master/Slave Configuration Selection

Table continues on the next page...

Table continued from the previous page...

Field	Function
MSTSLVCFG	MSTSLVCFG determines how the SCK and WS signals are used by all channel pairs in this Flexcomm module. 00b - Normal Slave Mode. Default mode. The SCK and WS signals are received from a master and used to transmit or receive data. 01b - WS Synchronized Master Mode. WS is received from another master and used to synchronize the generation of the SCK signal, when divided from the Flexcomm clock (FCLK). 10b - Master Using an Existing SCK Mode. The SCK signal is received and used directly to generate the WS signal, as well as transmitting or receiving data. 11b - Normal Master Mode. The SCK and WS signals are generated so they can be sent to one or more slave devices.
3-2 PAIRCOUNT	Pair Count Provides the number of I²S channel pairs in this Flexcomm module. This is a read-only field whose value may be different in other Flexcomm modules. 00b - One Pair. 1 I²S channel pairs in this Flexcomm module 01b - Two Pairs. 2 I²S channel pairs in this Flexcomm module 10b - Three Pairs. 3 I²S channel pairs in this Flexcomm module 11b - Four Pairs. 4 I²S channel pairs in this Flexcomm module
1 DATAPAUSE	Data Flow Pause Allows pausing data flow between the I²S serializer/deserializer and the FIFO. This could be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received. Once a data pause has been requested, the module may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. The check is done by monitoring the STAT[DATAPAUSED] flag. When DATAPAUSE is cleared, data transfer resumes at the beginning of the next frame. 0b - Normal operation. Normal operation, or resuming normal operation at the next frame if the I²S has already been paused. 1b - Pause. A pause in the data flow is being requested. The data pause is in effect when STAT[DATAPAUSED] = 1.
0 MAINENABLE	Main Enable Main enable for I²S function in this Flexcomm 0b - Disabled. All I²S channel pairs in this Flexcomm are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled. 1b - Enabled. This I²S channel pair is enabled. Other channel pairs in this Flexcomm may be enabled in their individual PAIRENABLE bits.

53.6.5.1.3 Configuration Register 2 for the Primary Channel Pair (CFG2)

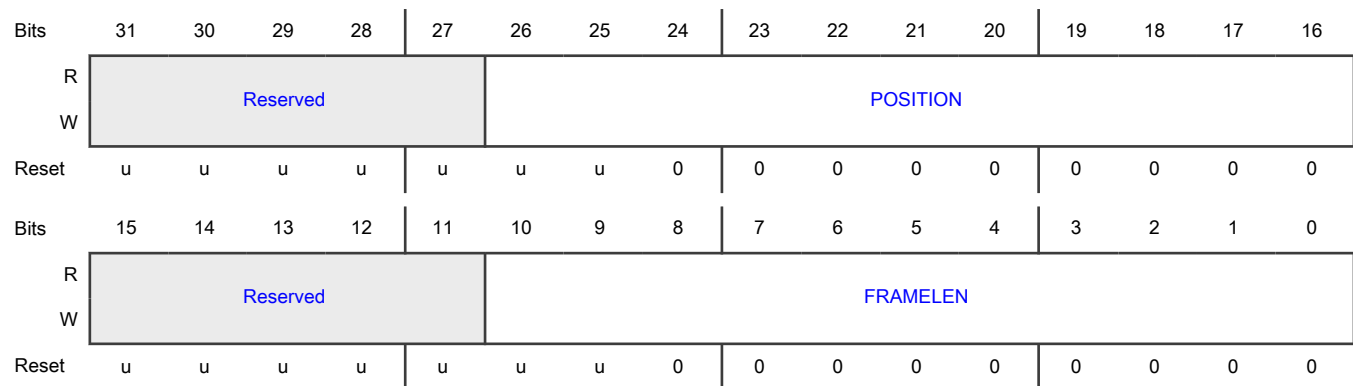
Offset

Register	Offset
CFG2	C04h

Function

The CFG2 register controls data configuration.

Diagram



Fields

Field	Function
31-27	Reserved
—	Read value is undefined, only zero should be written.
26-16 POSITION	Data Position Defines the location within the frame of the data for this channel pair. POSITION + CFG1[DATALEN] must be less than FRAMELEN. See Frame format. When CFG1[MODE] = 0x0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. CFG1[ONECHANNEL] = 1 while CFG1[MODE] = 0x0 is a special case, see the description of ONECHANNEL. The combination of CFG1[DATALEN] and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 000_0000_0000b - Data begins at bit position 0 (the first bit position) within the frame or WS phase 000_0000_0001b - Data begins at bit position 1 within the frame or WS phase 000_0000_0010b - Data begins at bit position 2 within the frame or WS phase
15-11	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined, only zero should be written.
10-0 FRAMELEN	Frame Length Frame length minus 1 when encoded defines the number of clocks and data bits in the frames that this channel pair participates in. See Frame format 0x000 to 0x002 are not supported NOTE If FRAMELEN defines an odd length frame (for example 33 clocks) in CFG1[MODE] = 0x0 or 0x1, the extra clock appears in the right half. NOTE When CFG1[MODE] = 0x3, FRAMELEN must be larger than DATALEN for the WS pulse to generate correctly. 000_0000_0011b - Frame is 4 bits in total length 000_0000_0100b - Frame is 5 bits in total length 001_1111_1111b - Frame is 512 bits in total length 111_1111_1111b - Frame is 2048 bits in total length

53.6.5.1.4 Status Register for the Primary Channel Pair (STAT)

Offset

Register	Offset
STAT	C08h

Function

The STAT register provides status flags for the I²S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only; some flags can be cleared by writing a 1 to them.

Diagram



Fields

Field	Function
31-4 —	Reserved Read value is undefined, only zero should be written.
3 DATAPAUSED	Data Paused The DATAPAUSED status flag applies to all I²S channels. 0b - Not Paused. Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the CFG1[DATA PAUSE] control bit. 1b - Paused. A data pause has been requested and is now in force.
2 LR	Left/Right Indication This LR flag is considered to be a debugging aid and is not expected to be used by an I²S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair. 0b - Left channel 1b - Right channel
1 SLVFRMERR	Slave Frame Error The SLVFRMERR flag applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between CFG2[FRAMELEN] and the actual incoming I²S stream. 0b - No error. No error has been recorded. 1b - Error. An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.
0 BUSY	Busy Status Busy status for the primary channel pair Other BUSY flags may be found in the STAT register for each channel pair.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Idle. The transmitter/receiver for channel pair is currently idle.
	1b - Busy. The transmitter/receiver for channel pair is currently processing data.

53.6.5.1.5 Clock Divider (DIV)

Offset

Register	Offset
DIV	C1Ch

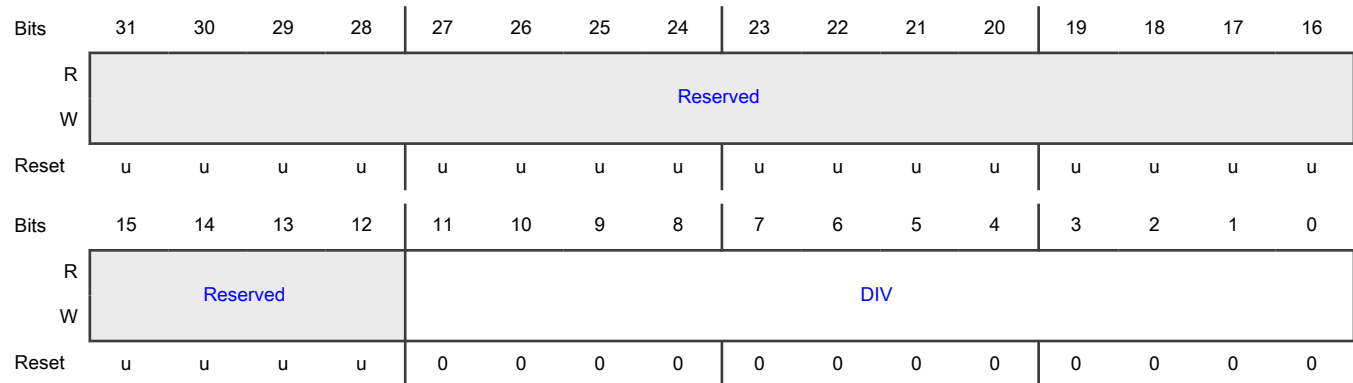
Function

The DIV register, used by all channel pairs, controls how the Flexcomm clock (FLCK) is used. See [Data rates](#) for more details.

NOTE

DIV must be set to 0 if the SCK signal is used as an input clock for the I²S function, which is the case when CFG1[MSTSLVCFG] is 0x1 or 0x2.

Diagram



Fields

Field	Function
31-12	Reserved
—	Read value is undefined, only zero should be written.
11-0	Divider
DIV	DIV controls how this I ² S block uses the Flexcomm clock (FLCK). The I ² S uses FCLK/(DIV + 1).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000_0000_0000b - FCLK is used directly. 0000_0000_0001b - FCLK is divided by 2. 0000_0000_0010b - FCLK is divided by 3. 1111_1111_1111b - FCLK is divided by 4,096.

53.6.5.1.6 Configuration Register 1 for Channel Pair a (P1CFG1 - P1CFG3)

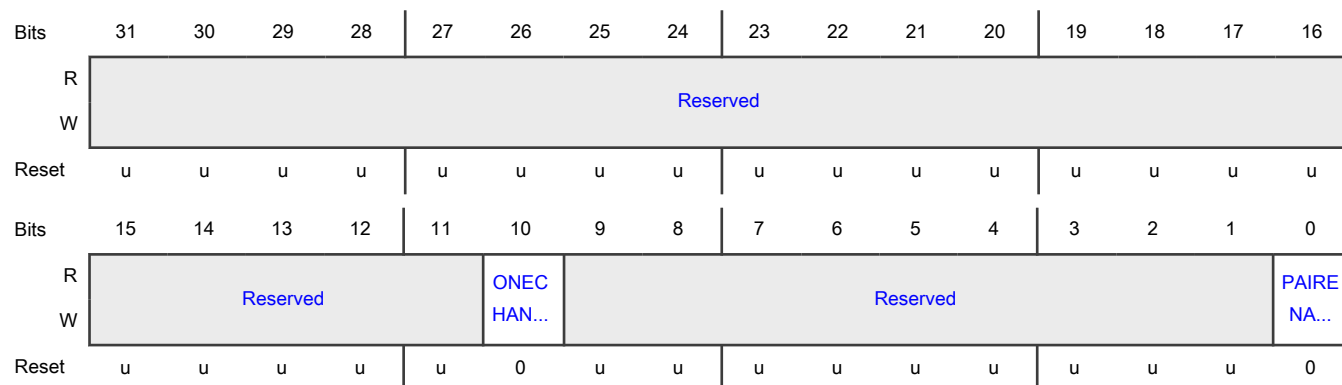
Offset

Register	Offset
P1CFG1	C20h
P1CFG2	C40h
P1CFG3	C60h

Function

The P1CFG1, P2CFG1, and P3CFG1 registers contain mode settings that apply to channel pairs other than the first pair, within the same Flexcomm module.

Diagram



Fields

Field	Function
31-11	Reserved
—	Read value is undefined, only zero should be written.
10	Single Channel Mode
ONECHANNEL	

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Dual Channel. I ² S data for this channel pair is treated as left and right channels. 1b - Single Channel. I ² S data for this channel pair is treated as a single channel, functionally the left channel for this pair.
9-1 —	Reserved Read value is undefined, only zero should be written.
0 PAIRENABLE	Pair Enable Enable for this channel pair 0b - Disabled. This I ² S channel pair is disabled. 1b - Enabled. This I ² S channel pair is enabled. Other channel pairs in this Flexcomm module may be enabled in their individual PAIRENABLE bits.

53.6.5.1.7 Configuration Register 2 for Channel Pair a (P2CFG1 - P2CFG3)

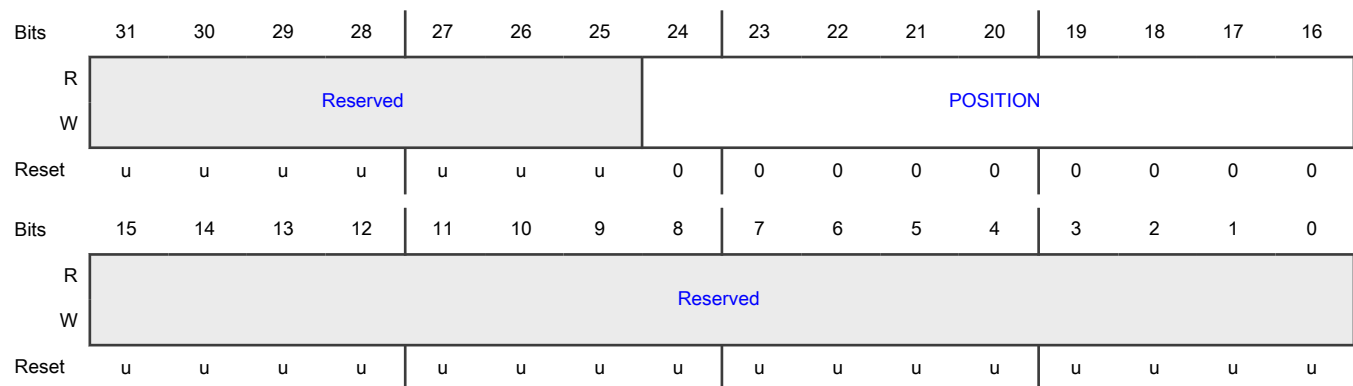
Offset

Register	Offset
P2CFG1	C24h
P2CFG2	C44h
P2CFG3	C64h

Function

These registers contain the frame position for channel pairs beyond the main pair.

Diagram



Fields

Field	Function
31-25 —	Reserved Read value is undefined, only zero should be written.
24-16 POSITION	Data Position Defines the location within the frame of the data for this channel pair. See details in the description of POSITION for the primary channel pair.
15-0 —	Reserved Read value is undefined, only zero should be written.

53.6.5.1.8 Status Register for Channel Pair a (PSTAT1 - PSTAT3)

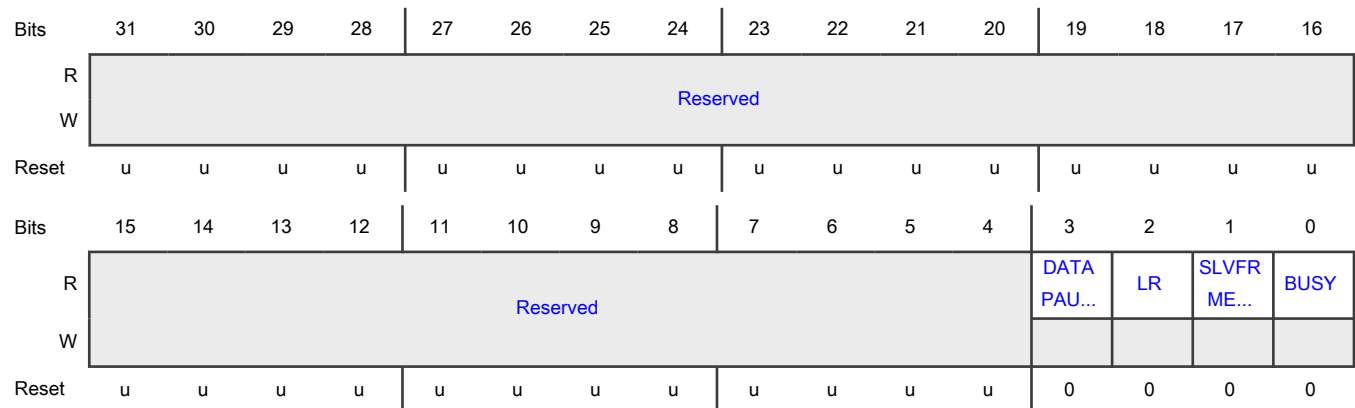
Offset

Register	Offset
PSTAT1	C28h
PSTAT2	C48h
PSTAT3	C68h

Function

These read-only registers provide status flags for additional channel pairs beyond the primary channel pair.

Diagram



Fields

Field	Function
31-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined, only zero should be written.
3 DATAPAUSED	Data Paused Status Flag This flag is shared by the STAT and PnSTAT registers. 0b - Data Not Paused. Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description in CFG1[DATAPAUSE]. 1b - Data Paused. A data pause has been requested and is now in force.
2 LR	Left/Right Indication This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details. 0b - Left channel 1b - Right channel
1 SLVFMERR	Save Frame Error Flag This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details. 0b - No Error. No error has been recorded. 1b - Error. An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.
0 BUSY	Busy Status for Channel Pair 0b - Idle. The transmitter/receiver for this channel pair is currently idle. 1b - Busy. The transmitter/receiver for this channel pair is currently processing data.

53.6.5.1.9 FIFO Configuration and Enable (FIFOCFG)

Offset

Register	Offset
FIFOCFG	E00h

Function

FIFOCFG configures FIFO use. A peripheral must be selected within the Flexcomm module prior to configuring the FIFO.

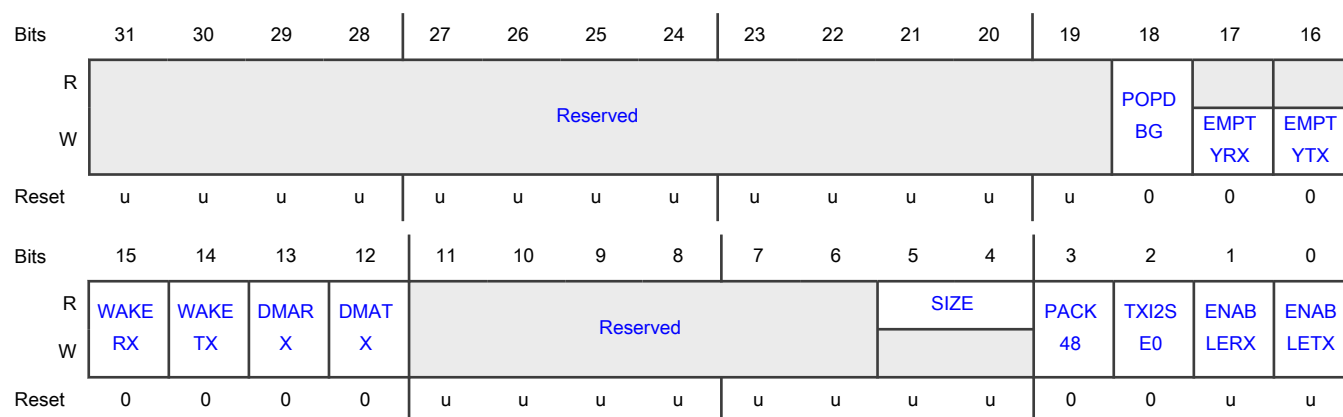
NOTE

Because all I²S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

NOTE

The FIFO for the selected I²S data direction must be enabled because the FIFO is the only means for accessing I²S data.

Diagram



Fields

Field	Function
31-19 —	Reserved Read value is undefined, only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0b - Debug reads of the FIFO do not pop the FIFO. 1b - A debug read causes the FIFO to pop.
17 EMPTYRX	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.
16 EMPTYTX	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU remains stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake-Up Control register in the chip if the feature is supported. 0b - Only enabled interrupts wake up the device from reduced power modes. 1b - A device wake-up for DMA occurs if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14 WAKETX	Wake-up for Transmit FIFO Level WAKETX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU remains stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake-Up Control register in the chip if the feature is supported.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled. Only enabled interrupts wake up the device from reduced power modes. 1b - Enabled. A device wake up for DMA occurs if the transmit FIFO level reaches the value specified by FIFOTRIG[TXLVL], even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Receive DMA configuration for receive 0b - Disabled. DMA is not used for the receive function. 1b - Enabled. Trigger DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Transmit DMA configuration for transmit 0b - Disabled. DMA is not used for the transmit function. 1b - Enabled. Trigger DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined, only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. 0x0, 0x1 are not applicable to I²S. 10b - Size 32 Bits. FIFO is configured as 8 entries of 32 bits, each corresponding to 2 16-bit data values for left and right channels. This setting occurs when the I²S CFG1[DATALEN] value is less than 16 bits, or from 25 to 32 bits. 11b - Size 48 Bits. FIFO is configured as 8 entries of 48 bits, each corresponding to 2 24-bit (?) data values for left and right channels. This setting occurs when the I²S CFG1[DATALEN] value is from 17 to 24 bits.
3 PACK48	Packing Format 48-bit data PACK48 relates to how data is entered into or taken from the FIFO by software or DMA. 0b - Bits_24. 48-bit I²S FIFO entries are handled as all 24-bit values. 1b - Bits_32_16. 48-bit I²S FIFO entries are handled as alternating 32-bit and 16-bit values.
2 TXI2SE0	Transmit I²S Empty 0 Transmit I²S Empty 0 Determines the value sent by the I²S in transmit mode if the transmit FIFO becomes empty. This value is sent repeatedly until the I²S is paused, the error is cleared, new data is provided, and the I²S is un-paused. 0b - Last value. If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when MONO = 1 for this channel pair.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Zero. If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.
1 ENABLERX	Enable Receive FIFO 0b - Disabled. The receive FIFO is not enabled. 1b - Enabled. The receive FIFO is enabled.
0 ENABLETX	Enable Transmit FIFO 0b - Disabled Transmit. The transmit FIFO is not enabled. 1b - Enabled transmit. The transmit FIFO is enabled.

53.6.5.1.10 FIFO Status (FIFOSTAT)

Offset

Register	Offset
FIFOSTAT	E04h

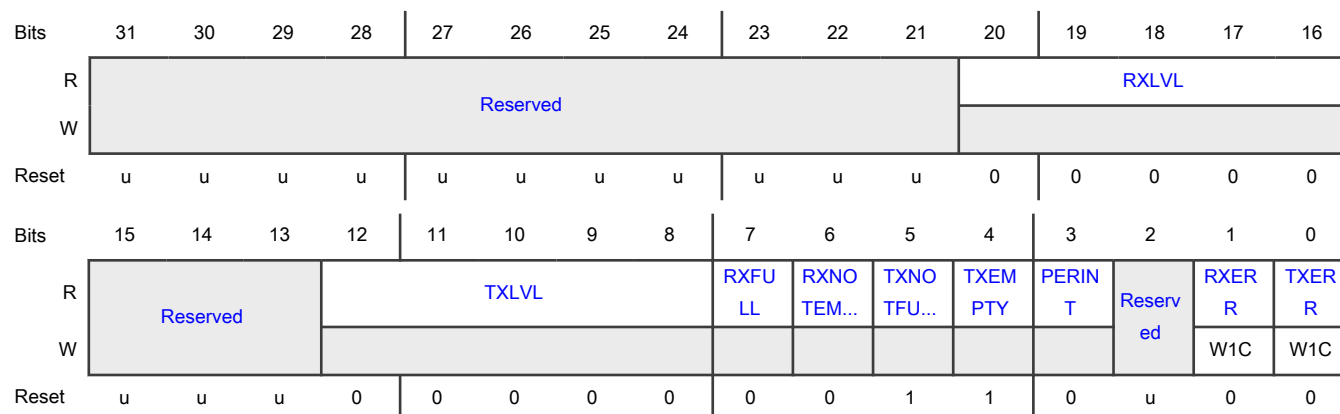
Function

FIFOSTAT provides status information for the FIFO and also indicates an interrupt from the peripheral function.

NOTE

Because all I²S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

Diagram



Fields

Field	Function
31-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags are 0. Other values tell how much data is in the RX FIFO at the point where the read occurs. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags are 1. 0_0000b - RX FIFO is empty
15-13 —	Reserved Read value is undefined, only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level A 0 means the transmit FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags are 1. Other values tell how much data is in the TX FIFO at the point where the read occurs. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags are 0. 0_0000b - TX FIFO is empty
7 RXFULL	Receive FIFO Full When 1, the receive FIFO is full. To prevent the peripheral from causing an overflow in the FIFO, data must be read out of the FIFO. 0b - Receive FIFO is not full 1b - Receive FIFO is full
6 RXNOTEMPTY	Receive FIFO Not Empty 0b - Receive FIFO is empty 1b - Receive FIFO is not empty, so data can be read.
5 TXNOTFULL	Transmit FIFO Not Full 0b - Transmit FIFO is full, and another write would cause an overflow 1b - Transmit FIFO is not full, so more data can be written
4 TXEMPTY	Transmit FIFO Empty 0b - Transmit FIFO is not empty 1b - Transmit FIFO is empty; however, the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register. 0b - No interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Interrupt
2	Reserved
—	Read value is undefined, only zero should be written.
1 RXERR	RX FIFO Error Set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit. 0b - No receive FIFO error occurred 1b - Receive FIFO error occurred
0 TXERR	TX FIFO Error Set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit. 0b - No transmit FIFO error occurred 1b - Transmit FIFO error occurred

53.6.5.1.11 FIFO Trigger Settings (FIFOTRIG)

Offset

Register	Offset
FIFOTRIG	E08h

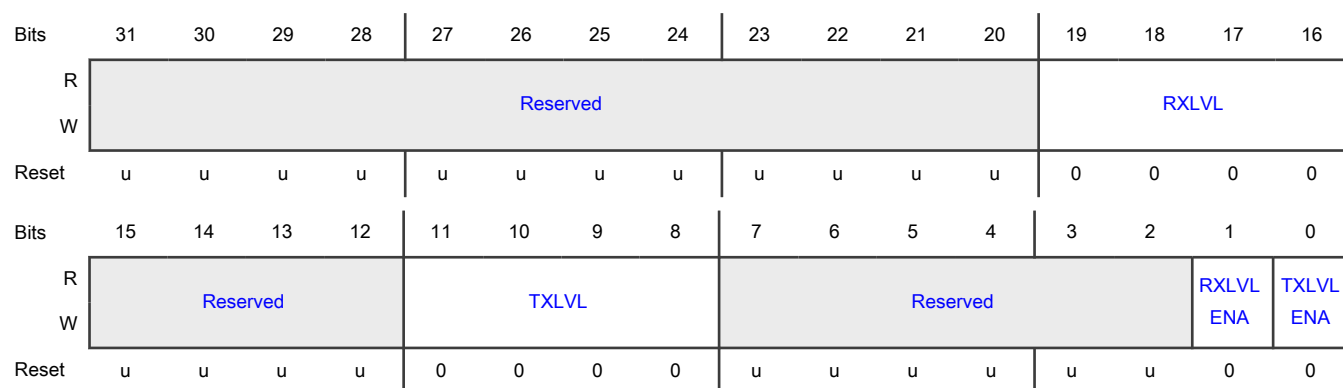
Function

The FIFOTRIG register allows selecting the trigger for FIFO-level related interrupts and DMA requests.

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

Diagram



Fields

Field	Function
31-20 —	Reserved Read value is undefined, only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. The RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. The RXLVL feature is enabled by the Hardware Wake Up Control register (if the chip supports it). NOTE FIFOCFG[WAKERX] allows the device to be woken from reduced power modes. 0000b - Trigger when the RX FIFO has received 1 entry (the FIFO is no longer empty). 0001b - Trigger when the RX FIFO has received 2 entries. 1111b - Trigger when the RX FIFO has received 16 entries (the FIFO has become full).
15-12 —	Reserved Read value is undefined, only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported NOTE FIFOCFG[WAKETX] allows the device to be woken from reduced power modes. 0000b - Trigger when the TX FIFO becomes empty. 0001b - Trigger when the TX FIFO level decreases to one entry. 1111b - Trigger when the TX FIFO level decreases to 15 entries (is no longer full).

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-2	Reserved
—	Read value is undefined, only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable This trigger becomes an interrupt if enabled in FIFOINTENSET[RXLVL], or a DMA trigger if FIFOCFG[DMARX] = 1. 0b - Receive FIFO level does not generate a FIFO level trigger. 1b - An trigger generates if the receive FIFO level reaches the value specified by the RXLVL.
0 TXLVLENA	Transmit FIFO Level Trigger Enable This trigger becomes an interrupt if enabled in FIFOINTENSET[TXLVL], or a DMA trigger if FIFOCFG[DMATX] = 1. 0b - Transmit FIFO level does not generate a FIFO level trigger. 1b - An trigger generates if the transmit FIFO level reaches the value specified by the TXLVL field in this register.

53.6.5.1.12 FIFO Interrupt Enable Set and Read (FIFOINTENSET)

Offset

Register	Offset
FIFOINTENSET	E10h

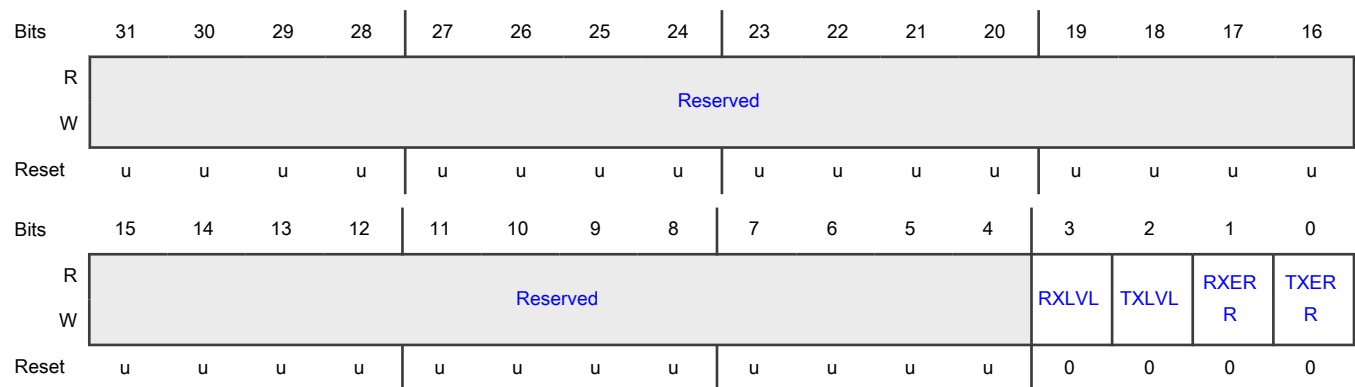
Function

The FIFOINTENSET register enables various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in the FIFOINTENSET register causes those bits to be set. The FIFOINTENCLR register can clear bits in this FIFOINTENSET register.

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Diagram



Fields

Field	Function
31-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive Level Interrupt Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[RXLVL] flag. 0b - Disabled. No interrupt generates based on the RX FIFO level. 1b - Enabled. If FIFOTRIG[RXLVLENA] = 1, an interrupt generates when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL].
2 TXLVL	Transmit Level Interrupt Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] flag. 0b - Disabled. No interrupt generates based on the TX FIFO level. 1b - Enabled. If FIFOTRIG[TXLVLENA] = 1, an interrupt generates when the TX FIFO level decreases to the level specified by the FIFOTRIG[TXLVL] flag.
1 RXERR	Receive Error Interrupt Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag. 0b - Disabled. No interrupt generates for a receive error. 1b - Enabled. An interrupt generates when a receive error occurs.
0 TXERR	Transmit Error Interrupt Determines whether an interrupt occurs when a transmit error occurs, based on the FIFOSTAT[TXERR] flag. 0b - Disabled. No interrupt generates for a transmit error. 1b - Enabled. An interrupt generates when a transmit error occurs.

53.6.5.1.13 FIFO Interrupt Enable Clear and Read (FIFOINTENCLR)

Offset

Register	Offset
FIFOINTENCLR	E14h

Function

The FIFOINTENCLR register clears interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												RXLVL	TXLVL	RXERR	TXERR
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0

Fields

Field	Function
31-4	Reserved
—	Read value is undefined; only zero should be written.
3 RXLVL	Receive Level Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0b - Interrupt is not cleared. 1b - Interrupt is cleared.
2 TXLVL	Transmit Level Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0b - Interrupt is not cleared. 1b - Interrupt is cleared.
1 RXERR	Receive Error Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0b - Interrupt is not cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Interrupt is cleared.
0 TXERR	Transmit Error Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0b - Interrupt is not cleared. 1b - Interrupt is cleared.

53.6.5.1.14 FIFO Interrupt Status (FIFOINTSTAT)

Offset

Register	Offset
FIFOINTSTAT	E18h

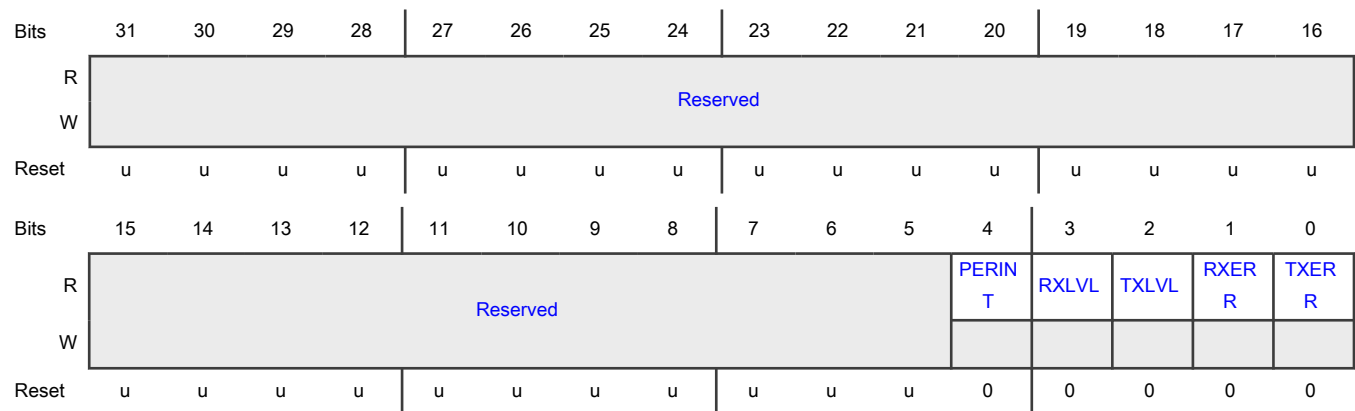
Function

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are pending. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in [FIFOSTAT](#) and [FIFOTRIG](#) for details.

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Diagram



Fields

Field	Function
31-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0b - Not pending 1b - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0b - Not pending 1b - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0b - Not pending 1b - Pending
1 RXERR	RX FIFO Error Interrupt Status 0b - Not pending 1b - Pending
0 TXERR	TX FIFO Error Interrupt Status 0b - Not pending 1b - Pending

53.6.5.1.15 FIFO Write Data (FIFOWR)

Offset

Register	Offset
FIFOWR	E20h

Function

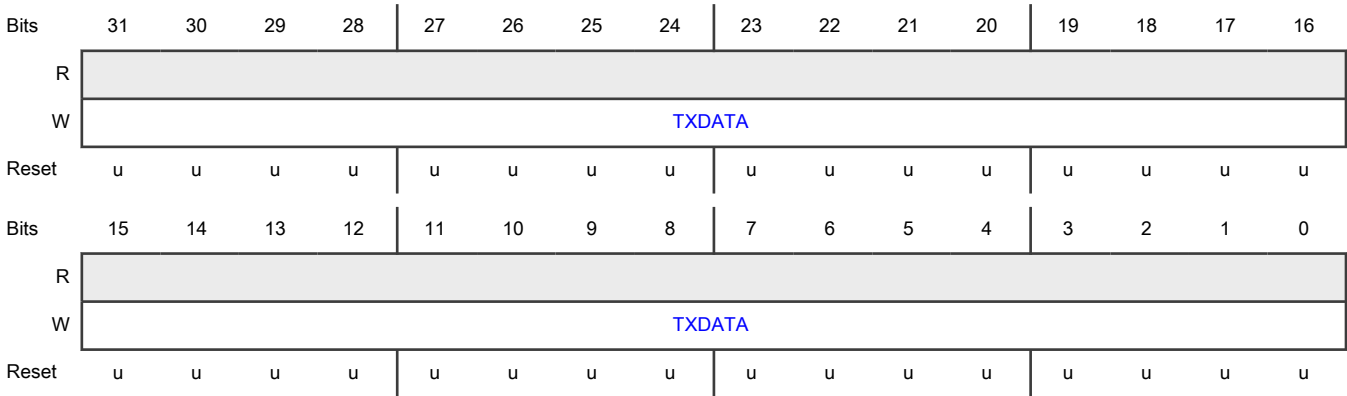
The FIFOWR register is used to write values to be transmitted to the FIFO.

Details of how FIFOWR and FIFOWR48H are used can be found in [FIFO buffer configurations and usage](#).

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Diagram



Fields

Field	Function
31-0 TXDATA	Transmit Data to the FIFO The number of bits used depends on configuration details.

53.6.5.1.16 FIFO Write Data for Upper Data Bits (FIFOWR48H)

Offset

Register	Offset
FIFOWR48H	E24h

Function

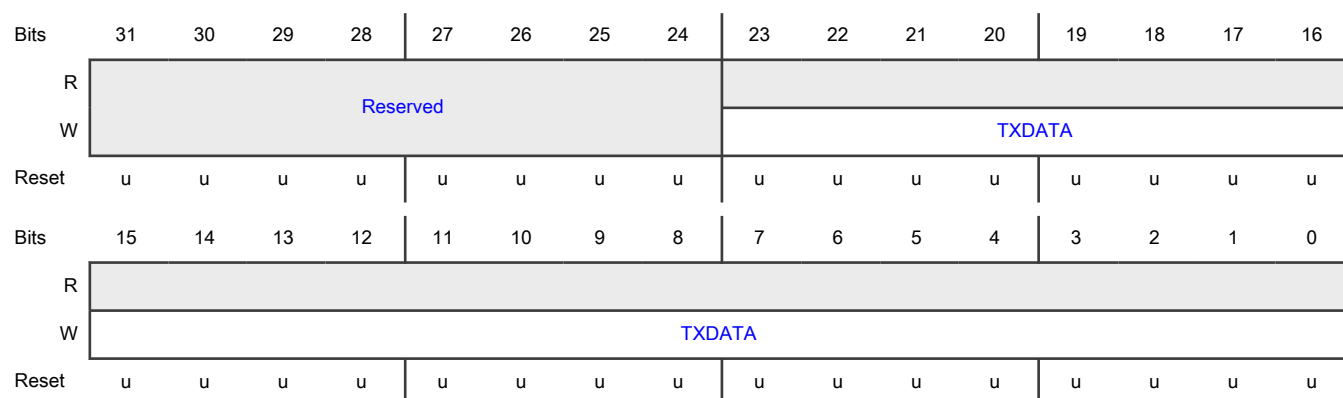
The FIFOWR48H register may only be used if the I²S is configured for 2x 24-bit data and not using DMA.

>

Details of how FIFOWR and FIFOWR48H are used can be found in [FIFO buffer configurations and usage](#).

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Diagram**Fields**

Field	Function
31-24	Reserved
—	Read value is undefined, only zero should be written.
23-0	Transmit Data to the FIFO
TXDATA	Whether the FIFOWR48H register is used and the number of bits used depends on configuration details.

53.6.5.1.17 FIFO Read Data (FIFORD)**Offset**

Register	Offset
FIFORD	E30h

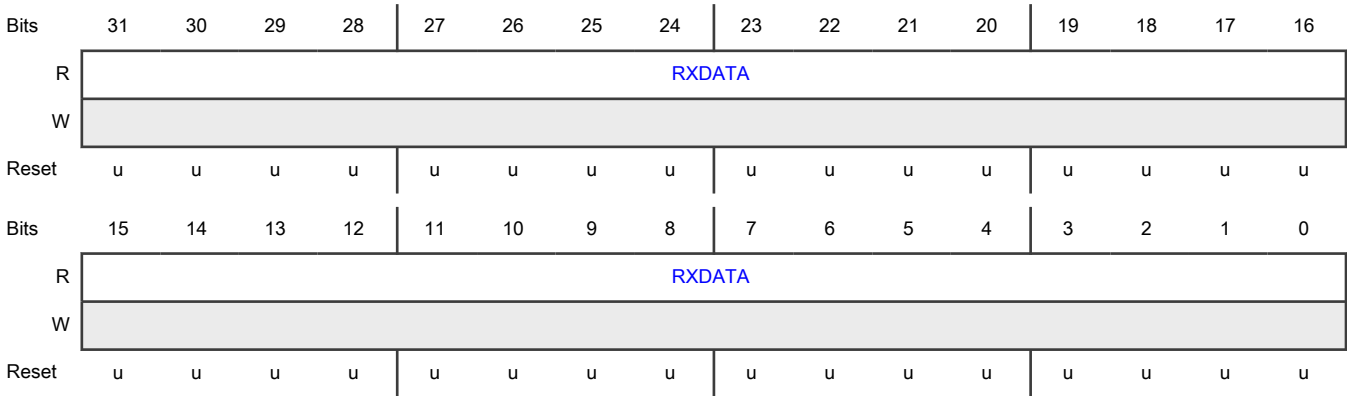
Function

The FIFORD register is used to read values that have been received by the FIFO. Details of how FIFORD and FIFORD48H are used can be found in [FIFO buffer configurations and usage](#)

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Diagram



Fields

Field	Function
31-0 RXDATA	Received Data from the FIFO The number of bits used depends on configuration details.

53.6.5.1.18 FIFO Read Data for Upper Data Bits (FIFORD48H)

Offset

Register	Offset
FIFORD48H	E34h

Function

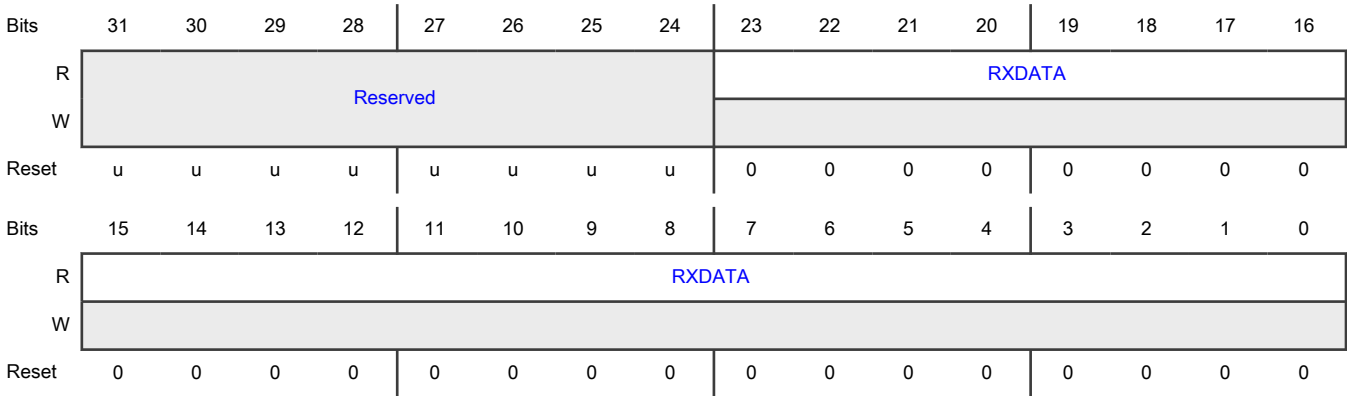
The FIFORD48H register may only be used if the I²S is configured for 2x 24-bit data and is not using DMA.

The FIFORD48H register is used under certain conditions to read values from the FIFO. Details of how FIFORD and FIFORD48H are used can be found in [FIFO buffer configurations and usage](#).

NOTE

Since all I²S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Diagram



Fields

Field	Function
31-24	Reserved
—	Read value is undefined, only zero should be written.
23-0	Received Data from the FIFO
RXDATA	Whether the FIFORD48H register is used and the number of bits used depends on configuration details.

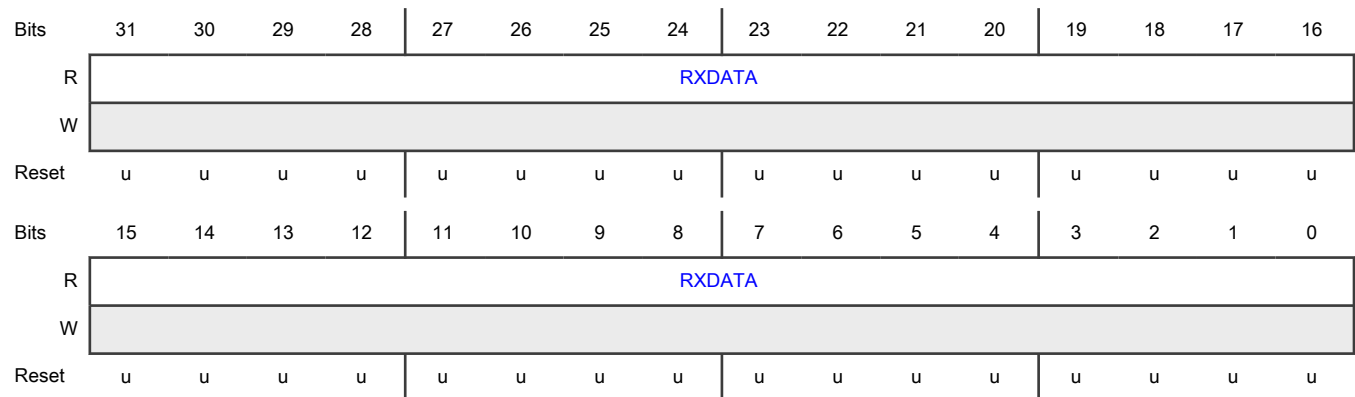
53.6.5.1.19 FIFO Data Read with No FIFO Pop (FIFORDNOPOP)

Offset

Register	Offset
FIFORDNOPOP	E40h

Function

The FIFORDNOPOP register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e., leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Diagram**Fields**

Field	Function
31-0 RXDATA	Received Data from the FIFO The number of bits used depends on configuration details.

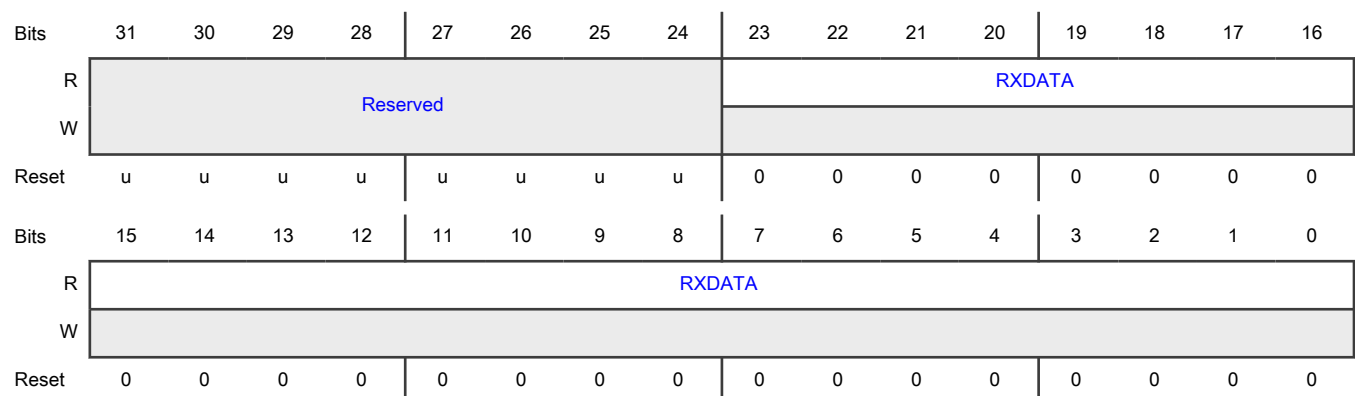
53.6.5.1.20 FIFO Data Read for Upper Data Bits with No FIFO Pop (FIFORD48HNOPOP)**Offset**

Register	Offset
FIFORD48HNOPOP	E44h

Function

May only be used if the I²S is configured for 2x 24-bit data and is not using DMA.

The FIFORD48HNOPOP register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e., leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Diagram

Fields

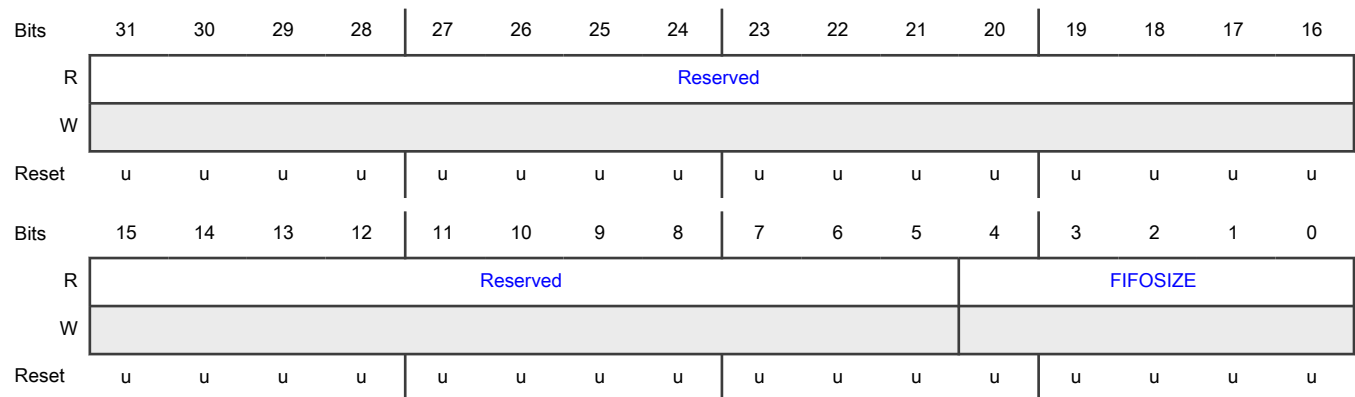
Field	Function
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 RXDATA	Received Data from the FIFO Whether the FIFORD48HNOPOP register is used and the number of bits used depends on configuration details.

53.6.5.1.21 FIFO Size Register (FIFOSIZE)**Offset**

Register	Offset
FIFOSIZE	E48h

Function

Provides the FIFO size.

Diagram**Fields**

Field	Function
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 8 entries for this chip. This field is read only.

53.6.5.1.22 I2S Module Identification (ID)

Offset

Register	Offset
ID	FFCh

Function

The ID register identifies the type and revision of the module. A generic software driver can use this information to implement specific behavior, based on a module type or a specific revision of that module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ID															
W																
Reset	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR_REV				MINOR_REV				APERTURE							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 ID	Module Identifier Unique module identifier for this IP block
15-12 MAJOR_REV	Major Revision Major revision of module implementation, starting at 0
11-8 MINOR_REV	Minor Revision Minor revision of module implementation, starting at 0
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

Chapter 54

Universal Serial Bus 2.0 HS Host Controller

54.1 Chip-specific USB2.0 HS Host Controller information

Table 563. Reference links to related information

Topic	Related module	Reference
Full description	USBHSH	USB2.0 HS Host
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

This chip does not support low speed mode.

54.1.1 Initialization

Initial configuration of the USB high speed host controller can be accomplished as follows:

- Enable the USB high speed host controller clocks by setting the USBHS_HOST_CLK and USBHS_SRAM bits in the PSCCTL0 register (See CLKCTL0 register descriptions).
- Select a clock source for the USB high speed host controller using the USBHSFCLKSEL register (See CLKCTL0 register descriptions).
- Select a clock divide for the USB high speed host controller using the USBHSFCLKDIV register (See CLKCTL0 register descriptions). To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 60 MHz.
- Clear the USB high speed host controller and USB RAM peripheral resets in the PRSTCTL0 register by writing to the PRSTCTL0_CLR register (See RSTCTL0 register descriptions).
- Enable the PDRUNCFG1[USBHS_SRAM_APD] and PDRUNCFG1[USBHS_SRAM_PPD] bits in to power-up the USB host controller (See SYSCTL).
- The High-speed USB device/host controller interrupt and USB activity interrupt are available to the NVIC (See NVIC). To allow the USB activity interrupt interrupts to wake-up the device from deep-sleep mode, enable this in the STARTEN1 register (See SYSCTL0 register descriptions).
- Configure the USB wake-up signal (See USB HS Device Controller).
- Use the IOPCTL to connect the (non-PHY) USB high speed host controller functions to external pins. (See IOPCTL).
- Port Control configuration
 - Enable port control configuration by setting the USB host clock control in the PSCCTL0 register (See CLKCTL0 register descriptions).
 - Set PORT_MODE[DEV_ENABLE] bit 16 to 1 to ensure that the port is routed to USB host controller.
 - To save power, disable the USB clocks described above.

54.2 Overview

The USB high-speed controller provides a plug-and-play connection of peripheral devices to a host with two different data speeds: high-speed with a data rate of 480 Mbps and full-speed with a data rate of 12 Mbps. Many portable devices can benefit from the

ability to communicate to each other over the USB interface without intervention of a host PC. The USB module contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB is not in use.

This chapter describes the host functionality of the controller.

54.2.1 Features

- Supports all high-speed and full-speed USB-compliant peripherals.
- Complies with *Universal Serial Bus specification 2.0*.
- Supports a hardware/software interface similar to the *Enhanced Host Controller Interface* (EHCI) specification.
- Supports USB 2.0 extension LPM mode.
- Supports port power switching.
- Supports power management.
- Integrated DMA engine can be used together with the audio PLL for USB streaming applications.
- On-chip USB PHY.

54.3 Signals

Table 564. USB Host pin description

Name	I/O	Description
USB_VBUS	I	USB Power.
USB_DP	I/O	Positive differential data.
USB_DM	I/O	Negative differential data.
USB_OVERCURRENTN	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry required to detect over-current condition).
USB_PORTPWRN	O	VBUS drive signal (towards external charge pump or power management unit); indicates that VBUS must be driven (active LOW).
USB_VDD3V3	-	USB 3.3 V power.
USB_VDD1V8	-	USB 1.8 V power.

54.4 Functional description

The following sections describe functional details of the USB_HS_HOST_SPEC module.

54.4.1 Proprietary Transfer Descriptor (PTD)

The standard Enhanced Host Controller Interface (EHCI) data structures as described in the *Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0*, are optimized for the host controller.

The optimized form of EHCI data structures is necessary because the controller does not have an AHB master interface. Instead, the controller exclusively uses its internal USB RAM to store and manage these data structures.

The controller manages schedules in two lists: periodic and asynchronous. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic, and reduce hardware and software complexity. The USB host controller executes transactions for devices by using a simple shared-memory schedule. This schedule consists of data structures organized into three lists:

- qISO — Isochronous transfer

- qINTL — Interrupt transfer
- qATL — Asynchronous transfer; for the control and bulk transfers

The system software maintains two lists for the host controller: periodic and asynchronous.

The high speed host controller has a maximum of 32 ISO, 32 INTL, and 32 ATL PTDs. These PTDs are used as channels to transfer data from the shared memory to the USB bus. These channels are allocated and de-allocated on receiving the transfer from the core USB driver.

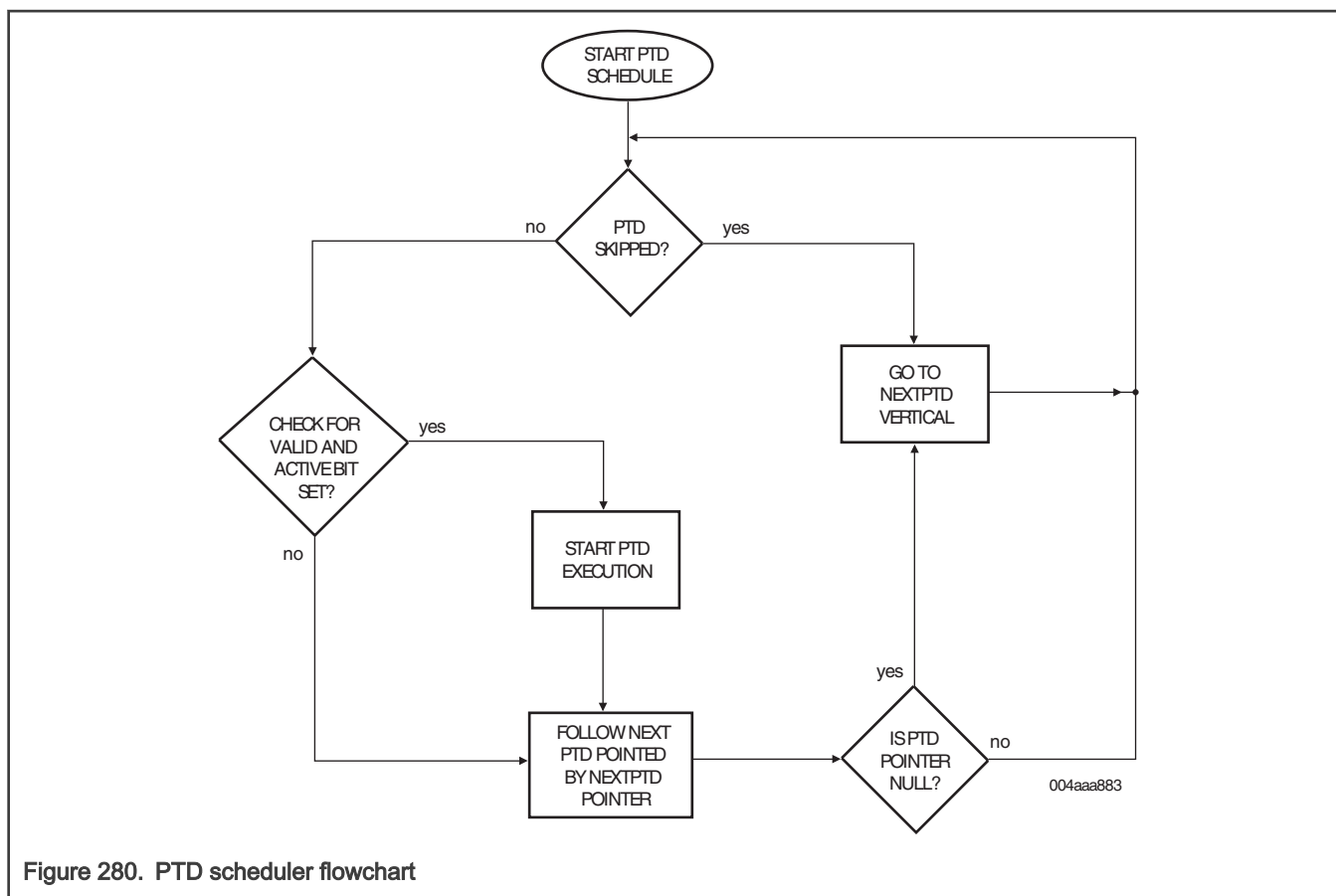
Multiple transfers are scheduled to the shared memory for various endpoints by traversing the next link pointer provided by endpoint data structures, until it reaches the end of the endpoint list. There are three endpoint lists: one for ISO endpoints, and the other for INTL and ATL endpoints. If the schedule is enabled, the host controller executes the ISO schedule, followed by the INTL schedule, and then the ATL schedule.

These lists are traversed and scheduled by the software according to the EHCI traversal rule. The host controller executes the scheduled ISO, INTL and ATL PTDs. The completion of a transfer is indicated to the software by the interrupt that can be grouped under various PTDs by using the AND or OR registers that are available for each schedule type: ISO, INTL and ATL. These registers are simple logic registers to decide the completion status of group and individual PTDs. When the logical conditions of the Done bit is true in the shared memory, it means that PTD has completed.

There are four types of interrupts in the high speed host controller: ISO, INTL, ATL and SOF. The NextPTD pointer is a feature that allows the high speed host controller to jump unused and skip PTDs. This will improve the PTD transversal latency time. The NextPTD pointer is not meant for same or single endpoint. The NextPTD works only in forward direction.

The NextPTD traversal rules defined by the high speed host controller are:

1. Start the PTD memory vertical traversal, considering the skip and LastPTD information.
2. If the current PTD is active and not done, perform the transaction.
3. Follow the NextPTD pointer as specified in bits 4 to 0 of DW4.
4. If combined with LastPTD, the LastPTD setting must be at a higher address than the NextPTD specified.
5. If combined with skip, the skip must not be set (logically) on the same position corresponding to NextPTD, pointed by the NextPTD pointer.
6. If PTD is set for skip, it will be neglected and the next vertical PTD will be considered.
7. If the skipped PTD already has a setting including a NextPTD pointer that will not be taken into consideration, the behavior will be the same as described in step 6.



54.4.1.1 PTD bit definition

Table 565. PTD bit definition

Symbol	Access	Description
V	SW - sets HW - resets	Valid: 0b: This bit is deactivated when the entire PTD is executed (Active bit is cleared), or when a error is encountered (Halt bit is set). 1b: Software updates to one when there is payload to be sent or received. The current PTD is active.
NrBytesToTransfer[14:0]	SW - writes	Number of Bytes to Transfer: This field indicates the number of bytes that can be transferred by this data structure. It is used to indicate the depth of the DATA field (32 kB - 1).
MaxPacketLength[10:0] TT_MPS_Len[10:0]	SW - writes	Transaction Translator Maximum Packet Size Length: This field indicates the maximum number of bytes that can be sent per start split, depending on the number of total bytes needed. If the total bytes to be sent for the entire millisecond is greater than 188 bytes, this field should be set to 188 bytes for an OUT token and 192 bytes for an IN token. Otherwise, this field should be equal to the total bytes sent.

Table continues on the next page...

Table 565. PTD bit definition (continued)

Symbol	Access	Description
Mult[1:0]	SW - writes	Multiplier: This field is a multiplier used by the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution.
I	SW - writes	Interrupt on Complete: If this bit is set and the PTD is completed, the interrupt status bit of the corresponding list is set to one.
EP[3:0]	SW - writes	Endpoint: This is the USB address of the endpoint within the function.
DeviceAddress[6:0]	SW - writes	Device Address: This is the USB address of the function containing the endpoint that is referred to by this buffer.
S	SW - writes	This bit indicates whether a split transaction has to be executed. 0 – No split transaction – speed is same as port speed. 1 – Split transaction.
Token[1:0]	SW - writes	Token: Identifies the token Packet Identifier (PID) for this transaction. 00 – OUT 01 – IN 10 – SETUP 11 – Undefined
EPTType[1:0]	SW - writes	Transaction type: 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt
SE[1:0]	SW - writes	This specifies the speed for a Control or Interrupt transaction to a device that is not high-speed: 00 – Full-speed 10 – Low-speed 01 or 11 – Undefined behavior For isochronous and bulk transactions this field must be set to 00. If a full-speed hub is connected to the port, the SE[1] bit indicates if a low-speed packet must be sent on a full-speed bus.
PortNumber[6:0]	SW - writes	Port Number: This indicates the port number of the hub.
HubAddress[6:0]	SW - writes	Hub Address: This indicates the hub address.
NextPTDPointer[4:0]	SW - writes	Next PTD Counter: Next PTD branching assigned by the PTDpointer.
J	SW - writes	Jump:

Table continues on the next page...

Table 565. PTD bit definition (continued)

Symbol	Access	Description
		0: increment the PTD pointer. 1: enable the next PTD branching.
DataStartAddress[15:0]	SW - writes	Data Start Address: "DataStartAddress + DataPayload_BaseAddress" is the start address that points to the start of the data buffer that will be sent or received on or from the USB bus. The hardware does not update this field when the transfer is completed.
uFrame[7:0]		This field is only applicable for interrupt and isochronous endpoints. Interrupt endpoint: Bits 2 to 0 of this field together with the μsA field represent the polling rate. When the polling rate is ≤ 1 ms, bits 2 to 0 are set to 0. If the polling rate is greater than 1 ms, bits 2 to 0 define the polling rate and bits 7 to 3 define the frame number when the packet must be transmitted. Isochronous endpoint: Bits 2 to 0 — Don't care. Bits 7 to 3 — Frame number at which this PTD will be sent.
NrBytesTransferred[14:0]	HW — writes	Number of Bytes Transferred: This field indicates the number of bytes sent or received for this transaction. If Mult[1:0] is greater than one, it is possible to store intermediate results in this field.
RL[3:0]	SW - writes	Reload: If RL is set to 0h, hardware ignores the NakCnt value. RL and NakCnt are set to the same value before a transaction.
NakCnt[3:0]	HW - writes SW - writes	NAK Counter: This field corresponds to the NakCnt field in TD. Software writes for the initial PTD launch. The V bit is reset if NakCnt decrements to zero and RL is a nonzero value. It reloads from RL if transaction is ACK-ed.
Cerr[1:0]	HW - writes SW - writes	Error Counter: This field corresponds to the Cerr[1:0] field in TD. The default value of this field is 0 for isochronous transactions. 00 — The transaction will not retry. 11 — The transaction will retry three times. The hardware will decrement these values.
DT	HW - updates SW - writes	Data Toggle: This bit is filled by software to start a PTD. If NrBytesToTransfer[14:0] is not complete, software needs to read this value and use this value to program the next PTD that will send data to the same endpoint.
P	SW - writes HW - updates	Ping: For high-speed transactions, this bit corresponds to the Ping state bit in the Status field of a TD. 0 — Ping is not set. 1 — Ping is set.

Table continues on the next page...

Table 565. PTD bit definition (continued)

Symbol	Access	Description
		For the first time, software sets the Ping bit to 0. For the successive asynchronous TD, software sets the bit in asynchronous TD based on the state of the bit for the previous asynchronous TD of the same transfer: The current asynchronous TD is completed with the Ping bit set. The next asynchronous TD will have its Ping bit set by the software.
SC	SW - writes 0 HW - updates	Start/Complete: 0 — Start split 1 — Complete split
X	HW - writes	Error: This bit corresponds to the Transaction Error bit in the Status field of iTD, siTD or TD. 0 — No PID error. 1 — If there are PID errors, this bit is set Active. The A and V bits are also set to inactive. This transaction is retried three times.
B	HW - writes	Babble: This bit corresponds to the Babble Detected bit in the Status field of iTD, siTD or TD. 1 — When babbling is detected, A and V are set to 0.
H	HW - writes	Halt: Set to a 1 by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this data structure. This can be caused by babble, the error counter counting down to 0, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to 0.
A	SW - sets	Active: This bit is the same as the Valid bit.
uSA[7:0]	SW - writes (0 → 1) HW - writes (1 → 0) after processing	This field is only used for periodic split transactions or if the port is enabled in HS mode. uSOF Active: When the frame number of bits uFrame[7:3] match the frame number of the USB bus, these bits are checked for 1 before they are sent for uSOF. For example: If uSA[7:0] = 1111 1111b: send ISO every uSOF of the entire millisecond. If uSA[7:0] = 0101 0101b: send ISO only on uSOF0, uSOF2, uSOF4, and SOF6.
Status0[2:0]	HW - writes	Isochronous IN or OUT status at uSOF0 Bit 2: Underrun Bit 1: Babble Bit 0: XactErr
Status1[2:0]	HW - writes	Isochronous IN or OUT status at uSOF1.

Table continues on the next page...

Table 565. PTD bit definition (continued)

Symbol	Access	Description
Status2[2:0]	HW - writes	Isochronous IN or OUT status at uSOF2.
Status3[2:0]	HW - writes	Isochronous IN or OUT status at uSOF3.
Status4[2:0]	HW - writes	Isochronous IN or OUT status at uSOF4.
Status5[2:0]	HW - writes	Isochronous IN or OUT status at uSOF5.
Status6[2:0]	HW - writes	Isochronous IN or OUT status at uSOF6.
Status7[2:0]	HW - writes	Isochronous IN or OUT status at uSOF7.
ISO_IN0[11:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
ISO_IN1[11:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
ISO_IN2[11:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 2 and frame number is correct.
ISO_IN3[11:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 3 and frame number is correct.
ISO_IN4[11:0]	HW - writes	Bytes received during uSOF4 if uSA[4] is set to 4 and frame number is correct.
ISO_IN5[11:0]	HW - writes	Bytes received during uSOF5 if uSA[5] is set to 5 and frame number is correct.
ISO_IN6[11:0]	HW - writes	Bytes received during uSOF6 if uSA[6] is set to 6 and frame number is correct.
ISO_IN7[11:0]	HW - writes	Bytes received during uSOF7 if uSA[7] is set to 6 and frame number is correct.
uSCS[7:0]	SW - writes (0 → 1) HW - writes (1 → 0) after processing	All bits can be set to one for every transfer. It specifies which uSOF the complete split needs to be sent. Start split and complete split Active bits, uSA = 0000 0001b, uSCS = 0000 0100b, will cause SS to execute in uFrame0 and CS in uFrame2.
SP_ISO_IN0[7:0]	HW - writes	Bytes received during uSOF0, if uSA[0] is set to 1 and frame number is correct.
SP_ISO_IN1[7:0]	HW - writes	Bytes received during uSOF1, if uSA[1] is set to 1 and frame number is correct.
SP_ISO_IN2[7:0]	HW - writes	Bytes received during uSOF2, if uSA[2] is set to 1 and frame number is correct.
SP_ISO_IN3[7:0]	HW - writes	Bytes received during uSOF3, if uSA[3] is set to 1 and frame number is correct.
SP_ISO_IN4[7:0]	HW - writes	Bytes received during uSOF4, if uSA[4] is set to 1 and frame number is correct.
SP_ISO_IN5[7:0]	HW - writes	Bytes received during uSOF5, if uSA[5] is set to 1 and frame number is correct.
SP_ISO_IN6[7:0]	HW - writes	Bytes received during uSOF6, if uSA[6] is set to 1 and frame number is correct.
SP_ISO_IN7[7:0]	HW - writes	Bytes received during uSOF7, if uSA[7] is set to 1 and frame number is correct.

Table continues on the next page...

Table 565. PTD bit definition (continued)

Symbol	Access	Description
S_Bytes	HW - writes	This field is used by HW to store an intermediate value of number of bytes received while handling a complete-split for interrupt IN transfers.

54.4.1.2 PTD on asynchronous list (qATL)

This data structure is used for both regular HS/FS transactions and split transactions. The table below shows the bit allocation of the bulk IN and OUT, asynchronous Transfer Descriptor (ATL PTD).

Table 566. PTD on asynchronous list (regular and split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]											uFrame[7:0]							J	R	NextPTDPointer [4:0]				V	0x00		
HubAddress[6:0]						PortNumber[6:0]					SE [1:0]		RL[3:0]			S	DeviceAddress[6:0]					EP[3:0]			0x04							
DataStartAddress[15:0]															I	NrBytesToTransfer[14:0]												0x08				
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0]			EP Type [1:0]		Token [1:0]		NrBytesToTransferred[14:0]												0x0C				

54.4.1.3 PTD on periodic list for regular transactions

This data structure is used for regular HS/FS transactions. The table below shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor.

Table 567. PTD on periodic list (regular transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	MaxPacketLength[10:0]										uFrame[7:0]							J	R	NextPTDPointer [4:0]				V	0x			

Table continues on the next page...

Table 567. PTD on periodic list (regular transaction) (continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																														00		
R												SE [1:0]		RL[3:0]			S	DeviceAddress[6:0]						EP[3:0]			0x04					
DataStartAddress[15:0]															I	NrBytesToTransfer[14:0]															0x08	
A	H	B	X	SC	P	DT	Cerr [1:0]		NakCnt[3:0]			EP Type [1:0]		Token [1:0]	NrBytesToTransferred[14:0]													0x0C				
Status7 [2:0]		Status6 [2:0]		Status5 [2:0]		Status4 [2:0]		Status3 [2:0]		Status2 [2:0]		Status1 [2:0]		Status0 [2:0]		uSA[7:0]				0x10												
ISO_IN_2[7:0]							ISO_IN_1[11:0]							ISO_IN_0[11:0]							0x14											
ISO_IN_5 [3:0]		ISO_IN_4[11:0]										ISO_IN_3[11:0]						ISO_IN_2 [11:8]				0x18										
ISO_IN_7[11:0]									ISO_IN_6[11:0]									ISO_IN_5[11:4]						0x1C								

54.4.1.4 PTD on periodic list for split transactions

This data structure is used for split transactions on the periodic list. The table below shows the bit allocation of the periodic IN and OUT, periodic Transfer Descriptor.

Table 568. PTD on periodic list (split transaction)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
R		Mult [1:0]		R	TT_MPS_LEN[10:0] / MaxPacketLength[10:0]										uFrame[7:0]						J	R	NextPTDPointer [4:0]				V	0x00				
HubAddress[6:0]						PortNumber[6:0]					SE[1:0]		RL[3:0]		S	DeviceAddress[6:0]						EP[3:0]			0x04							
DataStartAddress[15:0]															I	NrBytesToTransfer[14:0]															0x08	
A	H	B	X	SC	P	DT	Cerr [1:0]	NakCnt[3:0]		EP Type [1:0]	Token [1:0]	NrBytesToTransferred[14:0]															0x0C					
Status7 [2:0]		Status6 [2:0]		Status5 [2:0]		Status4 [2:0]		Status3 [2:0]		Status2 [2:0]		Status1 [2:0]		Status0 [2:0]		uSA[7:0]					0x10											
SP_ISO_IN_2[7:0]							SP_ISO_IN_1[7:0]							SP_ISO_IN_0[7:0] / S_Bytes[7:0]						uSCS[7:0]					0x14							
SP_ISO_IN_6[7:0]							SP_ISO_IN_5[7:0]							SP_ISO_IN_4[7:0]						SP_ISO_IN_3[7:0]					0x18							
R																								SP_ISO_IN_7[7:0]					0x1C			

54.4.1.5 Polling rate for Periodic transactions

It indicates how the hardware knows when to send a certain interrupt packet based on the polling rate. uFrame[2:0] defines the polling rate.

If set to 0 and the transaction is high-speed, the uSA determines during which uFrames a packet can be sent. If set to 0 and the transaction is a normal full-speed, the packet will be sent during every frame. If it is set to 0 and the transaction is a full-speed split transaction, the uSA and uCS fields will determine when to send a split transaction.

If uFrame[2:0] is different from 0 and the transaction is high-speed, the bits in uFrame[7:3] and the bits in uSA are used to know during which uFrames a packet must be sent.

If uFrame[2:0] is different from 0 and the transaction is full-speed, the bits in uFrame[7:3] are used to know during which uFrames a packet must be sent. If uFrame[2:0] is different from 0 and the transaction is a full-speed split transaction, the bits in uFrame[7:3] and the uSA and uCS fields will determine when to send a split transaction.

Table 569. Polling rate for periodic transactions

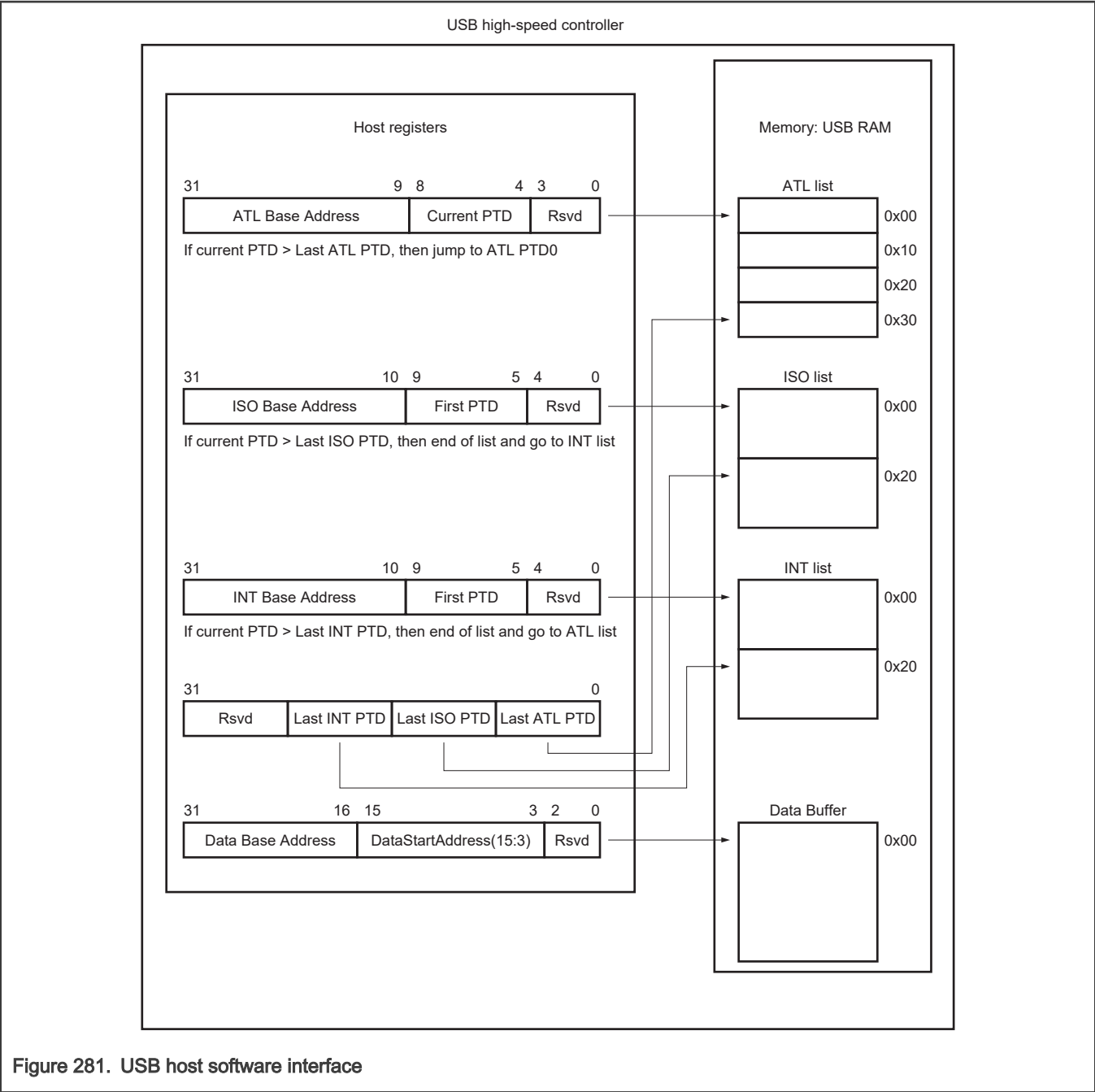
b	Rate	uFrame[2:0]	uFrame[7:3]	uSA[7:0]
1	1 uSOF	000b	Don't care	1111 1111b
2	2 uSOF	000b	Don't care	1010 1010b or 0101 0101b
3	4 uSOF	000b	Don't care	Any 2 bits set
4	1 mS	000b	Don't care	Any 1 bit set
5	2	001b	Bit 0 is compared with FRINDEX[3]	Any 1 bit set
6	4	010b	Bits[1:0] are compared with FRINDEX[4:3]	Any 1 bit set
7	8	011b	Bits[2:0] are compared with FRINDEX[5:3]	Any 1 bit set
8	16	100b	Bits[3:0] are compared with FRINDEX[6:3]	Any 1 bit set
9	32	101b	Bits[4:0] are compared with FRINDEX[7:3]	Any 1 bit set

54.5 Initialization

See Chip-specific USB2.0 HS Host Controller information.

54.6 Application information

The AHB slave interface of the USB host is used to access the registers and to configure the mode of the port (host mode or device mode). The figure below shows which part of the data is stored in registers and the location of the data in the USB RAM (16 KB, starting at address 0x4014 0000).



54.7 Memory map and register definition

This section includes the USB_HS_HOST_SPEC module memory map and detailed descriptions of all registers.

54.7.1 USB2.0 HS Host Controller register descriptions

54.7.1.1 USBHSH memory map

USBHSH0 base address: 4014_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (CAPLENGTH_CHIPID)	32	R	0101_0010h
4h	Host Controller Structural Parameters (HCSPARAMS)	32	R	0001_0011h
8h	INT PTD Base Address (HCCPARAMS)	32	RW	0000_0000h
Ch	Frame Length Adjustment (FLADJ_FRINDEX)	32	RW	0000_0020h
10h	ATL PTD Base Address (ATL_PTD_BASE_ADDRESS)	32	RW	0000_0000h
14h	ISO PTD Base Address (ISO_PTD_BASE_ADDRESS)	32	RW	0000_0000h
18h	INT PTD Base Address (INT_PTD_BASE_ADDRESS)	32	RW	0000_0000h
1Ch	DATA PAYLOAD Base Address (DATA_PAYLOAD_BASE_ADDRESS)	32	RW	0000_0000h
20h	USB Command (USBCMD)	32	RW	0000_0000h
24h	USB Interrupt Status (USBSTS)	32	RW	0000_0000h
28h	USB Interrupt Status (USBINTR)	32	RW	0000_0000h
2Ch	Port Status and Control (PORTSC1)	32	RW	0000_0000h
30h	ATL PTD Done Map (ATL_DONE)	32	RW	0000_0000h
34h	ATL PTD Skip Map (ATL_SKIP)	32	RW	0000_0000h
38h	ISO PTD Done Map (ISO_DONE)	32	RW	0000_0000h
3Ch	ISO PTD Skip Map (ISO_SKIP)	32	RW	0000_0000h
40h	INT PTD Done Map (INT_DONE)	32	RW	0000_0000h
44h	INT PTD Skip Map (INT_SKIP)	32	RW	0000_0000h
48h	Last PTD in use (LAST_PTD)	32	RW	0000_0000h
50h	Port Mode (PORT_MODE)	32	RW	0004_0000h

54.7.1.2 Version ID Register (CAPLENGTH_CHIPID)

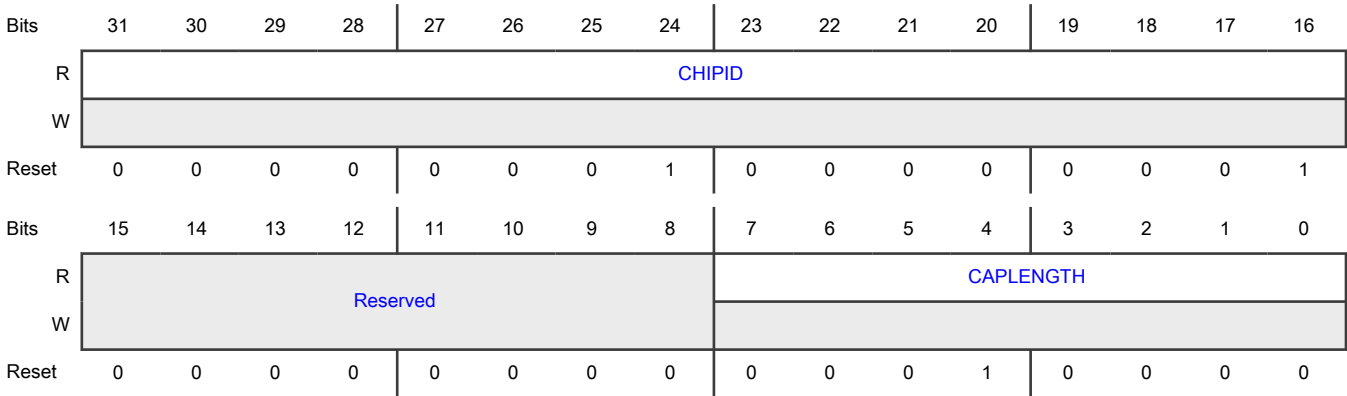
Offset

Register	Offset
CAPLENGTH_CHIPID	0h

Function

It describes the capability length and the revision of the USB IP.

Diagram



Fields

Field	Function
31-16 CHIPID	Chip identification. It indicates major and minor revision of the USB Controller. [31:24] = Major revision [23:16] = Minor revision Major revisions used: 0x01: USB2.0 high-speed host
15-8 —	Reserved
7-0 CAPLENGTH	Capability Length. This is used as an offset. It is added to the register base to find the beginning of the operational register space.

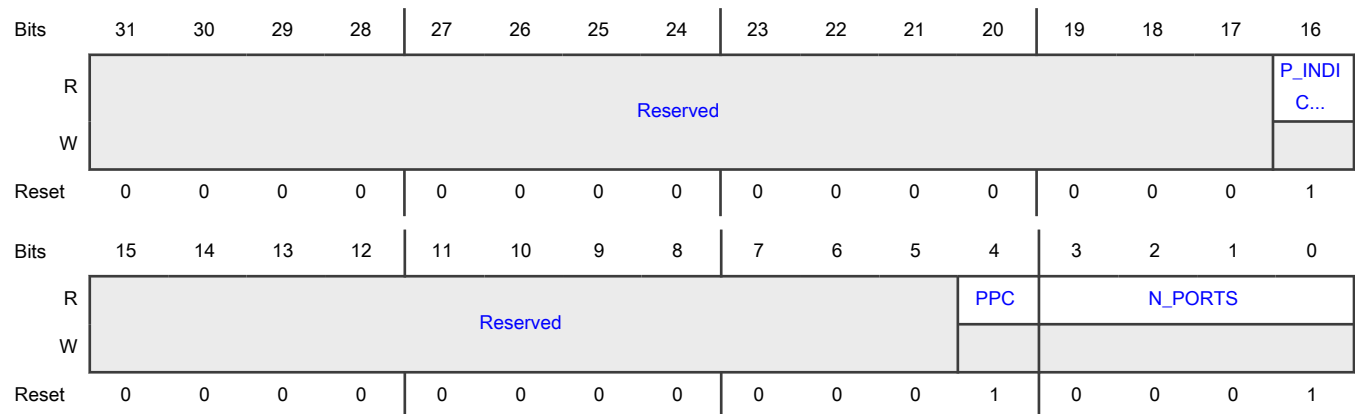
54.7.1.3 Host Controller Structural Parameters (HCSPARAMS)

Offset

Register	Offset
HCSPARAMS	4h

Function

It describes the USB host port configuration.

Diagram**Fields**

Field	Function
31-17 —	Reserved
16 P_INDICATOR	Port Indicator Control This bit indicates whether port indicator control is supported by the host controller. The value of this bit is controlled by the generic C_PORT_INDICATORS.
15-5 —	Reserved
4 PPC	Port Power Control This field indicates whether port power control is supported by the host controller. The value of this bit is controlled by the generic C_PORTPOWER_CONTROL.
3-0 N_PORTS	Number of Physical downstream ports. These bit specify the number of physical downstream ports implemented on this host controller. It is equal to 0x1 for this version.

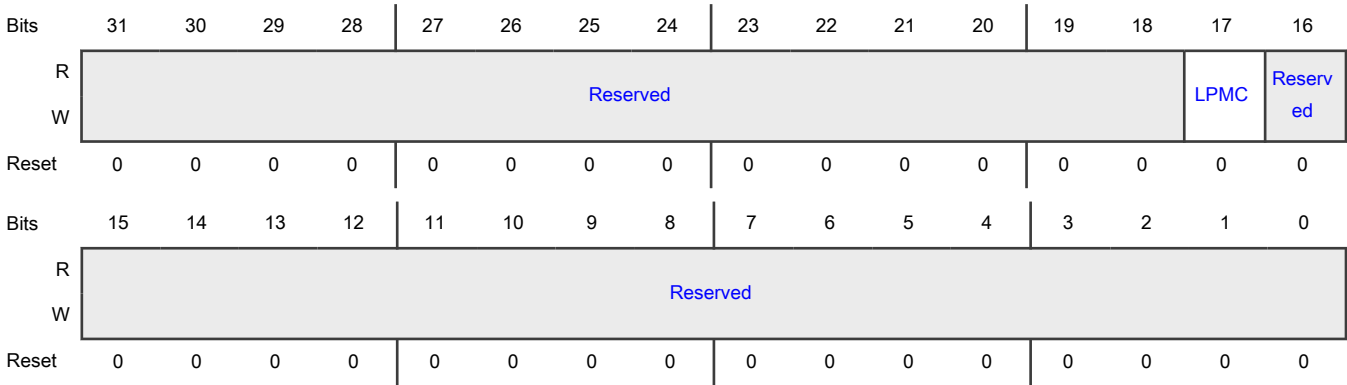
54.7.1.4 INT PTD Base Address (HCCPARAMS)**Offset**

Register	Offset
HCCPARAMS	8h

Function

It describes the USB host controller capability.

Diagram



Fields

Field	Function
31-18 —	Reserved
17 LPMC	Link Power Management Capability This indicates host controller support for the Link Power Management L1 state and associated PORTSC Suspend using L1, Suspend status and Device Address fields.
16-0 —	Reserved

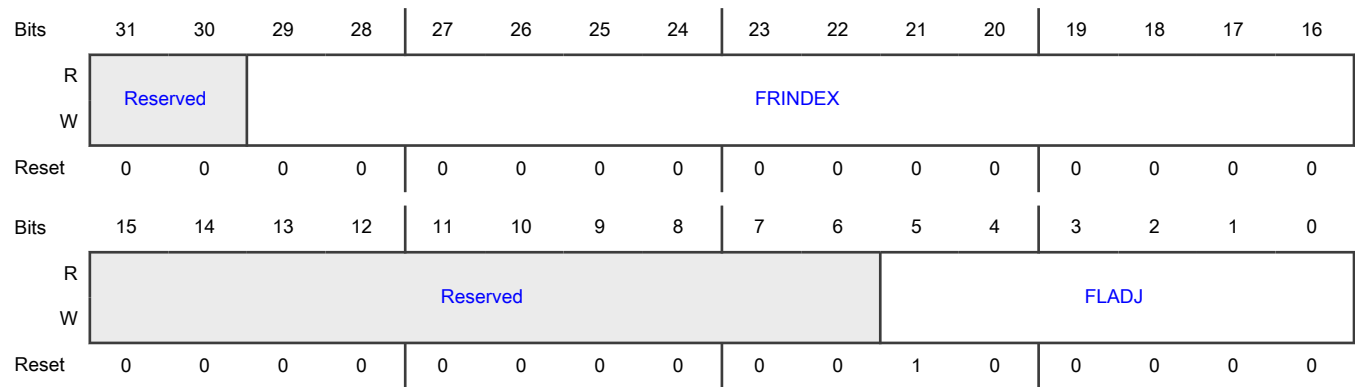
54.7.1.5 Frame Length Adjustment (FLADJ_FRINDEX)

Offset

Register	Offset
FLADJ_FRINDEX	Ch

Function

It controls the SOF frame length timing and the frame index.

Diagram**Fields**

Field	Function
31-30 —	Reserved
29-16 FRINDEX	Frame Index These bits indicate the frame number field in the SOF packet. The value in this field is incremented by one every 125 μ s (independent of the speed of the attached device). Software is only allowed to update this field when the Run/Stop bit is set to 0.
15-6 —	Reserved
5-0 FLADJ	Frame Length Timing Value. Each decimal value change to this register corresponds to 16 high-speed bit times. The SOF cycle time (number of SOF counter clock periods to generate a SOF micro-frame length) is equal to 59488 + value in this field. The default value is decimal 32 (20h), which gives a SOF cycle time of 60000.

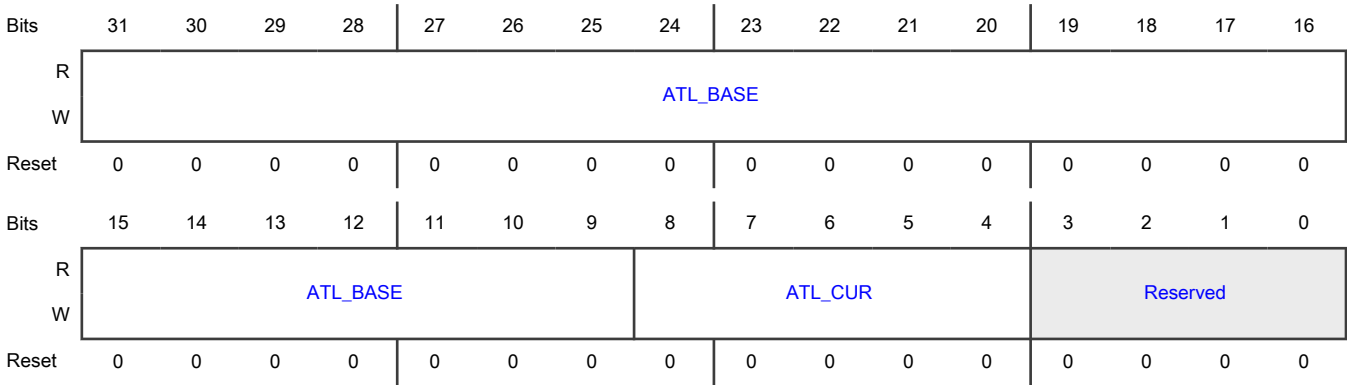
54.7.1.6 ATL PTD Base Address (ATL_PTD_BASE_ADDRESS)**Offset**

Register	Offset
ATL_PTD_BASE_ADDR ESS	10h

Function

It configures the start address of the ATL list. The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

Diagram



Fields

Field	Function
31-9 ATL_BASE	Start of ATL list. This is used by the hardware to find the start of the ATL list.
8-4 ATL_CUR	Current PTD This indicates the current PTD that is used by the hardware when it is processing the ATL list.
3-0 —	Reserved

54.7.1.7 ISO PTD Base Address (ISO_PTD_BASE_ADDRESS)

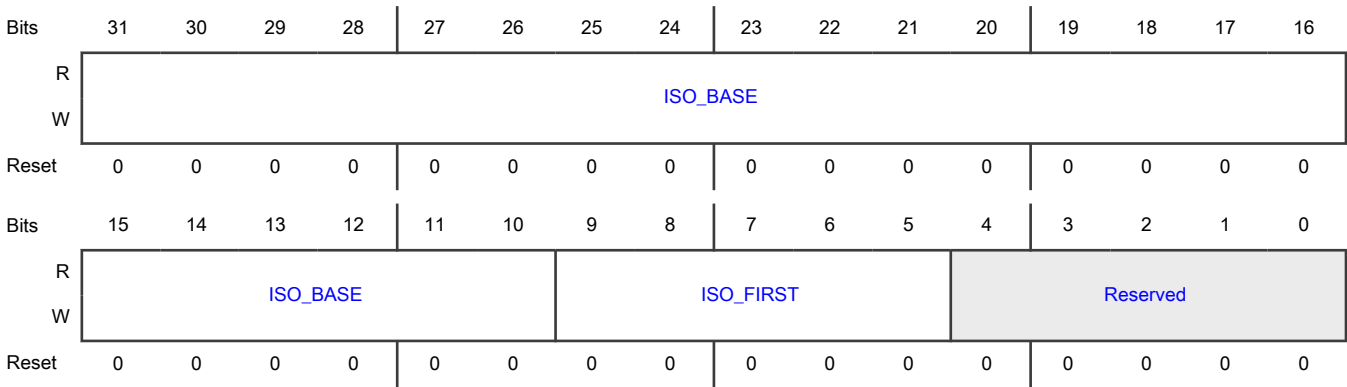
Offset

Register	Offset
ISO_PTD_BASE_ADDR ESS	14h

Function

It configures the start address of the ISO list. The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

Diagram



Fields

Field	Function
31-10 ISO_BASE	Start of ISO PTD list Base address to be used by the hardware to find the start of the ISO list.
9-5 ISO_FIRST	First PTD in the ISO list This indicates the first PTD that is used by the hardware when it is processing the ISO list.
4-0 —	Reserved

54.7.1.8 INT PTD Base Address (INT_PTD_BASE_ADDRESS)

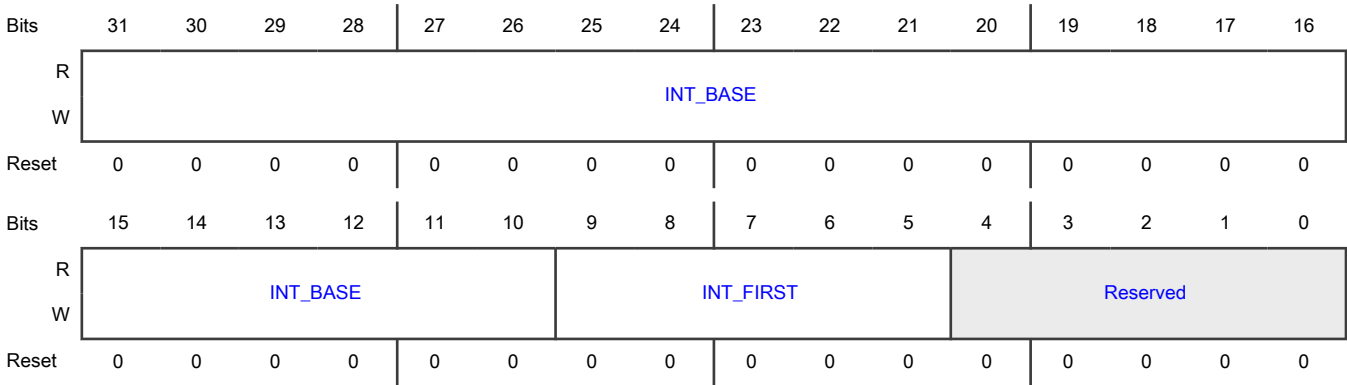
Offset

Register	Offset
INT_PTD_BASE_ADDR ESS	18h

Function

It configures the start address of the INT list. The hardware will only use the least significant bits of this register to find the correct location in RAM. For example, if the RAM is 4 kB, bits 11 to 0 of this register will be used to find the correct byte address.

Diagram



Fields

Field	Function
31-10 INT_BASE	Start of INT PTD list Base address to be used by the hardware to find the start of the INT list.
9-5 INT_FIRST	First PTD in the INT list This indicates the first PTD that is used by the hardware when it is processing the INT list.
4-0 —	Reserved

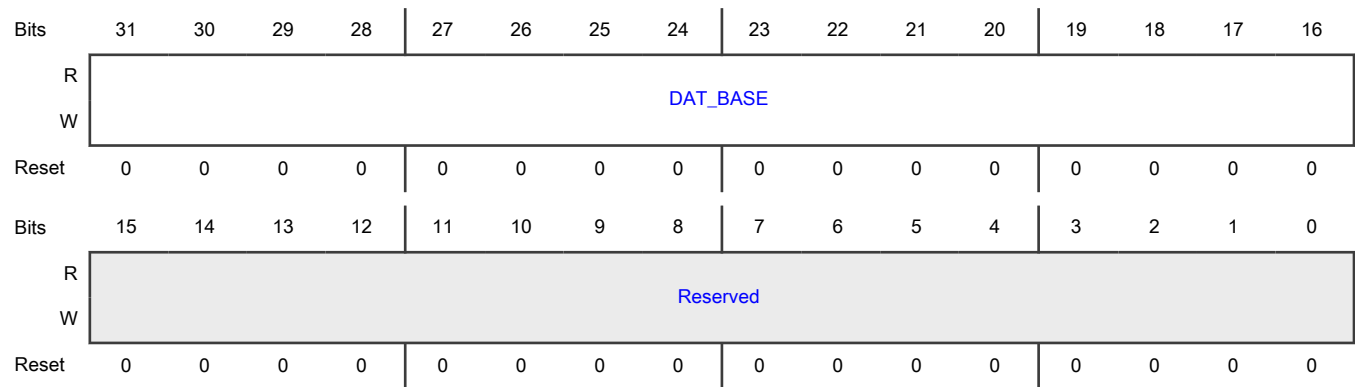
54.7.1.9 DATA PAYLOAD Base Address (DATA_PAYLOAD_BASE_ADDRESS)

Offset

Register	Offset
DATA_PAYLOAD_BASE_ADDRESS	1Ch

Function

The data payload base address register configures the start address of the data payload list.

Diagram**Fields**

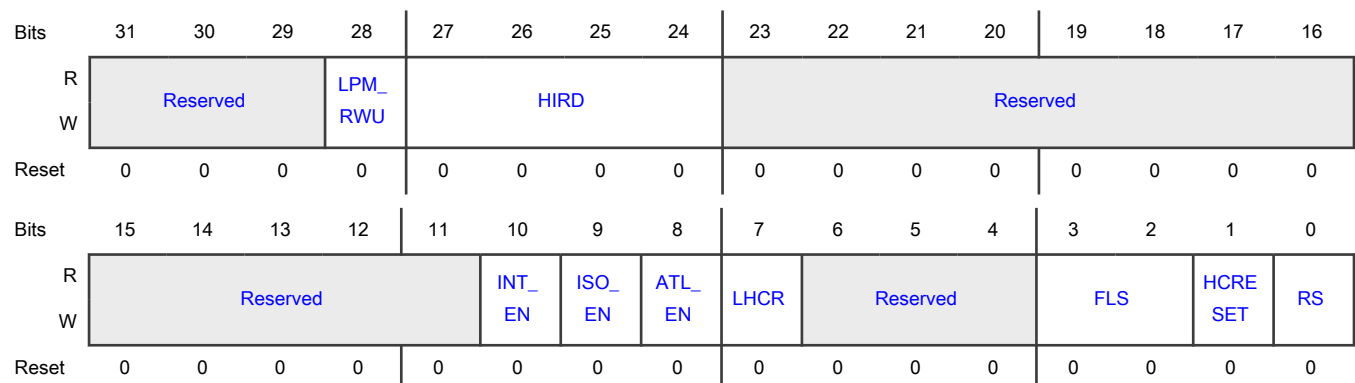
Field	Function
31-16 DAT_BASE	Data Payload Section Base Address It is used to find the to find the start of the data payload section. The hardware will only use the least significant bits required for addressing the correct location in RAM.
15-0 —	Reserved

54.7.1.10 USB Command (USBCMD)**Offset**

Register	Offset
USBCMD	20h

Function

The USB Command register controls the overall execution of the USB host scheduler.

Diagram

Fields

Field	Function
31-29 —	Reserved
28 LPM_RWU	Remote wake up. This bit indicates if the device is enabled for doing a remote wake up when it is in the L1 suspend state.
27-24 HIRD	Host-Initiated Resume Duration This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from a LPM state (example, L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM token's bmAttributes field) on entry into a low-power state. The host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. A host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM token, specifically: a value 0000b equals 50 μ s and each additional increment adds 75 μ s. For example, 0001b equals 125 μ s and a value 1111b equals 1175 μ s.
23-11 —	Reserved
10 INT_EN	INT List enabled When this bit is set, the hardware will process the INT list.
9 ISO_EN	ISO List enabled When this bit is set, the hardware will process the ISO list.
8 ATL_EN	ATL List enabled When this bit is set, the hardware will process the ATL list.
7 LHCR	Light Host Controller Reset This bit allows the driver software to reset the host controller without affecting the state of the ports.
6-4 —	Reserved
3-2 FLS	Frame List Size This field specifies the size of the frame list. This field is used to control when the frame list roll over interrupt bit must be set. 00b - 1024 elements 01b - 512 elements 10b - 256 elements 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 HCRESET	Host Controller Reset This control bit is used by the software to reset the host controller.
0 RS	Run/Stop If this bit is set to 1b, the USB clock will be always-on. 0b - Stop 1b - Run. The host controller executes the schedule.

54.7.1.11 USB Interrupt Status (USBSTS)

Offset

Register	Offset
USBSTS	24h

Function

It shows the interrupt status.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												SOF_ IRQ	INT_ IRQ	ISO_ IRQ	ATL_ IRQ
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												FLR	PCD	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-20 —	Reserved
19 SOF_IRQ	SOF Interrupt Request This bit is set when the host sends a Start of Frame token to the USB bus.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 INT_IRQ	INT Interrupt Request It indicates that an INT PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0b - No INT PTD event occurred. 1b - INT PTD event occurred.
17 ISO_IRQ	ISO Interrupt Request It indicates that an ISO PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0b - No ISO PTD event occurred. 1b - ISO PTD event occurred.
16 ATL_IRQ	ATL Interrupt Request It indicates that an ATL PTD (with I-bit set) was completed. The hardware interrupt line will be asserted if the respective enable bit in the USBINTR register is set. 0b - No ATL PTD event occurred. 1b - ATL PTD event occurred.
15-4 —	Reserved
3 FLR	Frame List Rollover Interrupt Request The host controller sets this bit to logic 1 when the frame list index rolls over its maximum value to 0. Software must write a one to clear the bit.
2 PCD	Port Change Detect Interrupt Request 0b - Disable 1b - Enable
1-0 —	Reserved

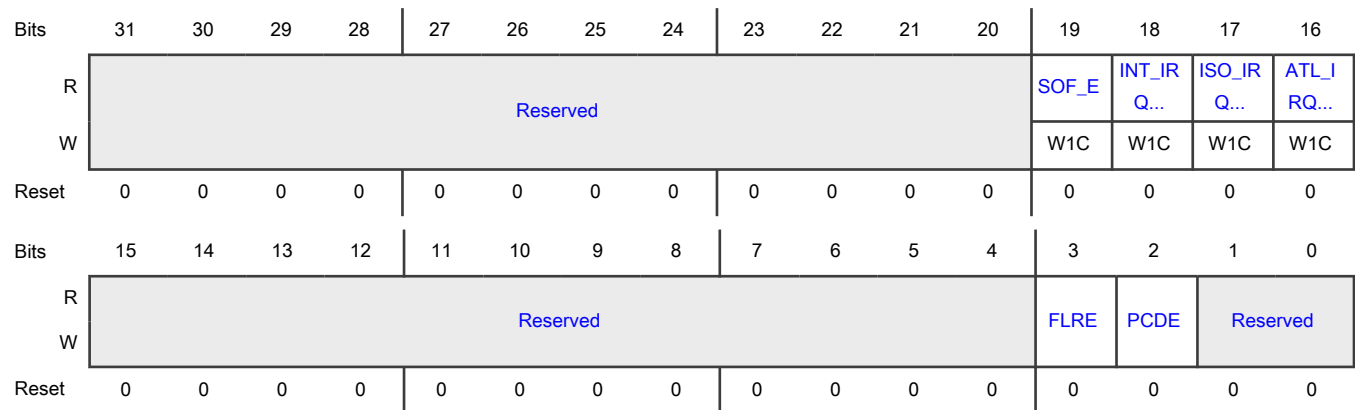
54.7.1.12 USB Interrupt Status (USBINTR)

Offset

Register	Offset
USBINTR	28h

Function

It shows the interrupt status.

Diagram**Fields**

Field	Function
31-20 —	Reserved
19 SOF_E	SOF Interrupt Request This bit is set when the host sends a Start of Frame token to the USB bus.
18 INT_IRQ_E	INT Interrupt Enable 0b - Disable 1b - Enable
17 ISO_IRQ_E	ISO Interrupt Enable 0b - Disable 1b - Enable
16 ATL_IRQ_E	ATL Interrupt Enable 0b - Disable 1b - Enable
15-4 —	Reserved
3 FLRE	Frame List Rollover Interrupt Enable 0b - Disable 1b - Enable
2 PCDE	Port Change Detect Interrupt Enable 0b - Disable 1b - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0	Reserved
—	

54.7.1.13 Port Status and Control (PORTSC1)

Offset

Register	Offset
PORTSC1	2Ch

Function

It indicates the port status and configures the port operation.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DEV_ADD								SUS_STAT		WOO	PSPD		PTC			
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	PIC		Reserv ed	PP	LS		SUS_ L1	PR	SUSP	FPR	OCC	OCA	PEDC	PED	CSC	CCS	
W													W1C			W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31-25 DEV_ADD	Device Address for LPM tokens 7-bit USB device address that is used when sending an LPM token to the device attached to and immediately downstream of the associated root port. A value of 0 indicates no device is present or support for LPM feature is not present on this device.
24-23 SUS_STAT	Suspend Status These bits are used by software to determine whether the most recent L1 suspend request was successful 10b Not supported – Device does not support the L1 state (STALL) 11b Timeout/Error – Device failed to respond or an error occurred. This field is updated by hardware immediately following the completion of an L1 transition request (via an LPM token). To avoid any race conditions with hardware, software should only consume the contents of this field when Suspend = 0b (port no longer in L1).

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - State transition was successful (ACK) 01b - Device was unable to enter the L1 state at this time (NYET) 10b - Device does not support the L1 state (STALL) 11b - Timeout/Error – Device failed to respond or an error occurred.
22 WOO	Wake on overcurrent enable Writing this bit to a one enables the port to be sensitive to overcurrent conditions as wake-up events. This field is 0 if Port Power is 0.
21-20 PSPD	Port Speed 00b - Reserved 01b - Full-speed 10b - High-speed 11b - Reserved
19-16 PTC	Port Test Control A non-zero value indicates that the port is operating in the test mode as indicated by the value. All other values are Reserved. The reserved values should not be written by software. 0000b - Test mode not enabled 0001b - Test J_STATE 0010b - Test K_STATE 0011b - TEST SE0_NAK 0100b - Test_Packet 0101b - Test Force_Enable
15-14 PIC	Port Indicator Control Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is logic 0. If P_INDICATOR is set to one, these bits will indicate the value of the port indicators 00b - Port Indicators are off 01b - Amber 10b - Green 11b - Undefined
13 —	Reserved
12 PP	Port Power

Table continues on the next page...

Table continued from the previous page...

Field	Function
	The function of this bit depends on the value of the Port Power Control (PPC) bit in the HCSPARAMS register. If PPC = 0b, this bit is read-only and will always be set to 1b. If PPC = 1b, this bit is RW, If the bit is set to 0, the port is not powered. If the bit is set to one the port is powered.
11-10 LS	Line Status This field reflects the current logical levels of the DP (bit 11) and DM (bit 10) signal lines.
9 SUS_L1	Suspend using L1 When this bit is set to a 1 and a non-zero value is specified in the Device Address field, the host controller will generate an LPM Token to enter the L1 state whenever software writes a one to the Suspend bit, as well as L1 exit timing during any device or host-initiated resume. When set to 0 the host controller will employ the legacy (L2) suspend mechanism. Software should only set this bit when the device attached immediately downstream of this root port supports L1 transitions. 0b - Suspend using L2 1b - Suspend using L1
8 PR	Port Reset Software writes a logic 1 to indicate the start of the reset. SW writes a logic 0 to end the reset sequence. If the reset sequence on the USB bus is finished HW will clear the bit. SW should only check the PSPD field to know the speed of the attached device when the Port Reset bit is 0. This field is 0 if Port Power is 0. 0b - Port is not in the reset state 1b - Port is in the reset state
7 SUSP	Suspend Suspend Using L1 field determines the suspend state the host controller attempts. In the suspend state, downstream propagation of data is blocked on this port, except for port reset. If this bit is set to a one when a transaction is in progress then the blocking will not occur until the end of the current transaction. A write of 0 is ignored by the hardware. The hardware will unconditionally set this bit to 0 when: Software sets the Force Port Resume bit to 0. Software sets the Port Reset bit to a one. This field is 0 if Port Power is 0 or Current Connect Status is 0. 0b - Enabled port is not suspended 1b - Enabled port is in the L1 or L2 suspend state
6 FPR	Force Port Resume The resume signaling is driven on the port as long as this bit remains a one. For legacy (L2) transitions, software must appropriately time the resume and set this bit to a 0 when the appropriate amount of time has elapsed. Software does not need to time resume signaling for L1 transactions as host controller hardware will automatically enforce the necessary timing and clear this bit when the port has fully resumed. Software can influence the amount of time the hardware will drive resume signaling during L1 exit via the HIRD field within the USBCMD register. This field is 0 if Port Power is 0. 0b - No Resume (K-state) detected or driven on the port. 1b - Resume (K-state) detected or driven on the port.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 OCC	Over-current active This bit will automatically move from one to 0 when the over-current condition is removed. 0b - OCA value has not changed 1b - OCA value has changed
4 OCA	Over-current active This bit will automatically move from one to 0 when the over-current condition is removed. 0b - Port does not have an over-current condition 1b - Port has an over-current condition
3 PEDC	Port Enabled/Disabled Change This field is 0 if Port Power is 0. 0b - PED value has not changed 1b - PED value has changed
2 PED	Port Enabled/Disabled Firmware can clear the bit to disable the port. Firmware cannot set the bit. This bit will be set at the end of a port reset sequence. This field is 0 if Port Power is 0. 0b - Port Disabled 1b - Port Enabled
1 CSC	Connect Status Change This bit is 0 if Port Power is 0. 0b - CCS value has not changed 1b - CCS value has changed
0 CCS	Current Connect Status This bit is 0 if Port Power is 0. 0b - No Device is present 1b - Device is present

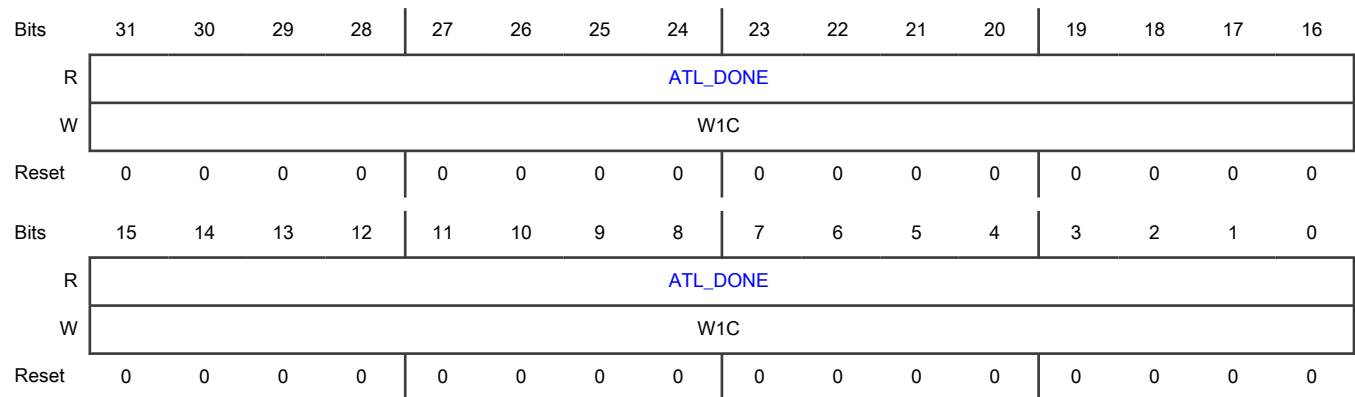
54.7.1.14 ATL PTD Done Map (ATL_DONE)

Offset

Register	Offset
ATL_DONE	30h

Function

It a direct map of the done status of the 32 ATL PTDs.

Diagram**Fields**

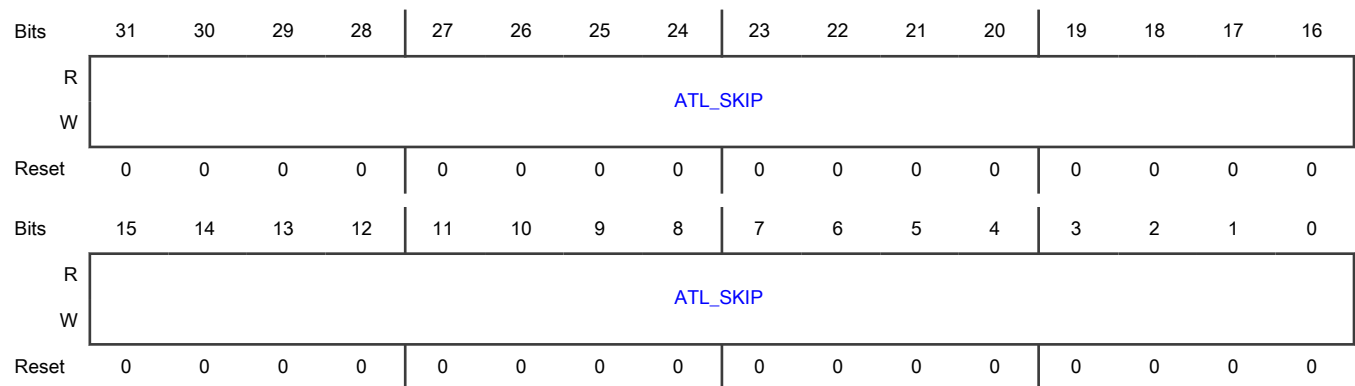
Field	Function
31-0	ATL Done
ATL_DONE	The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

54.7.1.15 ATL PTD Skip Map (ATL_SKIP)**Offset**

Register	Offset
ATL_SKIP	34h

Function

It a direct map of the skip status of the 32 ATL PTDs.

Diagram

Fields

Field	Function
31-0 ATL_SKIP	ATL PTD Skip Map When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

54.7.1.16 ISO PTD Done Map (ISO_DONE)**Offset**

Register	Offset
ISO_DONE	38h

Function

It a direct map of the done status of the 32 ISO PTDs.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ISO_DONE															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISO_DONE															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

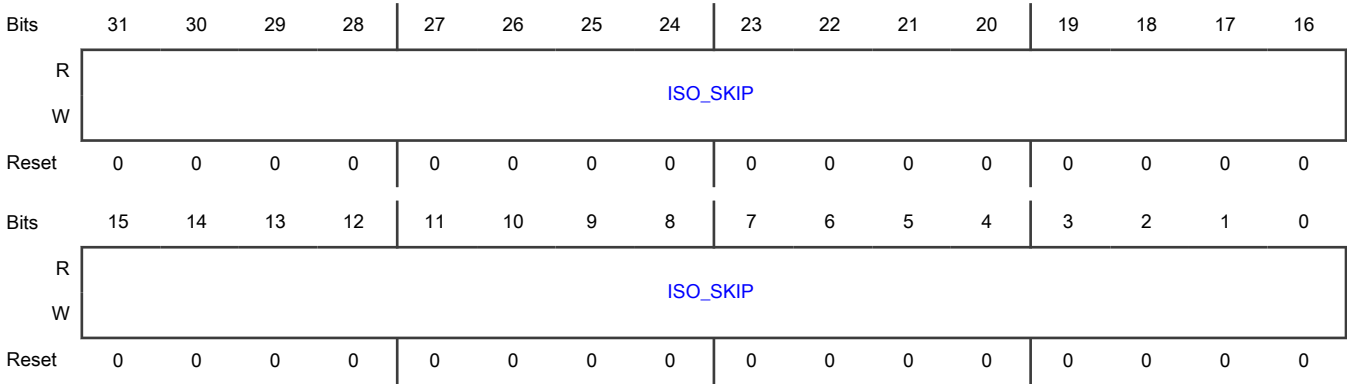
Field	Function
31-0 ISO_DONE	ISO Done The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

54.7.1.17 ISO PTD Skip Map (ISO_SKIP)**Offset**

Register	Offset
ISO_SKIP	3Ch

Function
It is a direct map of the skip status of the 32 ISO PTDs.

Diagram



Fields

Field	Function
31-0 ISO_SKIP	ISO Skip When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

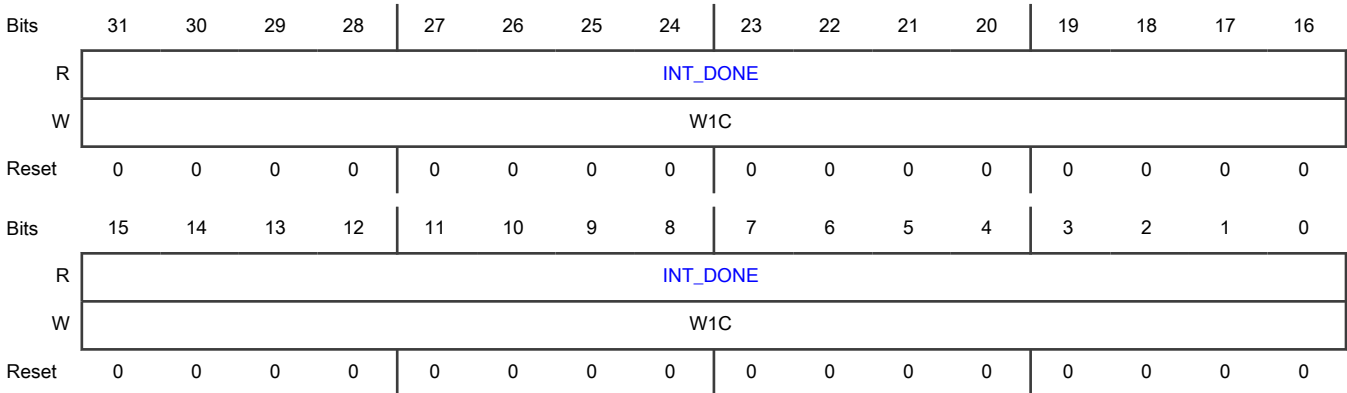
54.7.1.18 INT PTD Done Map (INT_DONE)

Offset

Register	Offset
INT_DONE	40h

Function
It is a direct map of the done status of the 32 INT PTDs.

Diagram



Fields

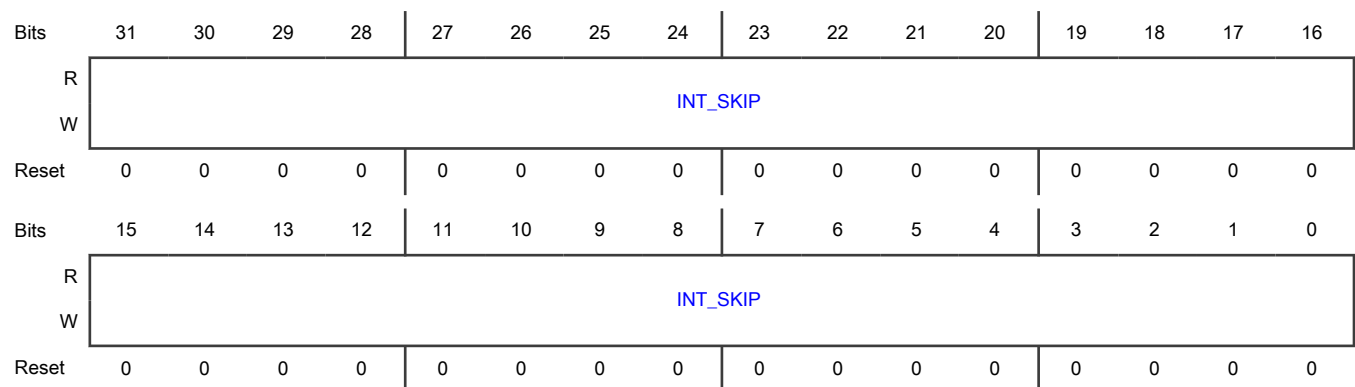
Field	Function
31-0 INT_DONE	INT Done The bit corresponding to a certain PTD will be set to logic 1 as soon as that PTD execution is completed. Writing a one to a bit in the Done Map register will clear the bit.

54.7.1.19 INT PTD Skip Map (INT_SKIP)**Offset**

Register	Offset
INT_SKIP	44h

Function

It a direct map of the skip status of the 32 INT PTDs.

Diagram**Fields**

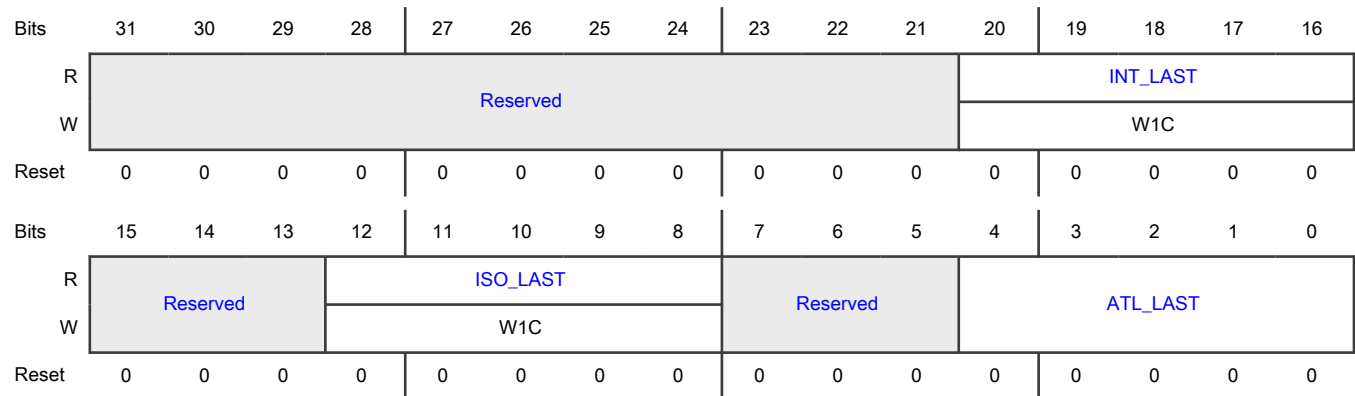
Field	Function
31-0 INT_SKIP	INT Skip When a bit in the PTD Skip Map is set to logic 1, the corresponding PTD will be skipped, independent of the V bit setting. The information in that PTD is not processed. Hardware will go automatically to the next PTD.

54.7.1.20 Last PTD in use (LAST_PTD)**Offset**

Register	Offset
LAST_PTD	48h

Function

It indicates the last PTD in the ATL list.

Diagram**Fields**

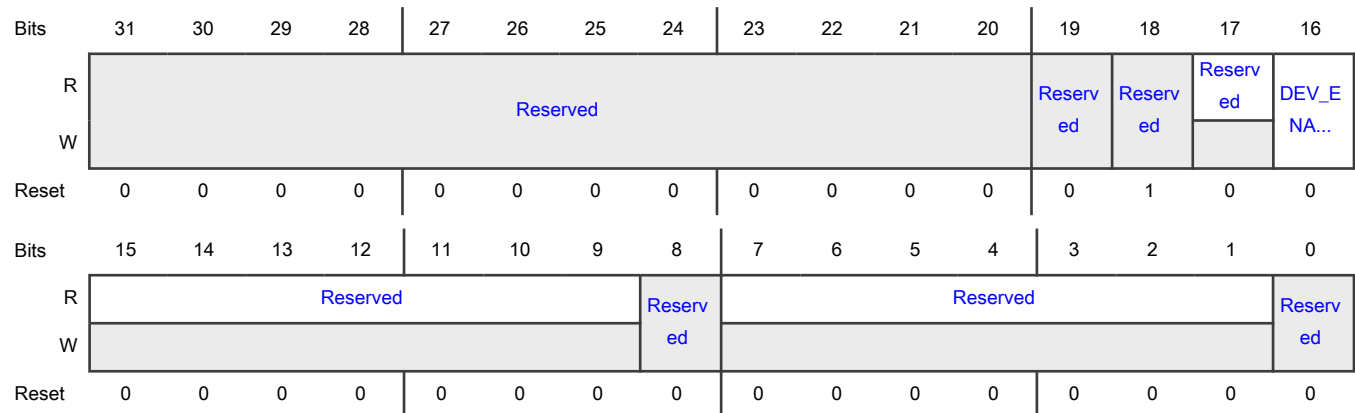
Field	Function
31-21 —	Reserved
20-16 INT_LAST	Last PTD in INT list This indicates the last PTD in the INT list. If hardware has reached this PTD, it will continue with processing the ATL list.
15-13 —	Reserved
12-8 ISO_LAST	Last PTD in ISO list This indicates the last PTD in the ISO list. If hardware has reached this PTD, it will continue with processing the INT list
7-5 —	Reserved
4-0 ATL_LAST	Last PTD in ATL list If hardware has reached this PTD and the J bit is not set, it will go to PTD0 as the next PTD to be processed.

54.7.1.21 Port Mode (PORT_MODE)**Offset**

Register	Offset
PORT_MODE	50h

Function

It controls the host or device role.

Diagram**Fields**

Field	Function
31-20 —	Reserved Read value is undefined, only zero should be written.
19 —	Reserved
18 —	Reserved
17 —	Reserved Read value is undefined, only zero should be written.
16 DEV_ENABLE	If this bit is set to one, the port will behave as a USB device. If this bit is set to zero, the port will be controlled by the USB host block.
15-9 —	Reserved Read value is undefined, only zero should be written.
8 —	Reserved
7-1 —	Reserved Read value is undefined, only zero should be written.
0 —	Reserved

Chapter 55

Universal Serial Bus 2.0 HS Device Controller

55.1 Chip-specific USB2.0 Device Controller information

Table 570. Reference links to related information

Topic	Related module	Reference
Full description	USBHSD	USB2.0 Device
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

When RT500 acts as host, both HS and FS device can be connected to. When RT500 acts as a device, it depends on the host connected to, if host is a HS, RT500 acts as a HS device. If host is a FS, RT500 acts as a FS device.

55.1.1 Initialization

Initial configuration of the USB high speed device controller can be accomplished as follows:

- Enable the USB high speed device controller clocks by setting the USBHS_DEVICE_CLK and USBHS_SRAM_CLK bits in the PSCCTL0 register (See CLKCTL0 registers).
- Select a clock source for the USB high speed device controller using the SEL bits in USBHSFCLKSEL register (See CLKCTL0 registers).
- Select a clock divide for the USB high speed device controller using the DIV bits in USBHSFCLKDIV register (See CLKCTL0 registers). To have high-speed USB operating, the CPU clock must be configured to a minimum frequency of 60 MHz.
- Clear the USB high speed device controller and USB RAM peripheral resets in the PRSTCTL0 register by writing to the USBHS_DEVICE and USBHS_SRAM bits in PRSTCTL0_CLR register (See RSTCTL0 registers).
- Enable the USBHS_SRAM_APD and USBHS_SRAM_PPD bits in PDRUNCFG1 register to power-up the USB device controller (See SYSTCTL0 registers).
- The High-speed USB device/host controller interrupt and USB activity interrupt are available to the NVIC (See NVIC). Enable USB0_NEEDCLK bits in STARTEN1 register (See SYSTCTL0 register section) to allow USB device interrupt wake-up from the deep-sleep mode.
- Configure the USB wake-up if necessary (See [USB Wake-up](#)).
- Use the IOPCTL configuration registers to connect USB high speed device controller functions to external pins (See IOPCTL).

55.2 About this module

55.2.1 Overview

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The USB device controller enables high-speed (480 Mb/s) data exchange with a USB host controller. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The USB block contains the USB RAM, which enables shared access of the endpoint buffer and control data between the controller and the AHB bus. It is also possible to use this RAM as generic memory when the USB is not in use.

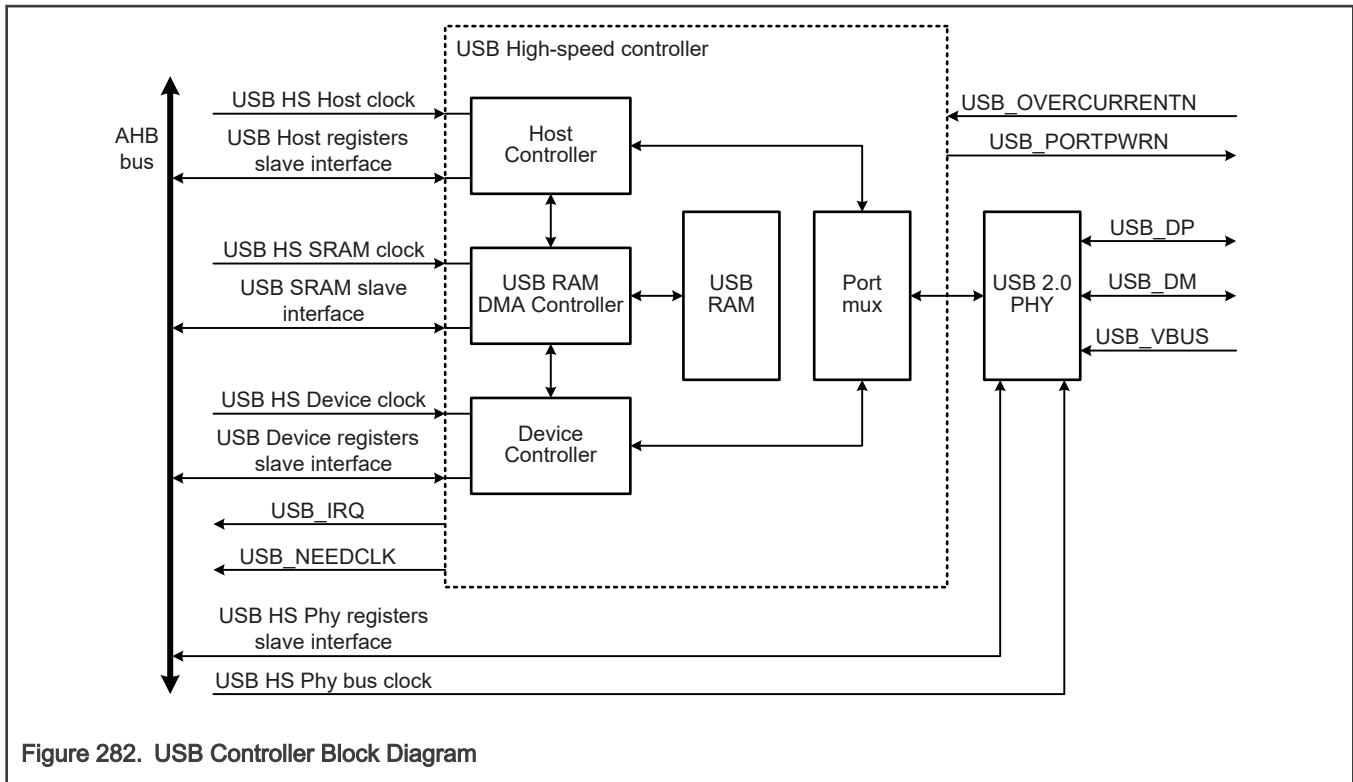
55.2.2 Features

USB2.0 high-speed device controller supports following features:

- Single and double buffering.
- Enables high-speed (480 Mb/s) data exchange with a USB host controller.
 - Host schedules transactions in 125 μ s frames. Each frame contains a Start Of Frame (SOF) marker and transactions that transfer data to or from device endpoints.
- Supports 12 physical (6 logical) endpoints including control endpoints.
 - Four types of transfers defined for the endpoints.
 - Control transfers are used to configure the device.
 - Interrupt transfers are used for periodic data transfer.
 - Bulk transfers are used when the latency of transfer is not critical.
 - Isochronous transfers have guaranteed delivery time but no error correction.
 - Non-control endpoints supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from deep-sleep mode on USB activity and remote wake-up.
- Supports A device/B device detection.
- Supports Link Power Management (LPM).
- On-chip USB PHY.
 - USB PHY sends/receives the bi-directional USB_DP and USB_DM signals of the USB bus.

55.2.3 Block Diagram

The figure below shows the block diagram of the USB controller.



55.3 Signals

Table 571. USB Device Controller Signals

Name	I/O	Description
USB_VBUS	I	VBUS status input. When this function is not enabled via its corresponding IOPCTL register, it is driven HIGH internally.
USB_DP	I/O	Positive differential data.
USB_DM	I/O	Negative differential data.
USB_VDD3V3	-	USB1 3.3 V power.
USB_VDD1V8	-	USB1 1.8 V power.
USB_NEEDCLK	I/O	USB activity interrupt wake-up
USB_FRAME	O	USB Frame. It reflects the 500 kHz clock (full-speed mode) or the 4 kHz clock (high-speed mode) derived from the incoming Start of Frame tokens sent by the USB host.

55.4 Functional description

55.4.1 Endpoint command/status list

The section below gives an overview on how the Endpoint List is organized in memory. The USB EP Command/Status List start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the figure.

USB EP Command/Status FIFO start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes										EP0 OUT Buffer Address Offset												0x00				
R	R	R	R	R	R	Reserved										SETUP Bytes Buffer Address Offset												0x04				
A	R	S	TR	TV	R	EP0 IN Buffer NBytes										EP0 IN Buffer Address Offset												0x08				
R	R	R	R	R	R	Reserved										Reserved												0x0C				
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes										EP1 OUT Buffer 0 Address Offset												0x10				
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes										EP1 OUT Buffer 1 Address Offset												0x14				
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes										EP1 IN Buffer 0 Address Offset												0x18				
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes										EP1 IN Buffer Address Offset												0x1C				
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes										EP2 OUT Buffer 0 Address Offset												0x20				
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes										EP2 OUT Buffer 1 Address Offset												0x24				
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes										EP2 IN Buffer Address Offset												0x28				
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes										EP2 IN Buffer Address Offset												0x2C				
...																																
A	D	S	TR	RF TV	T	EP5 OUT Buffer 0 NBytes										EP5 OUT Buffer 0 Address Offset												0x50				
A	D	S	TR	RF TV	T	EP5 OUT Buffer 1 NBytes										EP5 OUT Buffer 1 Address Offset												0x54				
A	D	S	TR	RF TV	T	EP5 IN Buffer 0 NBytes										EP5 IN Buffer 0 Address Offset												0x58				
A	D	S	TR	RF TV	T	EP5 IN Buffer 1 NBytes										EP5 IN Buffer 1 Address Offset												0x5C				

Figure 283. Endpoint command/status list

Figure 283. Endpoint command/status list

Table 572. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. HW can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to 1. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a one to the corresponding "skip" bit in the USB Endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a one to the "skip" bit. If hardware receives a token for an endpoint that is not active, it will return the following handshake or data: • Non-isochronous endpoint: NAK handshake is sent. • Isochronous IN endpoint: empty data packet is sent. • Isochronous OUT endpoint: received data is ignored and no handshake is sent.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. When a bus reset is received, firmware must set the disable bit of all endpoints to 1. Software can only modify this bit when the Active bit is 0.

Table continues on the next page...

Table 572. Endpoint command/status bit definitions (continued)

Symbol	Access	Description
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The Active bit has always higher priority than the Stall bit. This means that a Stall handshake is only sent when the Active bit is 0 and the stall bit is one. Software can only modify this bit when the Active bit is 0.
TR	R/W	Toggle Reset When software sets this bit to one, the HW will set the toggle value equal to the value indicated in the "toggle value" (TV) bit. For the control endpoint 0, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to 0 when the endpoint is reset.
RF / TV	R/W	Rate Feedback mode / Toggle value For the control endpoint 0 this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. For the non-control endpoints, this bit is used together with the T-bit to identify the type of endpoint When the endpoint type (T) is set to generic endpoint, this bit selects between bulk endpoint and interrupt endpoint in rate-feedback mode. 0: Bulk endpoint with maximum packet size of 512 bytes in HS mode and 64 bytes in FS mode 1: Interrupt endpoint in 'rate feedback mode'. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in 'rate feedback mode', the TR bit must always be set to 0. When the endpoint type (T) is set to periodic, this bit determines if the endpoint is interrupt or isochronous. 0: Isochronous endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024). 1: Interrupt endpoint (Max Packet Size is determined by the smallest value when comparing NBytes field with 1024).
T	R/W	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or rate feedback interrupt endpoint. In case of an rate feedback interrupt endpoint, the Maximum Packet Size in High-Speed mode can only be maximum 512 bytes. 1: Periodic endpoint. The RF / TV bit determines if the endpoint is isochronous or interrupt.
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred.

Table continues on the next page...

Table 572. Endpoint command/status bit definitions (continued)

Symbol	Access	Description
		NOTE If a short packet is received on an OUT endpoint, the Active bit clears and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 16 to 6 of the buffer start address. This address offset is updated by HW after each successful reception/transmission of a packet. HW increments the original value with the rounded up integer value when the packet size is divided by 64. E.g. if a packet of 200 bytes is successfully received, the Address Offset will be incremented by 4. Examples: • If a packet of 64 bytes is successfully received, the Address Offset is incremented by 1. • If a packet of less than 64 bytes is received, the Address Offset is also incremented by 1. • If a packet with 0 bytes is received, the Address Offset is not incremented.

When receiving a SETUP token for endpoint 0, the HW will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. HW will ignore all other fields. In case the SETUP stage contains more than 8 bytes, it will only write the first 8 bytes to memory. A USB compliant host must never send more than 8 bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the Active bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the Active bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See [Control endpoint 0](#).

When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive. When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

55.4.1.1 Fixed endpoint configuration

The table below shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in the table for each type of end point.

Table 573. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
1	3	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
2	4	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
2	5	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
3	6	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes

Table continues on the next page...

Table 573. Fixed endpoint configuration (continued)

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
3	7	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
4	8	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
4	9	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes
5	10	Interrupt/Bulk/Isochronous	Out	1024/512/1024	Yes
5	11	Interrupt/Bulk/Isochronous	In	1024/512/1024	Yes

55.4.1.2 Generic endpoint: single buffering

To enable single buffering, software must set the corresponding **BUF_SB** to 0. The **BUF** bits indicates which buffer will be used by hardware when the next token is received.

When software wants to transfer data, it programs the different bits in the Endpoint command/status list entry for the desired endpoint and sets the Active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes goes to 0, hardware clears the Active bit and sets the corresponding endpoint interrupt status bit in **USB Interrupt Status (INTSTAT)**.

Software must wait until hardware has cleared the Active bit to change the command/status bits in the Endpoint command/status list entry. This prevents hardware from overwriting a new value programmed by software with old values that were still cached.

If software wants to disable the Active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint **SKIP** bit.

55.4.1.3 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding **BUF_SB** bit to 1. The **BUF** bits indicates which buffer will be used by hardware when the next token is received.

When hardware clears the Active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding **BUF** bit.

55.4.1.4 Control endpoint 0

Use of the Active bit is slightly different between OUT and IN endpoints.

- When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive.
- When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

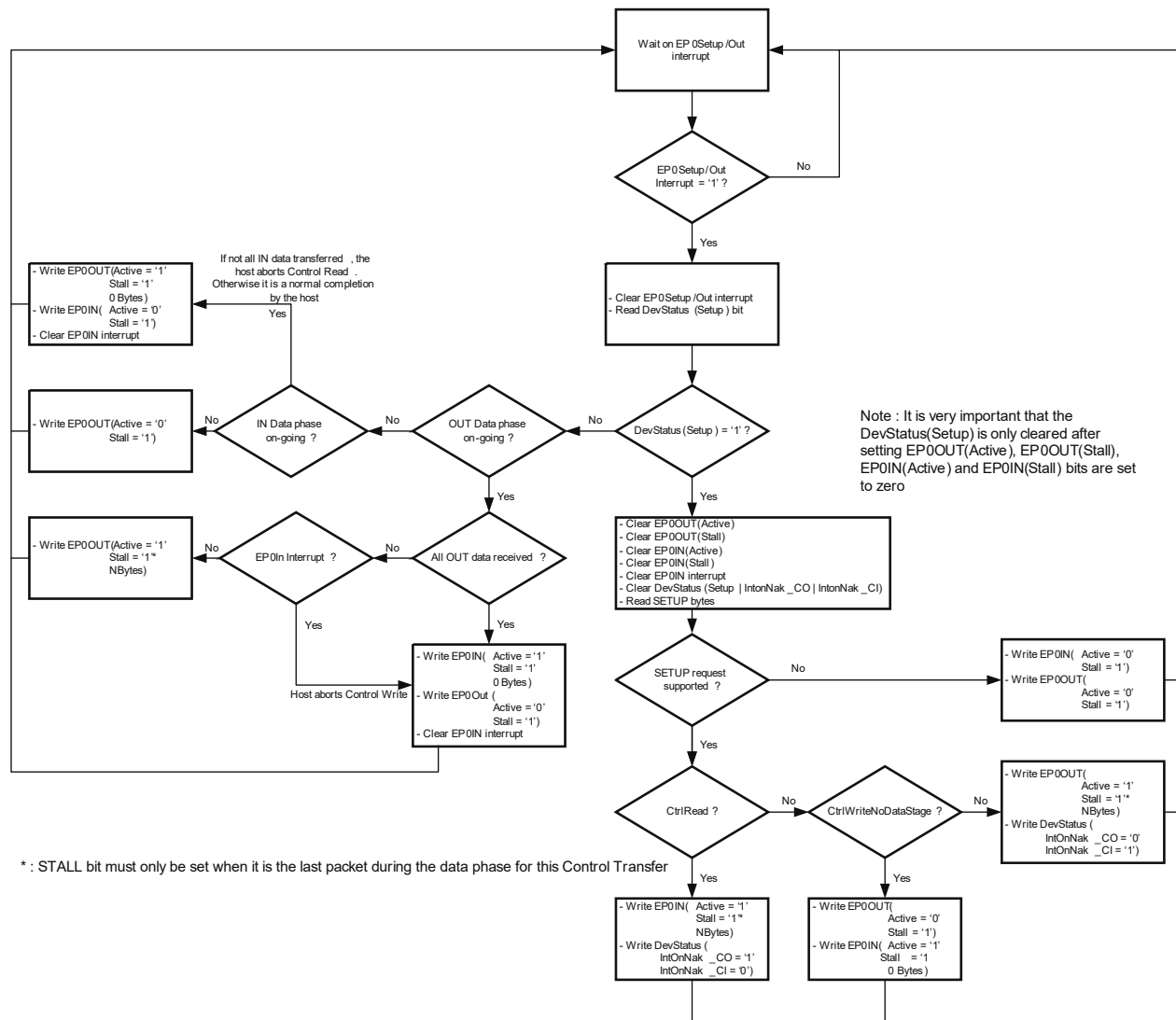
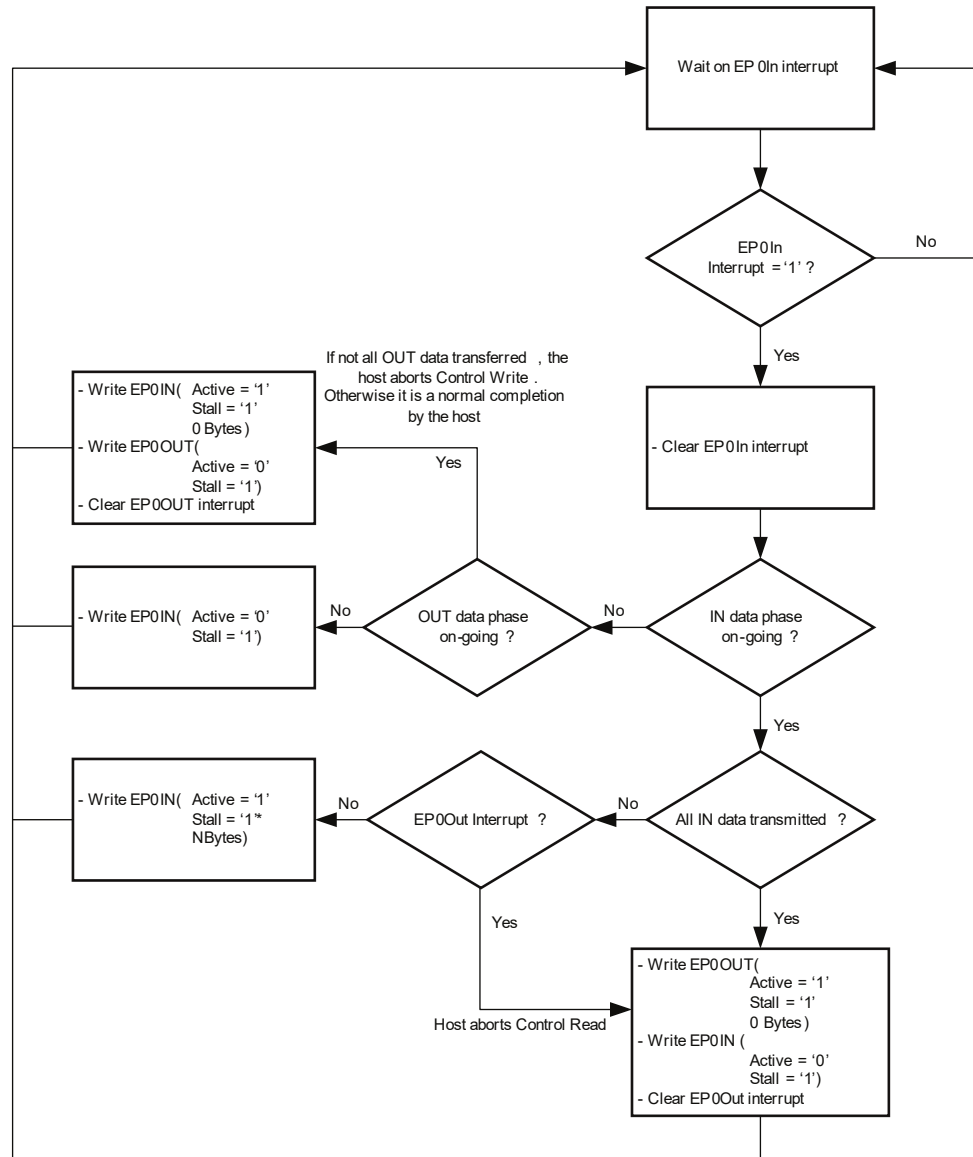


Figure 284. Flowchart of control endpoint 0 - OUT direction



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Figure 285. Flowchart of control endpoint 0 - IN direction

55.4.2 Operations

55.4.2.1 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100 mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 μ A.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up if there is transmission from the host (host-initiated wake-up). The USB controller also supports software initiated remote wake-up. To initiate remote wake-up, software on the device must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (On).

The assertion of USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for suspend.

If there is no activity for the next 2 ms, the USB1 NEEDCLK signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB NEEDCLK signal is activated. This process is fully combinatorial and hence no USB main clock is required to activate the USB NEEDCLK signal.

55.4.2.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0).
- The Active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to one.

55.4.2.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset back to 0. In order to do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

55.4.2.4 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in [Clear Feature \(endpoint halt\)](#) for every endpoint that will be used in this configuration.

For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to one.

55.4.3 Clocks

The USB device controller has the following clock connections:

- The USB main clock is a 48 MHz clock used for USB functions.
- The AHB system bus clock controls the USB device registers.

55.4.4 Interrupts

The USB controller supports an USB interrupt (USB_IRQ) and an USB activity wake-up interrupt (USB_NEEDCLK). A general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting). See [USB Interrupt Status \(INTSTAT\)](#) and [USB Interrupt Enable \(INTEN\)](#).

55.4.5 Low power modes

55.4.5.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake-up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit FORCE_NEEDCLK in [USB Device Command/Status \(DEVCMSTAT\)](#) to 0 (default) to enable automatic control of the USB_NEEDCLK signal.
2. Set DEV_EN bit in [USB Device Command/Status \(DEVCMSTAT\)](#) register to enable and then poll USB Clock Status register (See SYSCCTL) until USB_NEEDCLK goes low.
3. Poll the DSUS bit in [USB Device Command/Status \(DEVCMSTAT\)](#) register (DSUS = 1), until the USB device is suspended. The USB_NEEDCLK signal will be deasserted after another 2 ms. Poll the USB Clock Status register (See SYSCCTL) until the USB_NEEDCLK status bit is 0.
4. Clear pending USB Activity/wake-up interrupt before enabling it. Enable the USB activity wake-up interrupt in the NVIC (See NVIC).
5. Set bit 1 in the USB Clock Control register (See SYSCCTL) to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on this interrupt by enabling the USB_NEEDCLK signal in the Start enable 1 register (See SYSCCTL).
7. Enter deep-sleep mode via the power API.

The chip automatically wakes up and resumes execution on USB activity.

55.4.5.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set FORCE_NEEDCLK bit in [USB Device Command/Status \(DEVCMSTAT\)](#) register to 0 to enable automatic control of the USB_NEEDCLK signal.
2. Setup a wake-up source, for example, a PIN interrupt.
3. Force the USB clock on by writing a 1 to bit in [USB Device Command/Status \(DEVCMSTAT\)](#) register.
4. Write a 0 to the DSUS bit in [USB Device Command/Status \(DEVCMSTAT\)](#) register.
5. Wait until the USB device leaves the suspend state by polling the DSUS bit (DSUS=0) in [USB Device Command/Status \(DEVCMSTAT\)](#) register.

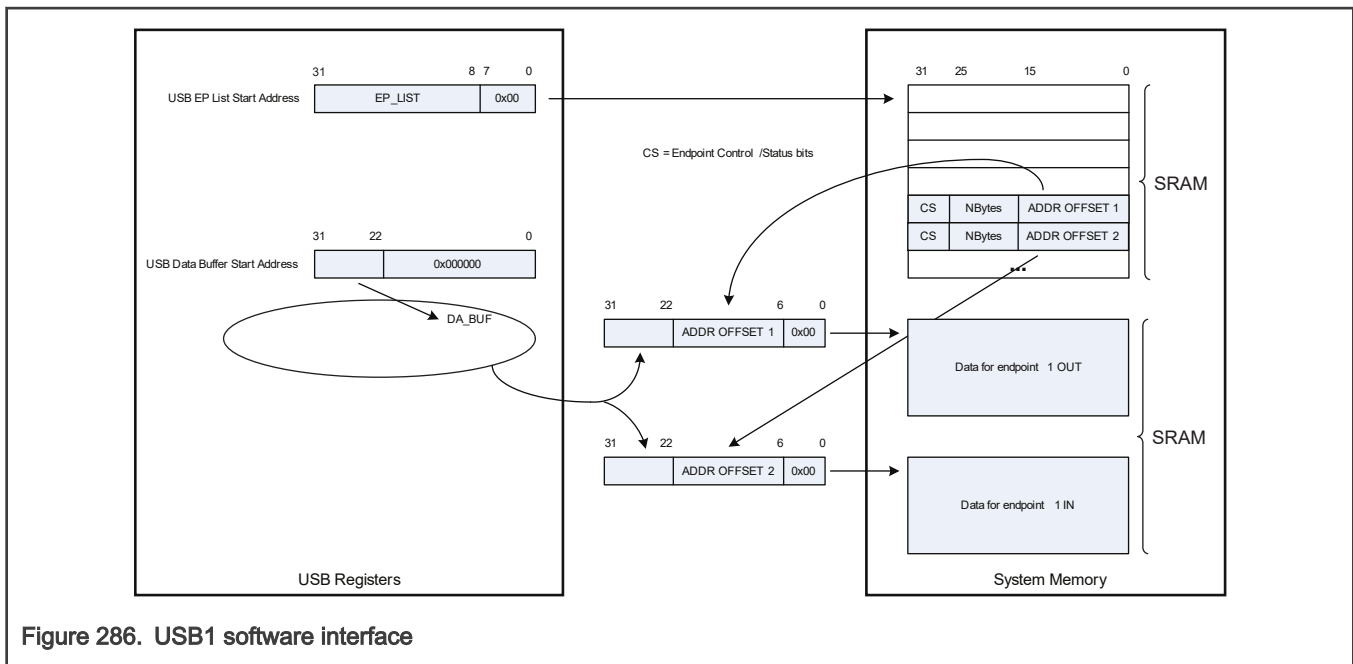
55.5 Initialization

See Chip-specific USB2.0 Device Controller information.

55.6 Application information

The PIE implements the high-speed USB protocol layer. It is completely hard-wired for speed and needs no software intervention. It handles transfer of data between the endpoint buffers in USB RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

55.6.1 USB1 software interface



55.7 Memory Map and register definition

This section includes the -T module memory map and detailed descriptions of all registers.

55.7.1 USB2.0 HS Device Controller register descriptions

55.7.1.1 USBHSD memory map

USBHSD0 base address: 4014_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	USB Device Command/Status (DEVCMSTAT)	32	RW	0000_0800h
4h	USB Info (INFO)	32	RW	2000_0000h
8h	USB EP Command/Status List Start Address (EPLISTSTART)	32	RW	4010_0000h
Ch	USB Data Buffer List Start Address (DATABUFSTART)	32	RW	0000_0000h
10h	USB Link Power Management (LPM)	32	RW	0000_0000h
14h	USB Endpoint Skip (EPSKIP)	32	RW	0000_0000h
18h	USB Endpoint Buffer in use (EPINUSE)	32	RW	0000_0000h
1Ch	USB Endpoint Buffer Configuration (EPBUFCFG)	32	RW	0000_0000h
20h	USB Interrupt Status (INTSTAT)	32	RW	0000_0000h
24h	USB Interrupt Enable (INTEN)	32	RW	0000_0000h
28h	USB Set Interrupt Status (INTSETSTAT)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
34h	USB Endpoint Toggle (EPTOGGLE)	32	R	0000_0000h

55.7.1.2 USB Device Command/Status (DEVCMSTAT)

Offset

Register	Offset
DEVCMSTAT	0h

Function

After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PHY_TEST_MODE			VBUS_DE...	Reserv ed	DRES_C	DSUS_C	DCON_C	SPEED		Reserv ed	LPM_REWP	LPM_SUS	Reserv ed	DSUS	DCON
W						W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTON_NA...	INTON_NA...	INTON_NA...	INTON_NA...	LPM_SUP	FORC_E_V...	FORC_E_N...	SETUP	DEV_EN	DEV_ADDR						
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29	PHY test mode
PHY_TEST_MODE	This field is written by firmware to put the PHY into a test mode as defined by the USB2.0 specification. 000b - Test mode disabled 001b - Test_J 010b - Test_K 011b - Test_SE0_NAK 100b - Test_Packet

Table continues on the next page...

Table continued from the previous page...

Field	Function
	101b - Test_Force_Enable
28 VBUS_DEBOUNCED	VBUS detect. This bit indicates if VBUS is detected or not. The bit raises immediately when VBUS becomes high. It drops to 0 if VBUS is low for at least 3 ms. If this bit is high and the DCon bit is set, the hardware will enable the pull-up resistor to signal a connect.
27 —	Reserved
26 DRES_C	Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a 1 to it.
25 DSUS_C	Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because • The device goes in the suspended state. • The device is disconnected. • The device receives resume signaling on its upstream port. • The bit is reset by writing a one to it.
24 DCON_C	Device status - connect change. The connect change bit is set when the pull-up resistor of the device is disconnected because VBUS disappeared. The bit is reset by writing a 1 to it.
23-22 SPEED	This field indicates the speed at which the device operates. 00b - Reserved 01b - Full-speed 10b - High-speed 11b - Reserved
21 —	Reserved
20 LPM_REWP	LPM Remote Wake-up Enabled by USB host. Hardware sets this bit to one when the RemoteWake bit in the LPM extended token is set to 1. Hardware will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.
19 LPM_SUS	Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by hardware when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μ s has elapsed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Software can only write a 0 to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to 1. 1b - When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a 0 to this bit, the device will generate a remote walk-up.
18 —	Reserved
17 DSUS	Device status suspend. The suspend bit indicates the current suspend state. 0b - When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect. 1b - It is set to 1 when the device has not seen any activity on its upstream port for more than 3 ms. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1).
16 DCON	Device status - connect. The connect bit must be set by software to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUS_DEBOUNCED bit is one.
15 INTONNAK_CI	Interrupt on NAK for interrupt and bulk OUT EP 0b - Only acknowledged packets generate an interrupt 1b - Both acknowledged and NAKed packets generate interrupts.
14 INTONNAK_CO	Interrupt on NAK for interrupt and bulk OUT EP 0b - Only acknowledged packets generate an interrupt 1b - Both acknowledged and NAKed packets generate interrupts.
13 INTONNAK_AI	Interrupt on NAK for interrupt and bulk IN EP 0b - Only acknowledged packets generate an interrupt 1b - Both acknowledged and NAKed packets generate interrupts.
12 INTONNAK_AO	Interrupt on NAK for interrupt and bulk OUT EP 0b - Only acknowledged packets generate an interrupt 1b - Both acknowledged and NAKed packets generate interrupts.
11 LPM_SUP	LPM Support 0b - LPM not supported. 1b - LPM supported.
10	Force VBUS

Table continues on the next page...

Table continued from the previous page...

Field	Function
FORCE_VBUS	If this bit is set to 1, the VBUS voltage indicators from the PHY are overruled. When this bit is set, the controller will consider the VBUS to be high and signal a connect when indicated by the other bits. When this bit is low, the real VBUS indications are taken into account by the controller.
9 FORCE_NEED CLK	Force the NEEDCLK output to always be on. 0b - USB_NEEDCLK has normal function. 1b - USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.
8 SETUP	SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by HW. SW must clear this bit by writing a one. If this bit is 0, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.
7 DEV_EN	USB device enable If this bit is set, the HW will start responding on packets for function address DEV_ADDR.
6-0 DEV_ADDR	USB Device Address After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.

55.7.1.3 USB Info (INFO)

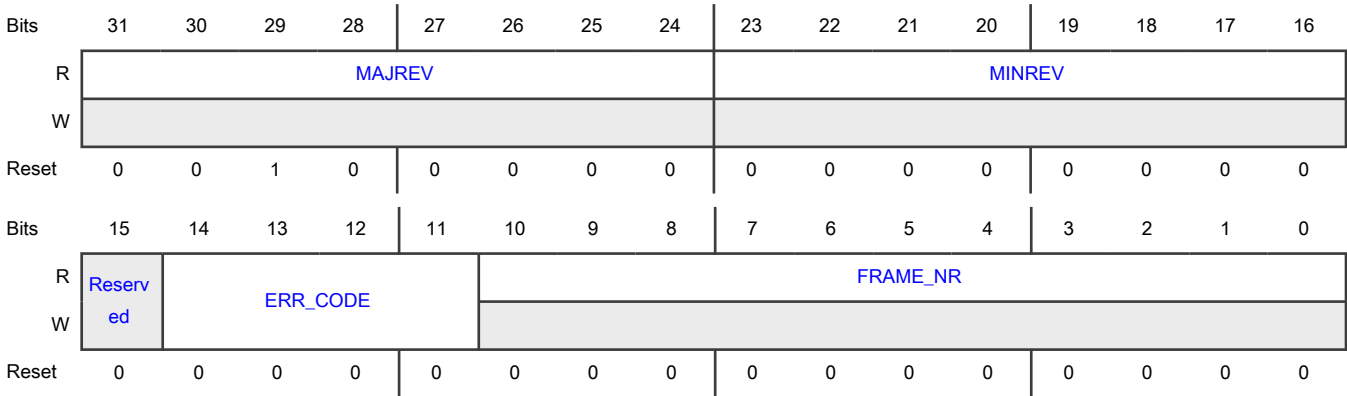
Offset

Register	Offset
INFO	4h

Function

USB Information

Diagram



Fields

Field	Function
31-24 MAJREV	Major revision
23-16 MINREV	Minor revision
15 —	Reserved
14-11 ERR_CODE	The error code which last occurred PID encoding error PID unknown Packet unexpected Token CRC error Data CRC error Time out Babble Truncated EOP Sent/Received NAK Sent Stall Overrun Sent empty packet Bitstuff error Sync error Wrong data toggle

Table continues on the next page...

Table continued from the previous page...

Field	Function
10-0 FRAME_NR	Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.

55.7.1.4 USB EP Command/Status List Start Address (EPLISTSTART)

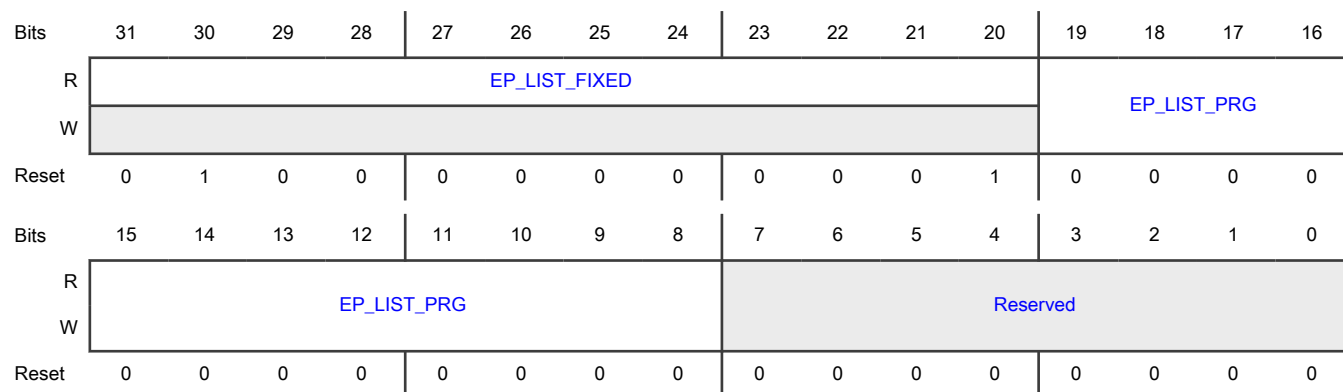
Offset

Register	Offset
EPLISTSTART	8h

Function

It indicates the start address of the USB EP Command/Status List. The USB EP Command/Status List must be placed within the USB RAM address space (16 KB, starting at address 0x4014 0000). Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256 byte boundary. Bits 19 to 8 can be programmed by software. Bits 31:20 are hard coded to 0x401, the address of the USB RAM.

Diagram



Fields

Field	Function
31-20 EP_LIST_FIXED	Fixed portion of the USB EP Command/Status List address.
19-8 EP_LIST_PRG	Programmable portion of the USB EP Command/Status List address.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 —	Reserved

55.7.1.5 USB Data Buffer List Start Address (DATABUFSTART)

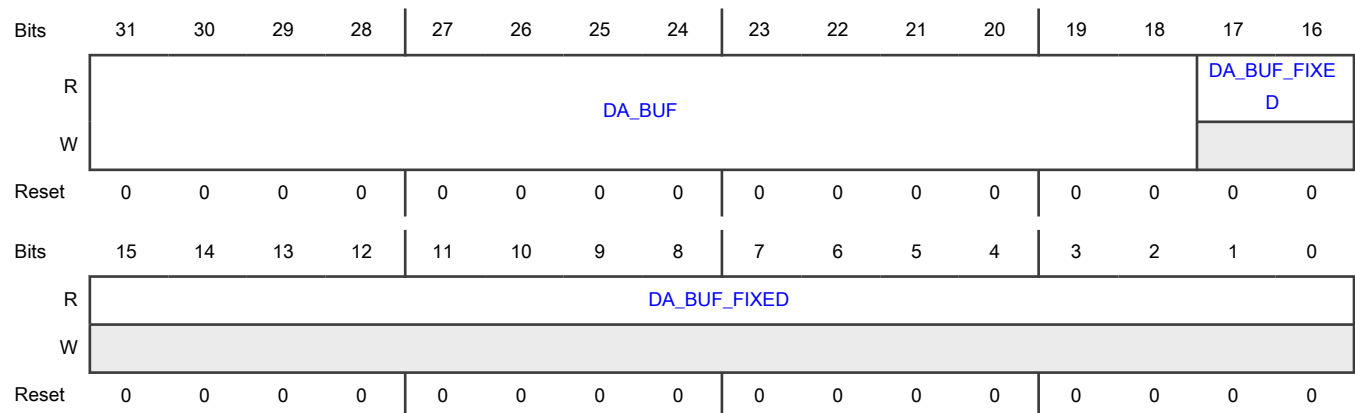
Offset

Register	Offset
DATABUFSTART	Ch

Function

This register indicates the page of the AHB address where the endpoint data is located. The endpoint data must be put in the USB RAM address space, hence the reset value of this register is the start address of the RAM, and should not be changed. The start address of each individual endpoint's buffer is an offset to the Data buffer start address. The endpoint's buffer address is set using the Address Offset field of the endpoint's corresponding entry in the "Endpoint command/status list".

Diagram



Fields

Field	Function
31-18 DA_BUF	Programmable portion of the the data buffer start address.
17-0 DA_BUF_FIXE D	Fixed portion of the data buffer start address.

55.7.1.6 USB Link Power Management (LPM)

Offset

Register	Offset
LPM	10h

Function

Link Power Management

Diagram

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

Fields

Field	Function
31-9 —	Reserved
8 DATA_PENDING	Data pending As long as this bit and LPM supported bit are both set to one, HW will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is 0, HW will return an ACK handshake on every LPM token it receives. If SW has still data pending and LPM is supported, it must set this bit to 1.
7-4 HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.
3-0 HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token.

55.7.1.7 USB Endpoint Skip (EPSKIP)

Offset

Register	Offset
EPSKIP	14h

Function

Endpoint Skip

Diagram

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

Fields

Field	Function
31-12 —	Reserved
11-0 SKIP	Endpoint skip. Writing 1 to one of these bits, will indicate to HW that it must deactivate the buffer assigned to this endpoint and return control back to software. When HW has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. NOTE In case of double buffering, HW will only clear the Active bit of the buffer indicated by the EPINUSE bit.

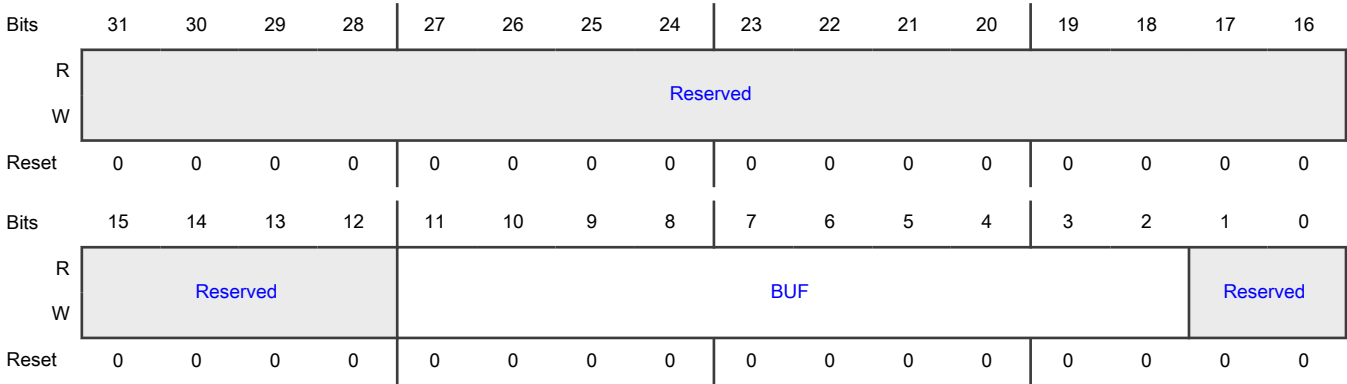
55.7.1.8 USB Endpoint Buffer in use (EPINUSE)

Offset

Register	Offset
EPINUSE	18h

Function
Endpoint Buffer in use

Diagram



Fields

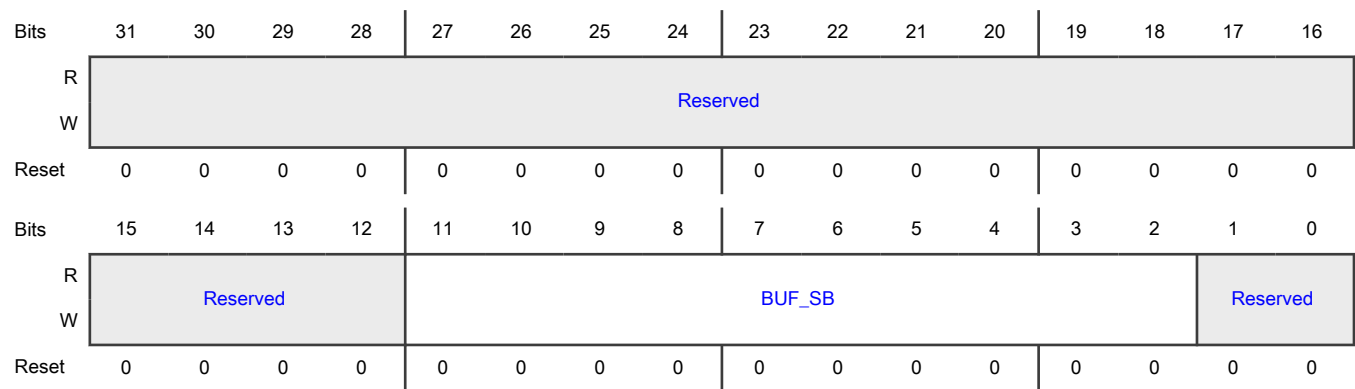
Field	Function
31-12 —	Reserved
11-2 BUF	Buffer in use. This register has one bit per physical endpoint. 00_0000_0000b - HW is accessing buffer 0. 00_0000_0001b - HW is accessing buffer 1.
1-0 —	Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.

55.7.1.9 USB Endpoint Buffer Configuration (EPBUFCFG)

Offset

Register	Offset
EPBUFCFG	1Ch

Function
Endpoint Buffer Configuration

Diagram**Fields**

Field	Function
31-12 —	Reserved
11-2 BUF_SB	Buffer in use. This register has one bit per physical endpoint. If the bit is set to single buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the Active bit. If the bit is set to double buffer (1), HW will toggle the EPINUSE bit when it clears the Active bit for the buffer. 00_0000_0000b - Single buffer. 00_0000_0001b - Double buffer.
1-0 —	Fixed to 0 because the control endpoint 0 is fixed to single buffering for each physical endpoint.

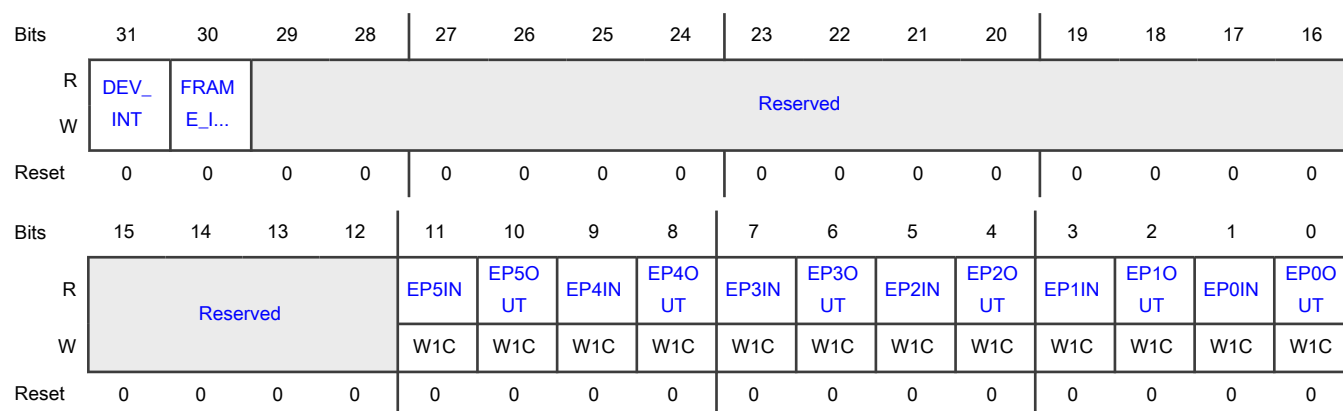
55.7.1.10 USB Interrupt Status (INTSTAT)**Offset**

Register	Offset
INTSTAT	20h

Function

Interrupt Status

Diagram



Fields

Field	Function
31 DEV_INT	Device status interrupt. This bit is set by HW when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.
30 FRAME_INT	Frame interrupt. This bit is set to one every millisecond when the VBUS_DEBOUNCED bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.
29-12 —	Reserved
11 EP5IN	Control EP5 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP5 OUT direction. Software can clear this bit by writing a one to it.
10 EP5OUT	Control EP5 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP5 OUT direction. Software can clear this bit by writing a one to it.
9 EP4IN	Control EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP4 OUT direction. Software can clear this bit by writing a one to it.
8	Control EP4 OUT direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
EP4OUT	This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP4 OUT direction. Software can clear this bit by writing a one to it.
7 EP3IN	Control EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP3 OUT direction. Software can clear this bit by writing a one to it.
6 EP3OUT	Control EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP3 OUT direction. Software can clear this bit by writing a one to it.
5 EP2IN	Control EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP2 OUT direction. Software can clear this bit by writing a one to it.
4 EP2OUT	Control EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP2 OUT direction. Software can clear this bit by writing a one to it.
3 EP1IN	Control EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP1 OUT direction. Software can clear this bit by writing a one to it.
2 EP1OUT	Control EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to 0 or the skip bit is set by software. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP1 OUT direction. Software can clear this bit by writing a one to it.
1 EP0IN	Control EP0 IN direction. This bit will be set if NBytes transitions to 0 or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.
0	Control EP0 OUT direction.

Table continues on the next page...

Table continued from the previous page...

Field	Function
EP0OUT	This bit will be set if NBytes transitions to 0 or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.

55.7.1.11 USB Interrupt Enable (INTEN)

Offset

Register	Offset
INTEN	24h

Function

Interrupt Enable

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEV_I	FRAM	Reserved													
W	NT...	E_I...														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				EP_INT_EN											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31	Device status interrupt.
DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.
30	Frame interrupt.
FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.
29-12 —	Reserved
11-0	End Point Interrupt Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line.

55.7.1.12 USB Set Interrupt Status (INTSETSTAT)

Offset

Register	Offset
INTSETSTAT	28h

Function

Set Interrupt Status

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEV_S	FRAM	Reserved													
W	ET...	E_S...														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				EP_SET_INT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 DEV_SET_INT	Device status interrupt. If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.
30 FRAME_SET_INT	Frame interrupt. If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.
29-12 —	Reserved
11-0 EP_SET_INT	End Point Set Interrupt Enable. If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.

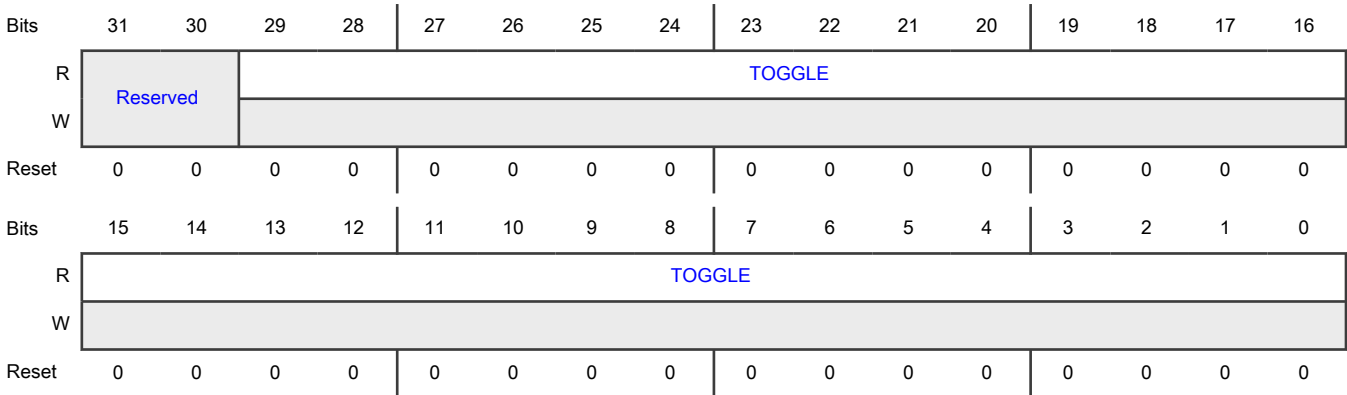
55.7.1.13 USB Endpoint Toggle (EPTOGGLE)

Offset

Register	Offset
EPTOGGLE	34h

Function
Endpoint Toggle

Diagram



Fields

Field	Function
31-30 —	Reserved
29-0 TOGGLE	Endpoint data toggle. This field indicates the current value of the data toggle for the corresponding endpoint.

Chapter 56

USB Device Charger Detection Module (USBHSDCD)

56.1 Chip-specific USBDCD information

Table 574. Reference links to related information

Topic	Related module	Reference
Full description	USBDCD	USBHSDCD
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

56.1.1 Module instances

This device has one instance of the USBHSDCD module, USBHSDCD0.

56.2 Overview

The USB Device Charger Detection (USBDCD) module works with the USB transceiver to detect whether the USB device is attached to a Charging Port, either a Dedicated Charging Port (DCP) or a Charging Downstream Port (CDP). System software coordinates the detection activities of the module and controls an off-chip integrated circuit that performs the battery charging.

There can be separate instantiations of the USBDCD module for each USB port. The instantiation for the High-Speed(HS) capable USB port is named as USBHSDCD. The instantiation for Full-Speed(FS)/Low-Speed(LS) USB port is named as USBDCD. See the chip-specific information for the specific instantiation of your device.

The use of the term "USB transceiver" in this module documentation applies to the USB physical layer instance on the chip, whether it is an FS/LS only transceiver or an HS/FS/LS capable PHY.

56.2.1 Block diagram

The following figure is a high level block diagram of the module.

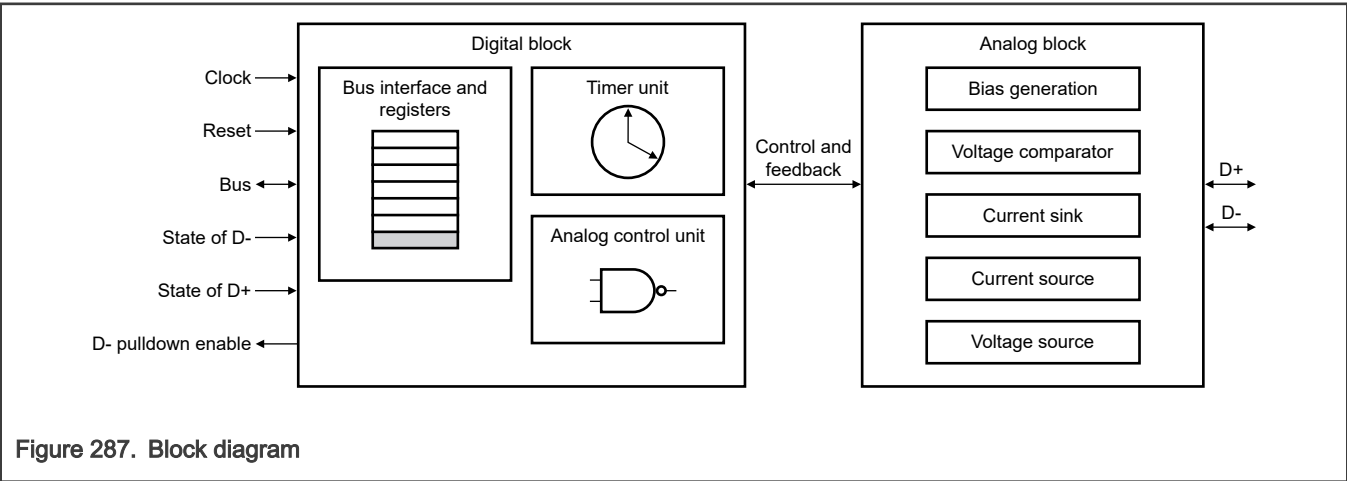


Figure 287. Block diagram

The USBDCD module consists of two main blocks:

- A Digital block provides the programming interface (memory-mapped registers) and includes the Timer unit and the Analog control unit.
- An Analog block provides the circuitry for the physical detection of the charger that includes the Bias generation, Voltage source, Current source, Current sink, and Voltage comparator circuitry. This block also contains necessary muxes between the source/sink circuits and the D- and D+ pins.

56.2.2 Features

The USBDCD module offers the following features:

- Compliant with the latest industry-standard specifications: *USB Battery Charging Specification, Revisions 1.1 and 1.2*
- Programmable timing parameters are by default set to values required by the industry standards:
 - Standard default values allows easy configuration - You only need to set the clock frequency before enabling the module.
 - Updates of the standards are programmable.

56.3 Functional description

Several hardware components are involved in the sequence of detecting the presence of the charging port and type of the charging port. System software coordinates this activity. This collection of interacting hardware and software is called USB battery charging subsystem. The following figure shows the USBDCD module as a component of the subsystem. The following table describes the components.

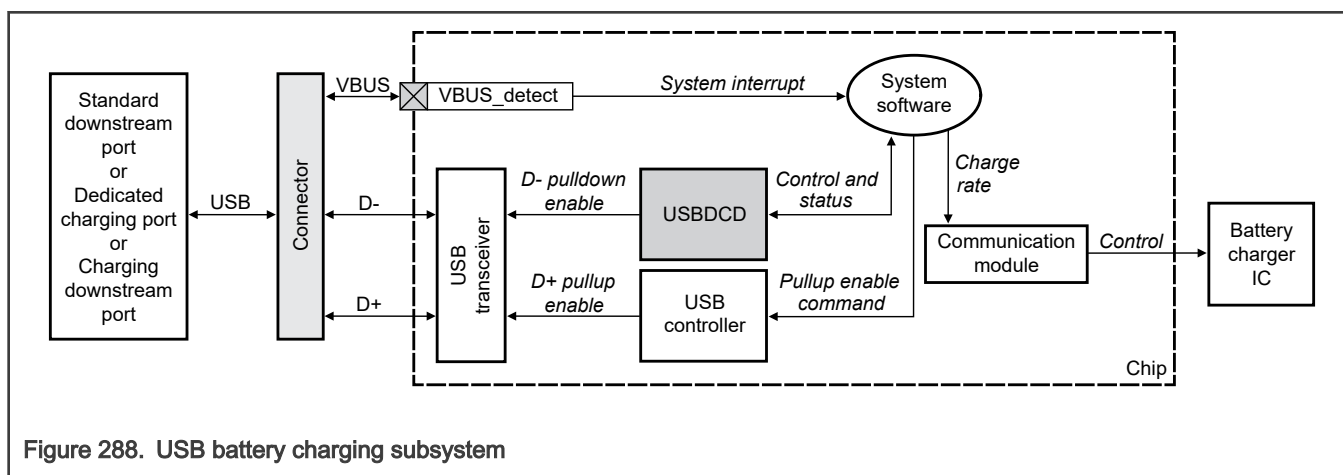


Figure 288. USB battery charging subsystem

Table 575. USB battery charger subsystem components

Component	Description	
Battery Charger IC	The external battery charger IC regulates the charge rate to the rechargeable battery. System software is responsible for communicating the appropriate charge rates.	
	Charger	Maximum current drawn
	Standard Downstream Port (SDP)	up to 500 mA (I_{CFG_MAX}) ¹
	Charging Downstream Port (CDP)	up to 1500 mA (I_{DEV_CHG})
	Dedicated Charging Port (DCP)	up to 1500 mA (I_{DEV_CHG})

Table 575. USB battery charger subsystem components (continued)

Component	Description
	1. If a USB SDP host has suspended the USB device, the current drawn by the device has to follow the rules of the Dead Battery Provision in the <i>USB Battery Charging Specification, Rev. 1.2</i> .
Comm Module	A communications module on the device can be used to control the charge rate of the battery charger IC.
System software	Coordinates the detection activities of the subsystem.
USB Controller	The D+ pullup enable control signal plays a role during the charger type detection phase. System software must issue a command to the USB controller to assert this signal. After this pullup is enabled, the device is considered to be connected to the USB bus. The host then attempts to enumerate it. NOTE In general, the USB controller is used only for USB device applications when using the USBDCD module. For USB host applications, the USBDCD module must be disabled.
USB Transceiver	The USB transceiver contains the pullup resistor for the USB D+ signal and the pulldown resistors for the USB D+ and D– signals. The D+ pullup and the D– pulldown are both used during the charger detection sequence in BC1.1, but it is not used during charger detection in BC1.2. The USB transceiver also outputs the digital state of the D+ and D– signals from the USB bus. The pullup and pulldown enable signals are controlled by other modules during the charger detection sequence in BC1.1: The D+ pullup enable is output from the USB controller and is under software control. The USBDCD module controls the D– pulldown enable.
USBDCD Module	Detects whether the device has been plugged into either an SDP, a CDP, or a DCP.
VBUS_detect	This interrupt pin connected to the USB VBUS signal detects when the device has been plugged into or unplugged from the USB bus. If the system requires waking up from a low power mode on being plugged into the USB port, this interrupt should also be a low power wake up source. If this pin multiplexes other functions, such as GPIO, the pin can be configured as an interrupt so that the USB plug or unplug event can be detected.

56.3.1 Modes of operation

The operating modes of the USBDCD module are shown in the following table.

Table 576. Module modes and their conditions

Module mode	Description	Conditions when used
Enabled for Charger Detection	The module performs the charger detection sequence.	System software must enable the module only if all of the following conditions are true: The system is using a rechargeable battery or is otherwise capable of being powered up by an SDP or a Charging Port.

Table continues on the next page...

Table 576. Module modes and their conditions (continued)

Module mode	Description	Conditions when used
		- The device is being used in a USB device application. - The device is attached to the USB cable.
Enabled for Signal Overrides	Module bias circuits are enabled for special signaling on DP/DM pins.	See description for SIGNAL_OVERRIDE[PS] bit field.
Disabled	The module is not active and is held in a low-power state.	System software must disable the module when either of the following conditions is true: - The charger detect sequence is complete. - The conditions for being enabled are not met.
Powered Off	The digital supply voltage dvdd is removed.	Low system performance requirements allow putting the device into a very low-power stop mode.

Operating mode transitions are shown in the following table.

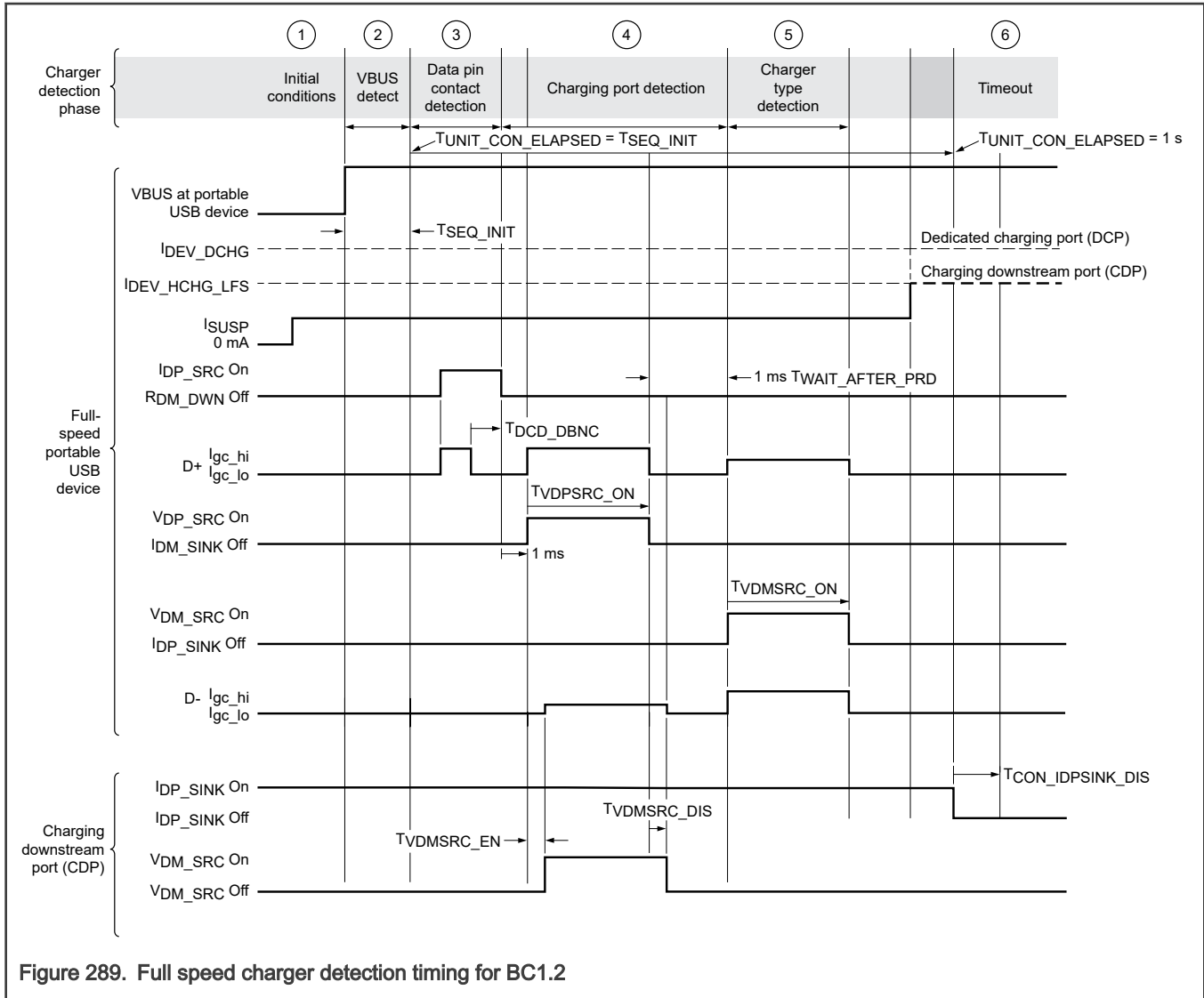
Table 577. Entering and exiting module modes

Module mode	Entering	Exiting	Mode after exiting
Enabled	Set CONTROL[START].	Set CONTROL[SR]. ¹	Disabled
Disabled	Take <i>either</i> of the following actions: - Set CONTROL[SR].¹ - Reset the module. By default, the module is disabled.	Set CONTROL[START].	Enabled
Powered Off	Perform the following actions: - Put the device into very low-power stop mode. - Adjust the supply voltages.	Perform the following actions: - Restore the supply voltages. - Take the device out of very low-power stop mode.	Disabled

1. The effect of setting the SR bit is immediate; that is, the module is disabled even if the sequence has not completed.

56.3.2 The charger detection sequence

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the *USB Battery Charging Specification, Rev. 1.2*.



Timing parameter values used in this module for BC1.2 are listed in the following table.

Table 578. Timing parameters for the charger detection sequence for BC1.2

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{DCD_DBNC} ¹	10 ms min (no max)	10 ms	0– 1023 ms
T_{VDPSRC_ON} ¹	40 ms min (no max)	40 ms	0 –1023 ms
$T_{WAIT\_AFTER\_PRD}$	N/A	1 ms	0– 1023 ms
T_{VDMSRC_ON}	40 ms	40 ms	0 –1023 ms
T_{SEQ_INIT}	N/A	16 ms	0 –1023 ms
T_{UNIT_CON} ¹	1 s	N/A	N/A

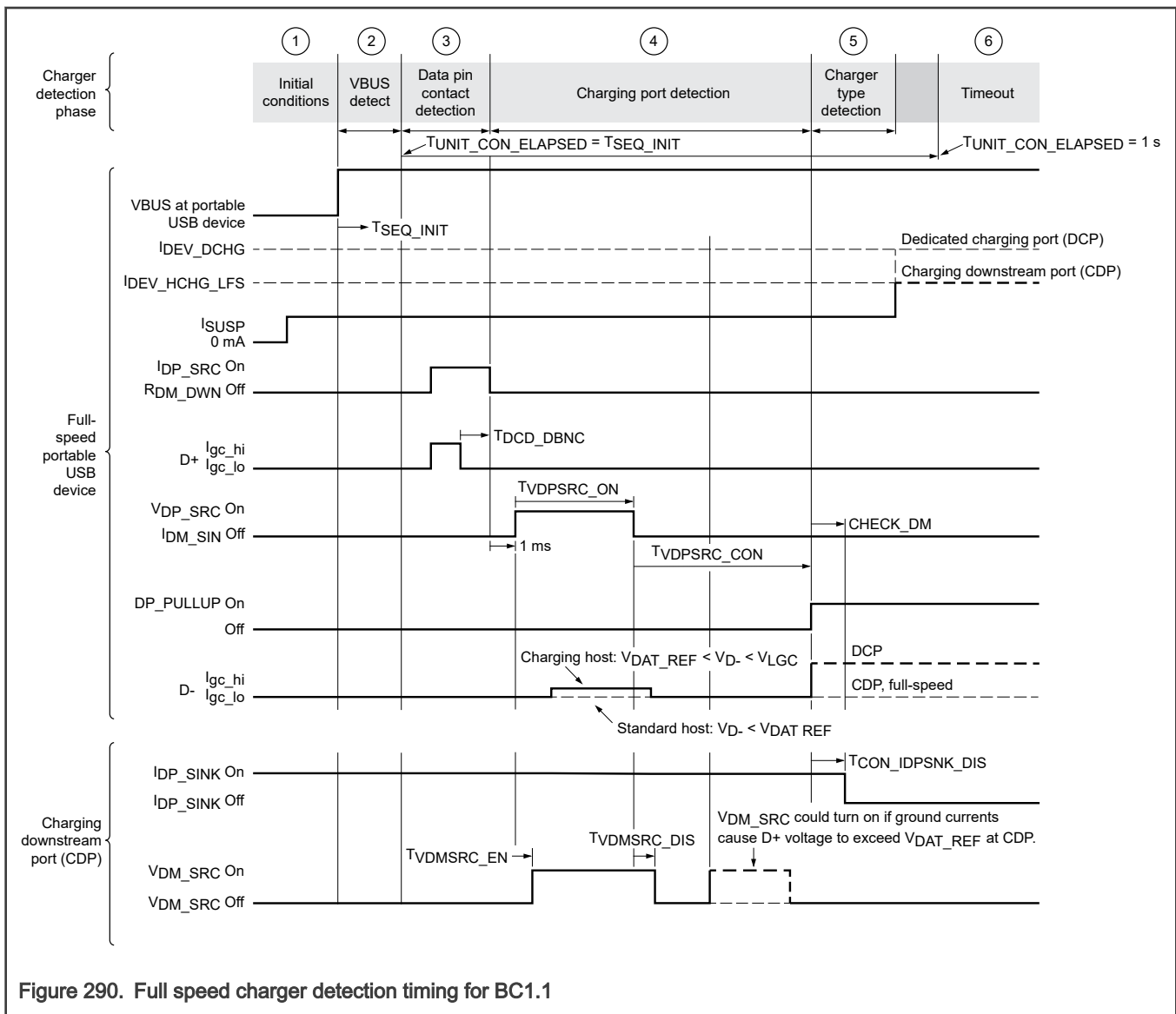
Table continues on the next page...

Table 578. Timing parameters for the charger detection sequence for BC1.2 (continued)

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{VDMSRC_EN} ¹	1– 20 ms	From the USB host	N/A
T_{VDMSRC_DIS} ¹	0 – 20 ms	From the USB host	N/A
$T_{CON\_IDPSINK\_DIS}$ ¹	0 – 10 ms	From the USB host	N/A

1. This parameter is defined by the *USB Battery Charging Specification, Rev. 1.2*.

The following figure illustrates the charger detection sequence in a simplified timing diagram based on the *USB Battery Charging Specification, Rev. 1.1*.



Timing parameter values used in this module for BC1.1 are listed in the following table.

Table 579. Timing parameters for the charger detection sequence for BC1.1

Parameter	USB Battery Charging Spec	Module default	Module programmable range
T_{DCD_DBNC} ¹	10 ms min (no max)	10 ms	0– 1023 ms
T_{VDPSRC_ON} ¹	40 ms min (no max)	40 ms	0 –1023 ms
T_{VDPSRC_CON} ¹	40 ms min (no max)	40 ms	0 –1023 ms
CHECK_DM	N/A	1 ms	0– 15 ms
T_{SEQ_INIT}	N/A	16 ms	0 –1023 ms
T_{UNIT_CON} ¹	1 s	N/A	N/A
T_{VDMSRC_EN} ¹	1– 20 ms	From the USB host	N/A
T_{VDMSRC_DIS} ¹	0 –20 ms	From the USB host	N/A
$T_{CON\_IDPSINK\_DIS}$ ¹	0– 20 ms	From the USB host	N/A

1. This parameter is defined by the *USB Battery Charging Specification, Rev. 1.1*.

The following table provides an overview description of the charger detection sequence shown in the preceding figure.

Table 580. Overview of the charger detection sequence

Phase		Overview description	Full description
1	Initial conditions	Initial system conditions that need to be met before the detection sequence is initiated.	Initial system conditions
2	VBUS detection	System software detects contact of the VBUS signal with the system interrupt pin VBUS_detect.	VBUS contact detection
3	Data pin contact detection	The USBDCD module detects that the USB data pins D+ and D– have made contact with the USB port.	Data pin contact detection
4	Charging port detection	The USBDCD module detects if the port is an SDP or either type of charging port, that is CDP or DCP.	Charging port detection
5	Charger type detection	The USBDCD module detects the type of charging port, if applicable.	Charger type detection
6	Sequence timeout	The USBDCD module did not finish the detection sequence within the timeout interval. The sequence will continue until halted by software.	Charger detection sequence timeout

56.3.2.1 Initial system conditions

The USBDCD module is intended for use with USB device applications using a rechargeable battery. The module does not have support for interfacing with ACA or ACA-Dock equipment as defined in the *USB Battery Charging Specification, Revision 1.2*. It cannot be used with USB applications that are embedded host or OTG.

In addition, before the USBDCD module's charger detection sequence can be initiated, the system must be:

- Powered-up and in run mode. The USBDCD instantiation of this module for the High-Speed port does not directly depend on the VBUS voltage and can operate as long as the avdd33 supply is in a valid range.
- Recently plugged into a USB port.
- Drawing not more than 2.5 mA total system current from the USB bus, except as allowed by the *USB 2.0 Connect Timing Update ECN*.

Examples of allowable precursors to this set of initial conditions include:

- A powered-down device is subsequently powered-up upon being plugged into the USB bus.
- A device in a low-power mode subsequently enters run mode upon being plugged into the USB bus.

56.3.2.2 VBUS contact detection

Once the device is plugged into a USB port, the VBUS_detect system interrupt is triggered. System software must do the following to initialize the module and start the charger detection sequence:

1. Restore power if the module is powered-off.
2. Set CONTROL[SR] to initiate a software reset.
3. Configure the USBDCD module by programming the CLOCK register and the timing parameters as needed.
4. Set CONTROL[IE] to enable interrupts, or clear the bit if the software polling method is used.
5. Program CONTROL[BC12] based on which revision of the USB Battery Charging Specification is needed.
6. Set CONTROL[START] to start the charger detection sequence.

56.3.2.3 Data pin contact detection

The module must ensure that the data pins have made contact because the detection sequence depends upon the state of the USB D+ and D- signals. USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. See the following figure.

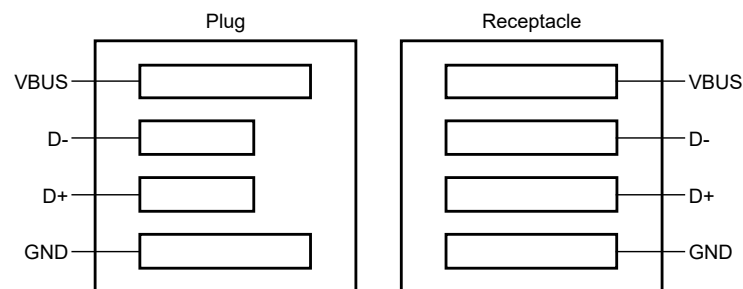


Figure 291. Relative pin positions in USB plugs and receptacles

As a result, when a portable USB device is attached to an upstream port, the portable USB device detects VBUS before the data pins have made contact. The time between power pins and data pins making contact depends on how fast the plug is inserted into the receptacle. Delays of several hundred milliseconds are possible.

56.3.2.3.1 Debouncing the data pin contact

When system software has initiated the charger detection sequence, as described in [Initial system conditions](#), the USBDCD module turns on the I_{DP_SRC} current source and enables the R_{DM_DWN} pulldown resistor. If the data pins have not made contact, the D+ line remains high. After the data pins make contact, the D+ line goes low and debouncing begins.

After the D+ line goes low, the module continuously samples the D+ line over the duration of the T_{DCD_DBNC} debounce time interval. By default, T_{DCD_DBNC} is 10 ms, but it can be programmed in the `TIMER0[TDCCD_DBNC]` field. See the description of the `TIMER0` register for details.

When it has remained low for the entire interval, the debouncing is complete. However, if the D+ line returns high during the debounce interval, the module waits until the D+ line goes low again to restart the debouncing. This cycle repeats until either of the following happens:

- The data pin contact has been successfully debounced (see [Success in detecting data pin contact \(phase completion\)](#)).
- A timeout occurs (see [Charger detection sequence timeout](#)).

56.3.2.3.2 Success in detecting data pin contact (phase completion)

After successfully debouncing the D+ state, the module does the following:

- Updates the `STATUS` register to reflect phase completion (See [Table 585](#) for field values.)
- Directly proceeds to the next step in the sequence: detection of a charging port (See [Charging port detection](#).)

56.3.2.4 Charging port detection

After it detects that the data pins have made contact, the module waits for a fixed delay of 1 ms, and then attempts to detect whether it is plugged into a charging port. The module connects the following analog units to the USB D+ or D– lines during this phase:

- The voltage source V_{DP_SRC} connects to the D+ line
- The current sink I_{DM_SINK} connects to the D– line
- The voltage comparator connects to the USB D– line, comparing it to the voltage V_{DAT_REF} .

After a time of T_{VDPSRC_ON} , the module samples the D– line. The T_{VDPSRC_ON} parameter is programmable and defaults to 40 ms. After sampling the D– line, the module disconnects the voltage source, current sink, and comparator.

The next steps in the sequence depend on the voltage on the D– line as determined by the voltage comparator. See the following table.

Table 581. Sampling D– in the charging port detection phase

If the voltage on D– is...	Then...	See...
Below V_{DAT_REF}	The port is an SDP that does not support using charging currents above I_{CFG_MAX} .	Standard Downstream Port (SDP)
Above V_{DAT_REF} but below V_{LGC}	The port is a charging port.	Charging port
Above V_{LGC}	This is an error condition.	Error in charging port detection

56.3.2.4.1 Standard Downstream Port (SDP)

As part of the charger detection handshake with a standard USB host, the module does the following without waiting for the interval ($T_{WAIT\_AFTER\_PRD}$ or T_{VDPSRC_CON}) to elapse:

- Updates the `STATUS` register to reflect that a Standard Downstream Port (SDP) is detected with `SEQ_RES = 01`. See [Table 585](#) for field values.
- Sets `CONTROL[IF]`.
- Generates an interrupt if enabled in `CONTROL[IE]`.

At this point, control is passed to system software via the interrupt. The rest of the sequence, which detects the type of charging port, is not applicable, so software must perform the following steps:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see [Table 575](#).

56.3.2.4.2 Charging port

As part of the charger detection handshake with any type of USB host, the module waits until the interval ($T_{\text{WAIT\_AFTER\_PRD}}$ or $T_{\text{VDPSRC\_CON}}$) has elapsed before it does the following:

For *USB Battery Charging Specification, Rev. 1.2*:

- Enables $V_{\text{DM\_SRC}}$.
- Updates the STATUS register to reflect that a charging port is detected with SEQ_RES = 10. See [Table 585](#) for field values.
- At this point the detection sequence progresses to [Charger type detection](#) without needing software interaction.

For *USB Battery Charging Specification, Rev. 1.1*:

- Updates the STATUS register to reflect that a charging port is detected with SEQ_RES = 10. See [Table 585](#) for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].
- At this point, control is passed to system software via the interrupt. Software must:
 1. Read the STATUS register.
 2. Set CONTROL[IACK] to acknowledge the interrupt.
 3. Issue a command to the USB controller to pullup the USB D+ line.
 4. Wait for the module to complete the final phase of the sequence. See [Charger type detection](#).

56.3.2.4.3 Error in charging port detection

For this error condition, the module does the following:

- Updates the STATUS register to reflect the error with SEQ_RES = 00. See [Table 585](#) for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

Note that in this case the module does not wait for the interval ($T_{\text{WAIT\_AFTER\_PRD}}$ or $T_{\text{VDPSRC\_CON}}$) to elapse.

At this point, control has been passed to system software via the interrupt. The rest of the sequence (detecting the type of charging port) is not applicable, so software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.

56.3.2.5 Charger type detection

For *USB Battery Charging Specification, Rev. 1.2*:

After the USBDCD module enables the V_{DM_SRC} , the module starts the T_{VDMSRC_ON} timer counting down the time interval programmed into the $TIMER2[T_{VDMSRC\_ON}]$ field.

Once the T_{VDMSRC_ON} timer is elapsed, the module samples the USB D+ line to determine the type of charger. See the following table.

Table 582. Sampling D+ in the charger type detection phase (BC1.2)

If the voltage on D+ is...	Then...	See...
High ($D+ > V_{DAT\_REF}$)	The port is a DCP. ¹	Dedicated Charging Port (DCP)
Low ($D+ < V_{DAT\_REF}$)	The port is a CDP. ²	Charging Downstream Port (CDP)

1. In a DCP, the D+ and D– lines are shorted together through a small resistor.

2. In a CDP, the D+ and D– lines are not shorted.

For *USB Battery Charging Specification, Rev. 1.1*:

After software enables the D+ pullup resistor, the module is notified automatically (via internal signaling) to start the $CHECK\_DM$ timer counting down the time interval programmed in the $TIMER2[CHECK\_DM]$ field.

After the $CHECK\_DM$ time is elapsed, the module samples the USB D– line to determine the type of charger. See the following table.

Table 583. Sampling D– in the charger type detection phase (BC1.1)

If the voltage on D– is...	Then...	See...
High	The port is a DCP. ¹	Dedicated Charging Port (DCP)
Low	The port is a CDP. ²	Charging Downstream Port (CDP)

56.3.2.5.1 Dedicated Charging Port (DCP)

For a Dedicated Charging Port (DCP), the module does the following:

- Updates the STATUS register to reflect that a dedicated charging port is detected with $SEQ\_RES = 11$. See [Table 585](#) for field values.
- Sets $CONTROL[IF]$.
- Generates an interrupt if enabled in $CONTROL[IE]$ bit.

The *USB Battery Charging Specification, Rev. 1.2* indicates that additionally if the detection sequence determines an attachment to a DCP, then the USB device should signal on the D+ pin by either enabling the D+ pullup resistor or enabling V_{DP_SRC} .

At this point, control has been passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Disable the USB controller to prevent transitions on the USB D+ or D– lines from causing spurious interrupt or wakeup events to the system.
3. Set $CONTROL[IACK]$ to acknowledge the interrupt.

4. Set CONTROL[SR] to issue a software reset to the module.
5. Disable the module.
6. Enable signaling on the D+ pin. This can be done by enabling the Transceiver D+ pullup resistor through configuration of the register bit fields in the USB controller instance for this USB port. On this product, this signaling can alternatively be done by enabling V_{DP_SRC} through setting the SIGNAL_OVERRIDE[PS] bit field to a value of 2'b10.
7. Communicate the appropriate charge rate to the external battery charger IC; see [Table 575](#).

56.3.2.5.2 Charging Downstream Port (CDP)

For a Charging Downstream Port (CDP), the module does the following:

- Updates the STATUS register to reflect that a CDP is detected with SEQ_RES = 10. See [Table 585](#) for field values.
- Sets CONTROL[IF].
- Generates an interrupt if enabled in CONTROL[IE].

At this point, control is passed to system software via the interrupt. Software should:

1. Read the STATUS register.
2. Set CONTROL[IACK] to acknowledge the interrupt.
3. Set CONTROL[SR] to issue a software reset to the module.
4. Disable the module.
5. Communicate the appropriate charge rate to the external battery charger IC; see [Table 575](#).

56.3.2.6 Charger detection sequence timeout

The maximum time allowed to connect according to the *USB Battery Charging Specification* is one second. If the Unit Connection Timer reaches the one second limit and the sequence is still running as indicated by the STATUS[ACTIVE] bit, the module does the following:

- Updates the STATUS register to reflect that a timeout error has occurred. See [Table 585](#) for field values.
- Sets the CONTROL[IF] bit.
- Generates an interrupt if enabled in CONTROL[IE].
- The detection sequence continues until explicitly halted by software setting the CONTROL[SR] bit.
- The Unit Connection Timer continues counting. See the description of the TIMER0 Register.

At this point, control is passed to the system software via the interrupt, which has two options: ignore the interrupt and allow more time for the sequence to complete, or halt the sequence. To halt the sequence, software must:

1. Read the STATUS register.
2. Set the CONTROL[IACK] bit to acknowledge the interrupt.
3. Set the CONTROL[SR] bit to issue a software reset to the module.
4. Disable the module.

This timeout function is also useful in case software does not realize that the USB device is unplugged from USB port during the charger detection sequence. If the interrupt occurs but the V_{BUS_DETECT} input is low, software can disable and reset the module.

System software might allow the sequence to run past the timeout interrupt under these conditions:

1. The USB Battery Charging Spec is amended to allow more time. In this case, software should poll TIMER0[T_UNITCON] periodically to track elapsed time after 1s; or
2. For debug purposes.

Note that the $T_{UNITCON}$ register field will stop incrementing when it reaches its maximum value so it will not rollover to zero and start counting up again.

56.3.2.7 Dead Battery Provision signaling

If the system needs to operate under the Dead Battery Provision of the *USB Battery Charging Specification, Revision 1.2*, the USBDCD module must be configured to signal V_{DP_SRC} on the USB_DP pin.

For information on Dead Battery Provision conditions and signaling, see the [Dead or weak battery](#) later in this chapter.

56.3.3 Resets

There are two ways to reset various register contents in this module: hardware resets and a software reset.

56.3.3.1 Hardware resets

Hardware resets originate at the system or device level and propagate down to the individual module level. They include start up reset, low-voltage reset, and all other hardware reset sources.

Hardware resets cause the register contents to be restored to their default state as listed in the register descriptions.

56.3.3.2 Software reset

A software reset re-initializes the module's status information, but leaves configuration information unchanged. The software reset allows software to prepare the module without needing to reprogram the same configuration each time the USB device is plugged into a USB port.

Setting CONTROL[SR] initiates a software reset. The following table shows all register fields that are reset to their default values by a software reset.

Table 584. Software reset and register fields affected

Register	Fields affected	Fields not affected
CONTROL ¹	IF	IE, START
STATUS	All	None
CLOCK	None	All
TIMER n	TUNITCON	All other

1. CONTROL[SR] and CONTROL[IACK] are self-clearing.

A software reset also returns all internal logic, timers, and counters to their reset states. If the module is already active (STATUS[ACTIVE] = 1), a software reset stops the sequence.

NOTE

Software must always initiate a software reset before starting the sequence to ensure the module is in a known state.

56.3.4 Interrupts

The USBDCD module has an interrupt to alert system software of certain events, which are listed in the following table. All events except the Phase Complete event for the Data Pin Detection phase can trigger an interrupt.

Table 585. Events triggering an interrupt by sequence phase

Sequence phase	Event	Event description	STATUS fields ¹	Phase description
Data Pin Detection	Phase Complete	The module has detected data pin contact. <i>No interrupt occurs: CONTROL[IF] = 0.</i>	ERR = 0 SEQ_STAT = 01 SEQ_RES = 00 TO = 0	VBUS contact detection
Charging Port Detection	Phase Complete	The module has completed the process of identifying if the USB port is a charging port or not. NOTE: An interrupt is triggered at completion of this phase for BC 1.1 operation whether an SDP or a Charging port is detected. For BC 1.2 operation, if an SDP is detected an interrupt is triggered but if a Charging port is detected the sequence goes directly to the Charger Type Detection phase.	ERR = 0 SEQ_STAT = 10 SEQ_RES = 01 or 10 TO = 0	Charging port detection
	Error	The module cannot identify the type of port because the D- line is above the USB's VLGC threshold.	ERR = 1 SEQ_STAT = 10 SEQ_RES = 00 TO = 0	Error in charging port detection
Charger Type Detection	Phase Complete	The module has completed the process of identifying the charger type detection. NOTE: The ERR flag always reads zero because no known error conditions are checked during this phase.	ERR = 0 SEQ_STAT = 11 SEQ_RES = 11 or 10 TO = 0	Charger type detection
Sequence Timeout	Error	The timeout interval from the time the USB device attaches to a USB port until it connects has elapsed.	ERR = 1 SEQ_STAT = last value ² SEQ_RES = last value ² TO = 1	Charger detection sequence timeout

1. See the description of the Status register for register information.

2. The SEQ_STAT and SEQ_RES fields retain the values held at the time of the timeout error.

56.3.4.1 Interrupt handling

Software reads which event caused the interrupt from the STATUS register during the interrupt service routine.

An interrupt is generated only if CONTROL[IE] is set. The CONTROL[IF] bit is always set under interrupt conditions, even if CONTROL[IE] is cleared. In this case, software can poll CONTROL[IF] to determine if an interrupt condition is pending.

Writes to CONTROL[IF] are ignored. To reset CONTROL[IF], set CONTROL[IACK] to acknowledge the interrupt. Writing to CONTROL[IACK] when CONTROL[IF] is cleared has no effect.

56.4 External signals

This section describes the module signals. The following table shows a summary of module signals that interface with the pins of the device.

Table 586. Signal descriptions

Signal	Description	I/O
USB_DM	USB D– analog data signal. The analog block interfaces directly to the D– signal on the USB bus.	I/O
USB_DP	USB D+ analog data signal. The analog block interfaces directly to the D+ signal on the USB bus.	I/O
avdd33 ¹	3.3 V regulated analog supply	I
vss_bulk	Digital/Analog ground	I
dvdd	Core voltage digital supply	I

1. Voltage must be 3.3 V $\pm$ 10% for full functionality of the module. That is, the charger detection function does not work when this voltage is below 3.0 V, and the CONTROL[START] bit should not be set.

NOTE

The transceiver module also interfaces to the USB_DM and USB_DP signals. Both modules and the USB host/hub use these signals as bidirectional, tristate signals.

Information about the signal integrity aspects of the lines including shielding, isolated return paths, input or output impedance, packaging, suggested external components, ESD, and other protections can be found in the USB 2.0 Specification and in [Application information](#).

56.5 Initialization information

This module is designed for minimal configuration while retaining significant programmability. The CLOCK register needs to be initialized to the actual system clock frequency, unless the default value already matches the system requirements.

The proper operation of the USBHSDCD instantiation of this module requires that the USBPHY_ANACTRL[DEV_PULLDOWN] bit field be reset to value 1'b0 during the period when this module is in use.

The other registers generally do not need to be modified, because they default to values that comply with the USB Battery Charging Specification. However, several timing parameters can be changed for a great deal of flexibility if a particular system requires it.

All module configuration must occur *before* initiating the charger detection sequence. Configuration changes made *after* setting CONTROL[START] result in undefined behavior.

56.6 Application information

This section provides application information.

56.6.1 External pullups

Any external pullups applied to the USB D+ or D– data lines must be capable of being disabled to prevent incorrect pullup values or incorrect operation of the USB subsystem.

56.6.2 Dead or weak battery

According to the *USB Battery Charging Specification, Revision 1.2*, a USB device with a dead or weak battery that is attached to an SDP can draw charging current from VBUS without connecting for several minutes, until the battery is charged to the point that the USB device can connect. The conditions for this operation are referred to as the Dead Battery Provision.

The *USB Battery Charging Specification, Revision 1.2* limits this charging condition to 100 mA for a duration of 45 minutes. The later *USB 2.0 Connect Timing Update ECN* modifies this charging condition to 500 mA for a duration of 2 minutes. Customers designing systems to take advantage of the Dead Battery Provision should study both specifications closely.

The USBDCD module is compatible with systems that do not check the strength of the battery. Therefore, this module assumes that the battery is good, so the USB device must immediately connect to the USB bus by pulling the USB_DP pin high after the USBDCD module has determined that the device is attached to an SDP or CDP. (By definition, a DCP does not support connection but still requires a signaling event, see [Dedicated Charging Port \(DCP\)](#).)

The module is also compatible with systems that have other circuitry to check the strength of the battery. In these systems, if it is known that the battery is weak or dead, software can delay connecting to the USB while charging using the Dead Battery Provision. Once the battery is charged to the good battery threshold, software must then connect to the USB host by pulling the USB_DP pin high.

While using the Dead Battery Provision, the USB_DP pin must signal by enabling V_{DP_SRC}. On this product that signaling can be done most simply by setting the SIGNAL_OVERRIDE[PS] bit field to value 2'b10.

56.6.3 Handling unplug events

If the device is unplugged from the USB bus during the charger detection sequence, the contents of the STATUS register must be ignored and the USBDCD module must get a Software Reset, as described in [Software reset](#).

56.7 References

The following publications are referenced in this document. For updates to these specifications, see www.usb.org.

- *USB Battery Charging Specification, Revision 1.1, USB Implementers Forum*
- *USB Battery Charging Specification (Including errata and ECNs through March 15, 2012), Revision 1.2, USB Implementers Forum*
- *Universal Serial Bus Specification Revision 2.0, USB Implementers Forum*
- *USB 2.0 Connect Timing Update ECN, USB Implementers Forum*

56.8 Glossary

The following table shows a glossary of terms used in this document.

Table 587. Glossary of terms

Term	Definition
Transceiver	Module that implements the physical layer of the USB standard (FS or LS only).
PHY	Module that implements the physical layer of the USB standard (HS capable).
Attached	Device is physically plugged into USB port, but has <i>not enabled</i> either D+ or D– pullup resistor.
Connected	Device is physically plugged into USB port, and has <i>enabled</i> either D+ or D– pullup resistor.
Suspended	After 3 ms of no bus activity, the USB device enters suspend mode.
Component	The hardware and software that make up a subsystem.
CDP	Charging Downstream Port, as defined in <i>USB Battery Charging Specification, Rev. 1.2</i>
Charging Port	A Charging Port supported by this module is either a CDP or a DCP

Table continues on the next page...

Table 587. Glossary of terms (continued)

Term	Definition
DCP	Dedicated Charging Port, as defined in <i>USB Battery Charging Specification, Rev. 1.2</i>
FS	Full speed (12 Mbit/s)
HS	High speed (480 Mbit/s)
I_{CFG_MAX}	Current limit for a USB device attached to an SDP, after configuration
I_{DEV_CHG}	Current limit for a USB device attached to a Charging Port
I_{DM_SINK}	Current sink for the D– line
I_{DP_SINK}	Current sink for the D+ line
I_{DP_SRC}	Current source for the D+ line
I_{SUSP}	Current drawn when the USB device is suspended
LDO	Low dropout
LS	Low Speed (1.5 Mbit/s)
N/A	Not applicable
OTG	On-The-Go
R_{DM_DWN}	D– pulldown resistance for data pin contact detect
SDP	Standard Downstream Port, as defined in <i>USB Battery Charging Specification, Rev 1.2</i>
V_{DAT_REF}	Data detect reference voltage for the voltage comparator
V_{DP_SRC}	Voltage source for the D+ line
V_{DM_SRC}	Voltage source for the D– line
V_{LGC}	Threshold voltage for logic high

56.9 Memory map/Register definition

This section describes the memory map and registers for the USBHSDCD module.

56.9.1 USBDCD register descriptions

56.9.1.1 USBDCD memory map

USBDCD0 base address: 4013_B800h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CONTROL)	32	RW	0001_0000h
4h	Clock (CLOCK)	32	RW	0000_00C1h
8h	Status (STATUS)	32	R	0000_0000h
Ch	Signal Override (SIGNAL_OVERRIDE)	32	RW	0000_0000h
10h	TIMER0 (TIMER0)	32	RW	0010_0000h
14h	TIMER1 (TIMER1)	32	RW	000A_0028h
18h	TIMER2_BC11 (TIMER2_BC11)	32	RW	0028_0001h
18h	TIMER2_BC12 (TIMER2_BC12)	32	RW	0001_0028h

56.9.1.2 Control (CONTROL)

Offset

Register	Offset
CONTROL	0h

Function

Controls various global functions, such as software reset, initiating the charger detection sequence, BC12 compatibility, and interrupt enablement.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						0	0	0						BC12	IE
W							SR	STAR T								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							IF	0						0	
W															IACK	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25 SR	Software Reset Determines whether a software reset is performed. 0b - Do not perform a software reset. 1b - Perform a software reset.
24 START	Start Change Detection Sequence Determines whether the charger detection sequence is initiated. 0b - Do not start the sequence. Writes of this value have no effect. 1b - Initiate the charger detection sequence. If the sequence is already running, writes of this value have no effect.
23-18 —	Reserved
17 BC12	BC12 BC1.2 compatibility. This bit cannot be changed after start detection. 0b - Compatible with BC1.1 (default) 1b - Compatible with BC1.2
16 IE	Interrupt Enable Enables/disables interrupts to the system. 0b - Disable interrupts to the system. 1b - Enable interrupts to the system.
15-9 —	Reserved
8 IF	Interrupt Flag Determines whether an interrupt is pending. 0b - No interrupt is pending. 1b - An interrupt is pending.
7-1 —	Reserved
0 IACK	Interrupt Acknowledge Determines whether the interrupt is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Do not clear the interrupt. 1b - Clear the IF bit (interrupt flag).

56.9.1.3 Clock (CLOCK)

Offset

Register	Offset
CLOCK	4h

Function

Controls the clock speed.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0				CLOCK_SPEED										0		CLOC
W																	K_U...
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	

Fields

Field	Function
31-12 —	Reserved
11-2 CLOCK_SPEED	Numerical Value of Clock Speed in Binary The unit of measure is programmed in CLOCK_UNIT. The valid range is from 1 to 1023 when the clock unit is MHz and 4 to 1023 when the clock unit is kHz. Examples with CLOCK_UNIT = 1: - For 48 MHz: 0b00_0011_0000 (48) - For 24 MHz: 0b00_0001_1000 (24) Examples with CLOCK_UNIT = 0:

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- For 100 kHz: 0b00_0110_0100 (100) - For 500 kHz: 0b01_1111_0100 (500)
1 —	Reserved
0 CLOCK_UNIT	Unit of Measurement Encoding for Clock Speed Specifies the unit of measure for the clock speed. 0b - kHz Speed (between 4 kHz and 1023 kHz) 1b - MHz Speed (between 1 MHz and 1023 MHz)

56.9.1.4 Status (STATUS)

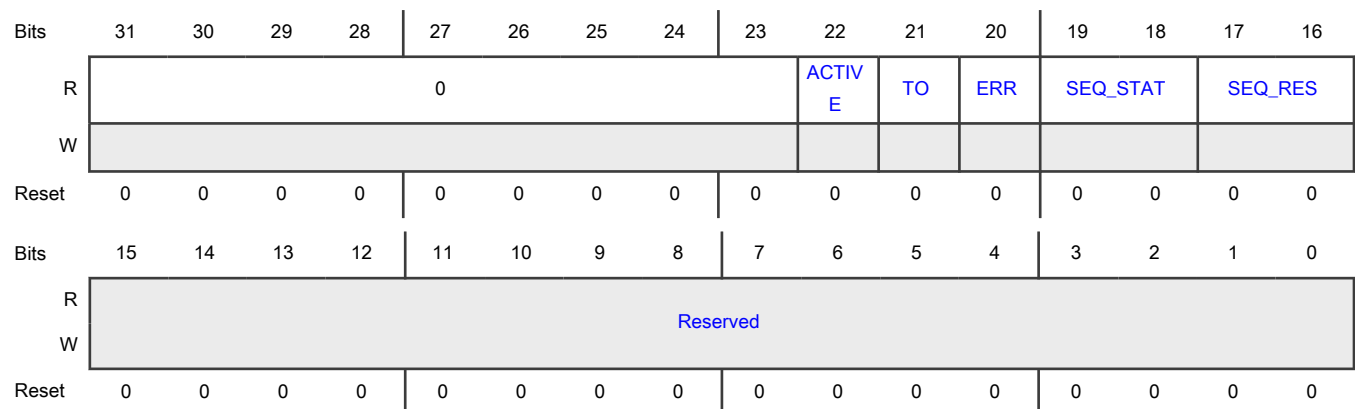
Offset

Register	Offset
STATUS	8h

Function

Stores the current state of the module for system software monitoring.

Diagram



Fields

Field	Function
31-23	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
22 ACTIVE	Active Status Indicator Indicates whether the sequence is running. 0b - The sequence is not running. 1b - The sequence is running.
21 TO	Timeout Flag Indicates whether the detection sequence is passed the timeout threshold. 0b - The detection sequence is not running for over 1 s. 1b - It is over 1 s since the data pin contact was detected and debounced.
20 ERR	Error Flag Indicates whether there is an error in the detection sequence. 0b - No sequence errors. 1b - Error in the detection sequence. See the SEQ_STAT field to determine the phase in which the error occurred.
19-18 SEQ_STAT	Charger Detection Sequence Status Indicates the status of the charger detection sequence. 00b - The module is either not enabled, or the module is enabled but the data pins have not yet been detected. 01b - Data pin contact detection is complete. 10b - Charging port detection is complete. 11b - Charger type detection is complete.
17-16 SEQ_RES	Charger Detection Sequence Results Reports how the charger detection is attached. 00b - No results to report. 01b - Attached to an SDP. Must comply with USB 2.0 by drawing only 2.5 mA (max) until connected. 10b - Attached to a charging port. The exact meaning depends on bit 18 (value 0: Attached to either a CDP or a DCP. The charger type detection has not completed. value 1: Attached to a CDP. The charger type detection has completed.) 11b - Attached to a DCP.
15-0 —	Reserved NOTE Bits do not always read as 0.

56.9.1.5 Signal Override (SIGNAL_OVERRIDE)

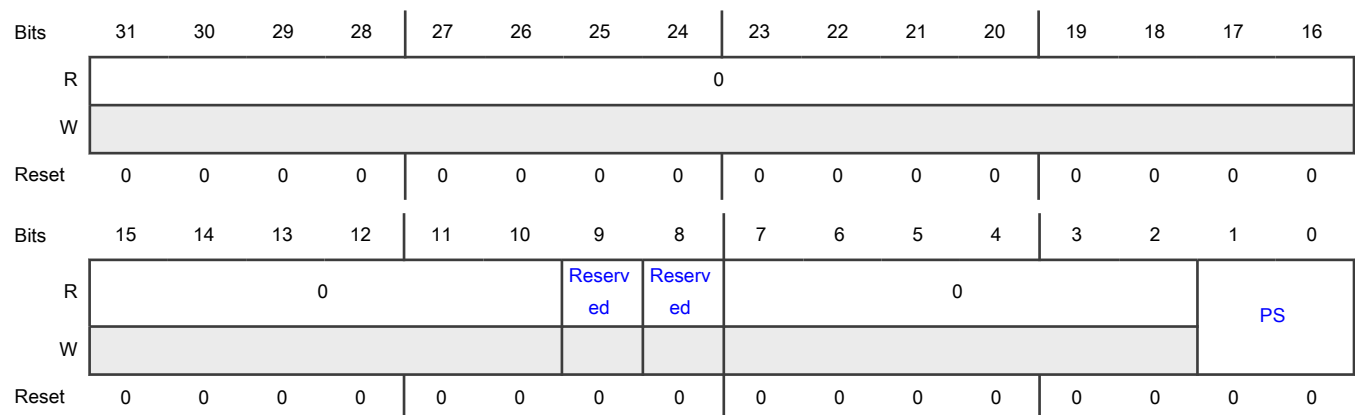
Offset

Register	Offset
SIGNAL_OVERRIDE	Ch

Function

Provides a way for the customer to enable signaling required by the USB BC Rev. 1.2 Specification after the battery charger detection sequences is completed.

Diagram



Fields

Field	Function
31-10 —	Reserved
9 —	Reserved Reserved, not for customer use. The value of this bit field may change during module operation.
8 —	Reserved Reserved, not for customer use. The value of this bit field may change during module operation.
7-2 —	Reserved
1-0 PS	Phase Selection Enables specified voltage and current source circuits on the USB_DP and USB_DM pins. Use of the override value below enables the charger detection bias circuits without needing to set the CONTROL[START] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Customers may set this bit field to 2'b10 for required signaling if attached to a Dedicated Charging Port, or during operation under the Dead Battery Provision. 00b - No overrides. Bit field must remain at this value during normal USB data communication to prevent unexpected conditions on USB_DP and USB_DM pins. (Default) 01b - Reserved, not for customer use. 10b - Enables VDP_SRC voltage source for the USB_DP pin and IDM_SINK current source for the USB_DM pin. 11b - Reserved, not for customer use.

56.9.1.6 TIMER0 (TIMER0)

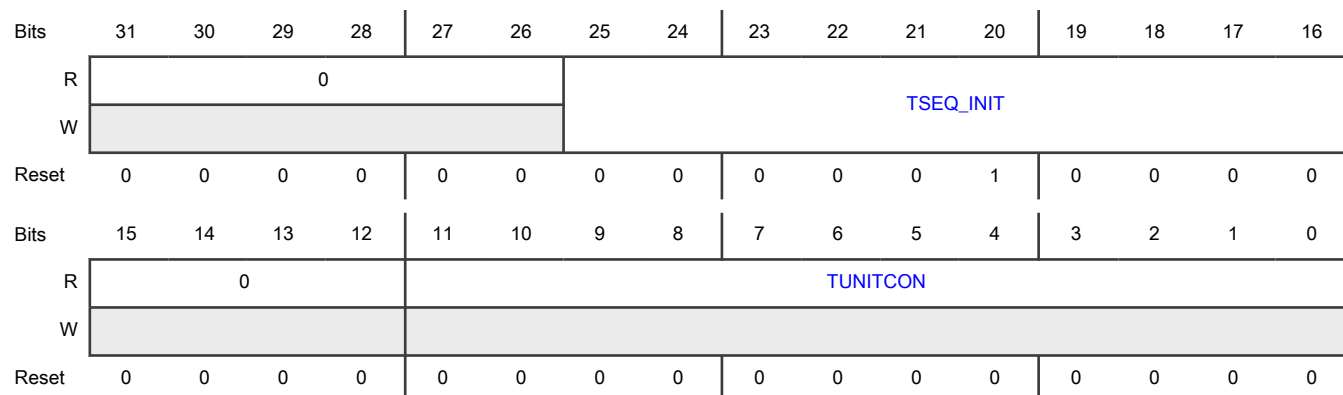
Offset

Register	Offset
TIMER0	10h

Function

This register has a TSEQ_INIT field that represents the system latency in ms. Latency is measured from the time when VBUS goes active until the time system software initiates the charger detection sequence in the USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, however the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1 s or less.

Diagram



Fields

Field	Function
31-26	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
25-16 TSEQ_INIT	Sequence Initiation Time TSEQ_INIT represents the system latency (in ms) measured from the time VBUS goes active to the time system software initiates the charger detection sequence in the USBDCD module. When software sets the CONTROL[START] bit, the Unit Connection Timer (TUNITCON) is initialized with the value of TSEQ_INIT. Valid values are 0-1023, but the USB Battery Charging Specification requires the entire sequence, including TSEQ_INIT, to be completed in 1 s or less. 00_0000_0000b-11_1111_1111b - 0ms - 1023ms
15-12 —	Reserved
11-0 TUNITCON	Unit Connection Timer Elapse (in ms) Displays the amount of elapsed time since the event of setting the CONTROL[START] bit plus the value of TSEQ_INIT. The timer is automatically initialized with the value of TSEQ_INIT before starting to count. This timer enables compliance with the maximum time allowed to connect T_{UNIT_CON} under the USB Battery Charging Specification. If the timer reaches the one second limit, the module triggers an interrupt and sets the error flag STATUS[ERR]. The timer continues counting throughout the charger detection sequence, even when control has been passed to software. As long as the module is active, the timer continues to count until it reaches the maximum value of 0xFFF (4095 ms). The timer does not rollover to zero. A software reset clears the timer.

56.9.1.7 TIMER1 (TIMER1)

Offset

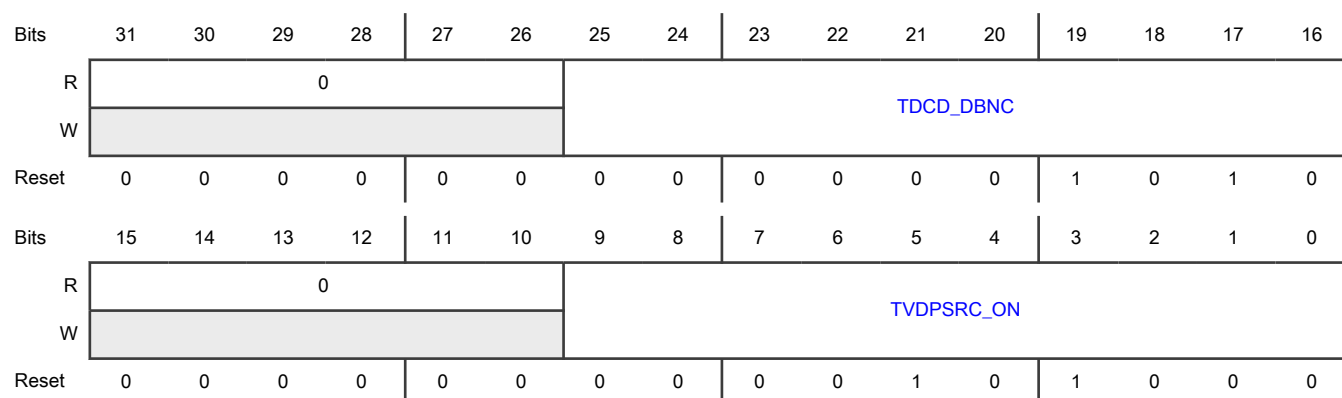
Register	Offset
TIMER1	14h

Function

Contains timing parameters.

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram**Fields**

Field	Function
31-26 —	Reserved
25-16 TDCD_DBNC	Time Period to Debounce D+ Signal Sets the time period (ms) to debounce the D+ signal during the data pin contact detection phase. See "Debouncing the data pin contact" Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 10 ms. 00_0000_0001b-11_1111_1111b - 1ms - 1023ms
15-10 —	Reserved
9-0 TVDPsrc_ON	Time Period Comparator Enabled This timing parameter is used after detection of the data pin. See "Charging Port Detection". Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. 00_0000_0001b-11_1111_1111b - 1ms - 1023ms

56.9.1.8 TIMER2_BC11 (TIMER2_BC11)**Offset**

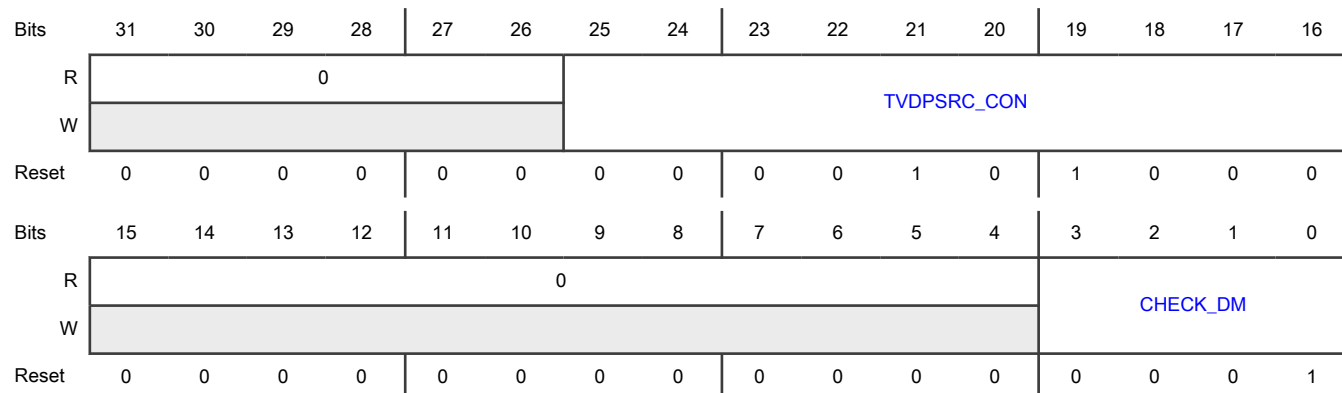
Register	Offset
TIMER2_BC11	18h

Function

TIMER2_BC11 contains timing parameters for *USB Battery Charging Specification, Rev 1.1*.

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram**Fields**

Field	Function
31-26 —	Reserved
25-16 TVDPsrc_CON	Time Period Before Enabling D+ Pullup Sets the time period (ms) that the module waits after charging port detection before system software must enable the D+ pullup to connect to the USB host. Valid values are 1-1023, but the USB Battery Charging Specification requires a minimum value of 40 ms. 00_0000_0001b-11_1111_1111b - 1ms - 1023ms
15-4 —	Reserved
3-0 CHECK_DM	Time Before Check of D- Line Sets the amount of time (in ms) that the module waits after the device connects to the USB bus until checking the state of the D- line to determine the type of charging port. See "Charger Type Detection." Valid values are 1-15 ms. 0001b-1111b - 1ms - 15ms

56.9.1.9 TIMER2_BC12 (TIMER2_BC12)**Offset**

Register	Offset
TIMER2_BC12	18h

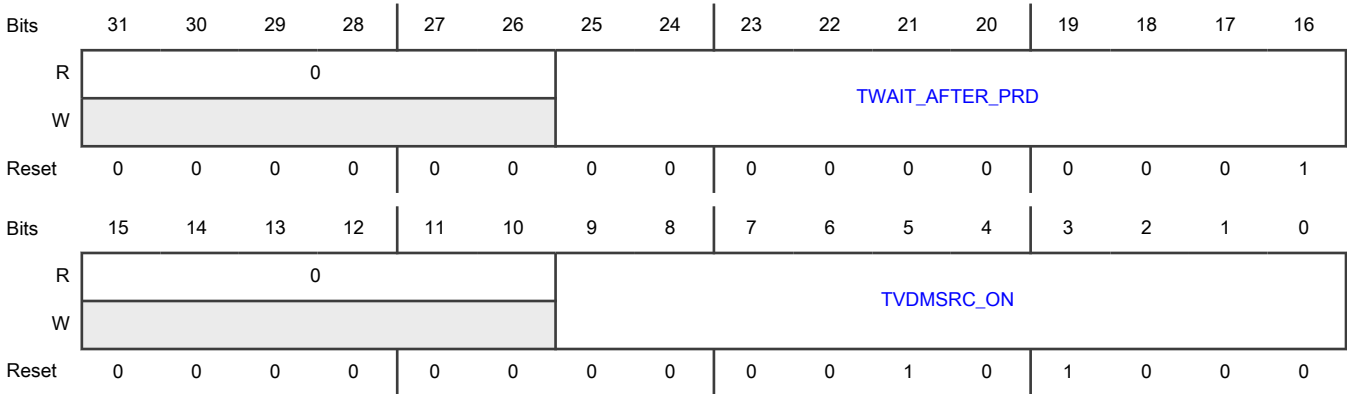
Function

Contains timing parameters for *USB Battery Charging Specification, Rev 1.2*.

NOTE

Register values can be written that are not compliant with the USB Battery Charging Specification, so care should be taken when overwriting the default values.

Diagram



Fields

Field	Function
31-26 —	Reserved
25-16 TWAIT_AFTER_PRD	TWAIT_AFTER_PRD Sets the amount of time (in ms) that the module waits after primary detection before start to secondary detection. Valid values are 1-1023ms. 00_0000_0001b-11_1111_1111b - 1ms - 1023ms
15-10 —	Reserved
9-0 TVDMSRC_ON	TVDMSRC_ON Sets the amount of time (in ms) that the module enables the V _{DM_SRC} . Valid values are 0-40ms. 00_0000_0000b-00_0010_1000b - 0ms - 40ms

Chapter 57

Universal Serial Bus 2.0 Integrated PHY (USBPHY)

57.1 Chip-specific USB2.0 PHY information

Table 588. Reference links to related information

Topic	Related module	Reference
Full description	USBPHY	USBPHY
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

NOTE

This chip does not support low speed mode.

57.1.1 Initialization

Initial configuration of a PHY is accomplished as follows:

- In the PDRUNCFG0 register (see SYSCCTL), clear the SYSXTAL_PD to enable the main oscillator. Also clear the LPOSC_PD bit if the low power oscillator will be used to source 32k_wake_clk.
- In the PSCCTL0 register (see CLKCTL) set the USBHS_PHY_CLK bit to enable the PHY bus clock. Also program PFC1DIV to make sure the PHY bus clock is not higher than 120 MHz (see CLKCTL).
- Configure SYSOSCCTL0, SYSOSC_BYPASS, USBHSFCLKSEL, and USBHSFCLKDIV in order to provide a reference clock to the PHY (see CLKCTL).
- With the WAKECLK32KHZSEL register (see CLKCTL), select a wake clock.

Enable some settings via the PLL Control/Status Register in the USB PHY (See USBPHY).

- PLL_SIC[PLL_BYPASS]: Clear this bit to clear bypass feature
- PLL_SIC[PLL_REG_ENABLE]: Enable USB PLL regulator
- Delay more than 15 us.
- PLL_SIC[PLL_BYPASS]: Clear this bit to clear bypass
- PLL_SIC[PLL_POWER]: Power-up the USB PLL
- PLL_SIC[PLL_EN_USB_CLKS]: Enable the USB clock
- PLL_SIC[PLL_ENABLE]: Enable the clock output from the USB PLL

The system oscillator ranges from 16 MHz to 32 MHz. PLL_SIC[PLL_DIV_SEL[2:0]] (See USBPHY) should be programmed properly as per the external reference selected.

57.2 Overview

The integrated USB 2.0 PHY is capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, the full-speed (FS) rate of 12 Mbits/s, or the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 High-Speed PHY.

57.2.1 Block diagram

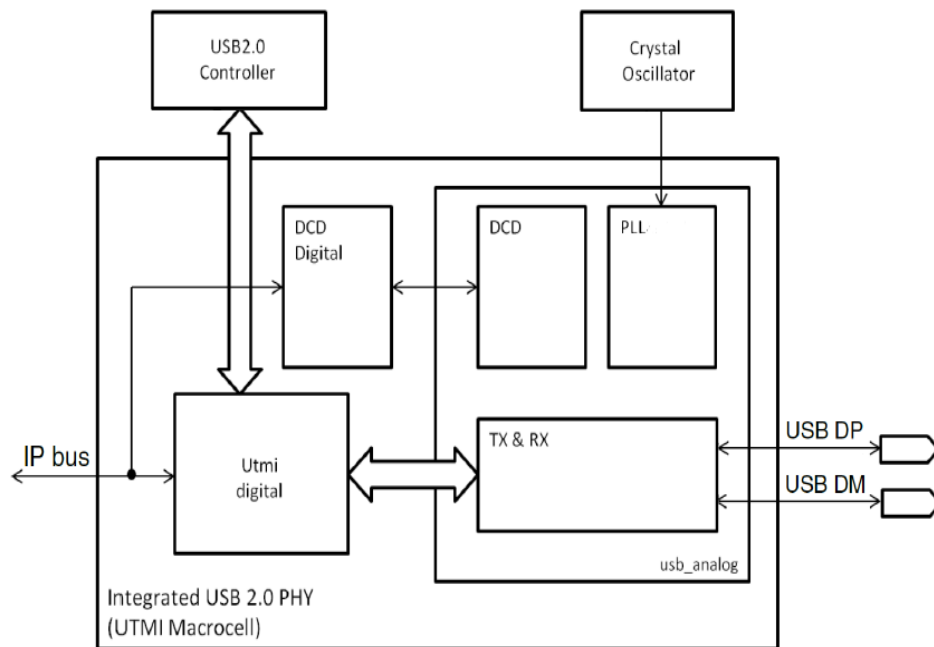


Figure 292. USB2.0 PHY block diagram

57.2.2 Features

The USB2.0 PHY includes the following features:

- Supports USB low-speed (LS) rate of 1.5 Mbits/s, the full-speed (FS) rate of 12 Mbits/s, or the USB 2.0 high-speed (HS) rate of 480 Mbits/s.
- Standard UTMI+ interface
- Integrated 480 MHz PLL and DCD (Device Charger Detector) analog circuits, with an IP bus interface for configurations.
- USB_DP and USB_DM pins connecting directly to a USB connector.

57.3 Functional description

The following sections describe functional details of the USB PHY module.

57.3.1 USB PHY Blocks

The UTMI provides a 16-bit interface to the USB controller and is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter.

57.3.1.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection.

The USB PHY PLL supplies a 480 MHz clock to UTMI digital. The UTMI block divides the clock to generate the 30 MHz clock used in the interface.

57.3.1.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

57.3.1.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and 480 MHz, 9X oversampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

57.3.1.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module, as shown in the figure below.

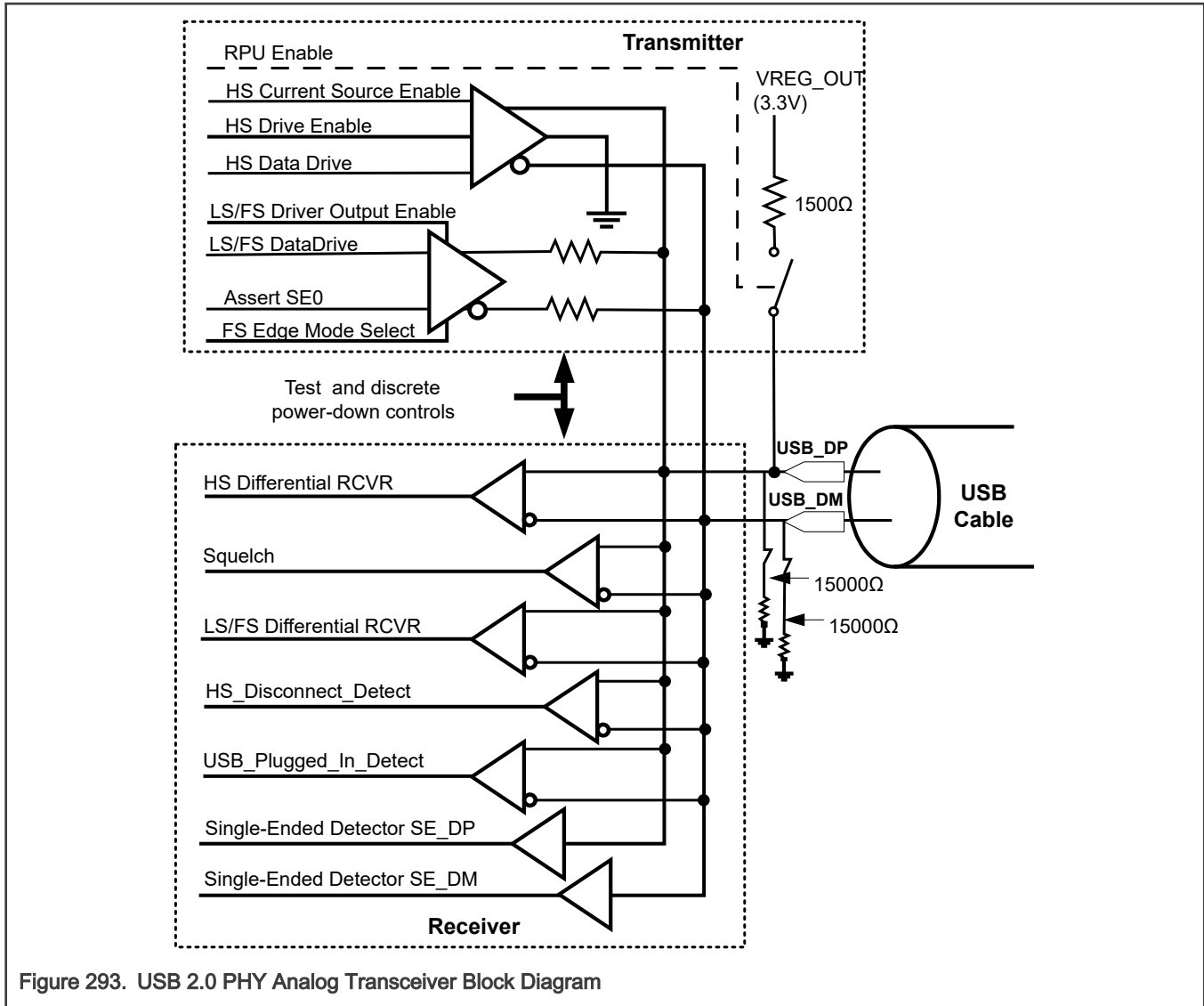


Figure 293. USB 2.0 PHY Analog Transceiver Block Diagram

57.3.1.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is high (1'b1) if the differential signal is greater than a 0-V threshold.

Otherwise, its output is low (1'b0). Its purpose is to discriminate the ± 400 -mV differential voltage resulting from the high-speed drivers current flow into the dual 45 Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

57.3.1.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is high (1'b1) if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is low (1'b0).

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

57.3.1.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_DP line is above the crossover point and the USB_DM line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

57.3.1.4.4 HS Disconnect Detector

The HS Disconnect Detector is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

57.3.1.4.5 USB Plugged-In (Cable Attach) Detection

The USB HS PHY when operating in device mode provides three different methods for the local USB device to detect the attachment of a cable also attached to a remote USB host or hub. Only one of these methods should be enabled at a time when waiting to identify the cable attachment event, and all of them must be disabled during USB data communication.

The preferred standards-based method is to use the Data Contact Detect function per the USB Battery Charging Specification, Revision 1.2. The use of this method is described in the USB Device Charger Detection Module (USBDCD) chapter in this Reference Manual.

The second method, called the USB plugged-in detector, looks for both USB_DP and USB_DM to be high. There is a pair of large on-chip switchable pullup resistors (200 K Ω) that hold both USB_DP and USB_DM high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case. When operating in device mode, the upstream port in the remote host/hub interface contains a 15 K Ω pulldown resistor which could easily override the 200 K Ω pullup resistor. When the cable attachment event occurs, at least one signal in the pair will be low, which will force the plugged-in detector output high. The USB plugged-in detector is controlled through the USBPHY_CTRL[ENVPLUGINDET] bit field and results are observable through the USBPHY_STATUS[DEVPLUGIN_STATUS] bit field.

The final method is a legacy resistive detection function described in the now-obsolete USB Battery Charging Specification, Revision 1.0. This method enables a 125 K Ω pullup resistor on the USB_DP pin and a 375 K Ω pulldown resistor on the USB_DM pin. When no cable is plugged in, the USB_DP pin will be high and the USB_DM pin will be low. If the local device has a cable attached to a remote host or hub, due to the host 15 K Ω pulldowns both the USB_DP and USB_DM pins will be low. If the local device has a cable attached to a dedicated charger with no pullup/pulldown resistors, both the USB_DP and USB_DM pins will be high. This method is enabled through the USBPHY_USB1_VBUS_DETECT[EN_CHARGER_RESISTOR] bit field. Results can be observed using the single ended receiver status in the USBPHY_CHRG_DET_STAT[DP_STATE, DM_STATE] bit fields.

57.3.1.4.6 Single-Ended USB_DP Receiver

The single-ended USB_DP receiver output is high whenever the USB_DP input is above its nominal 1.8 V threshold.

57.3.1.4.7 Single-Ended USB_DM Receiver

The single-ended USB_DM receiver output is high whenever the USB_DM input is above its nominal 1.8 V threshold.

57.3.1.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

57.3.1.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 K Ω pullup resistor.

57.3.1.5.1 Switchable High-Speed 45 Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with $45\ \Omega$ at each end, each driver sees a $22.5\ \Omega$ load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device. The **TXCAL45DP** bit, allows one of 16 trimming resistor values to be placed in parallel with the $45\ \Omega$ terminator on the USB_DP signal.

57.3.1.5.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially a pair of low-impedance pullup/pulldown devices that are switched in a differential mode for most low-speed or full-speed signaling. One output is driven high while the other output is driven low to signal the "J" state or the "K" state. Both outputs are driven low for the low-speed/fullspeed "SE0" state.

57.3.1.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (I_{ref}) and essentially steers it down either the USB_DP signal or the USB_DM signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the $22.5\ \Omega$ termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (V_{bg}) circuit.

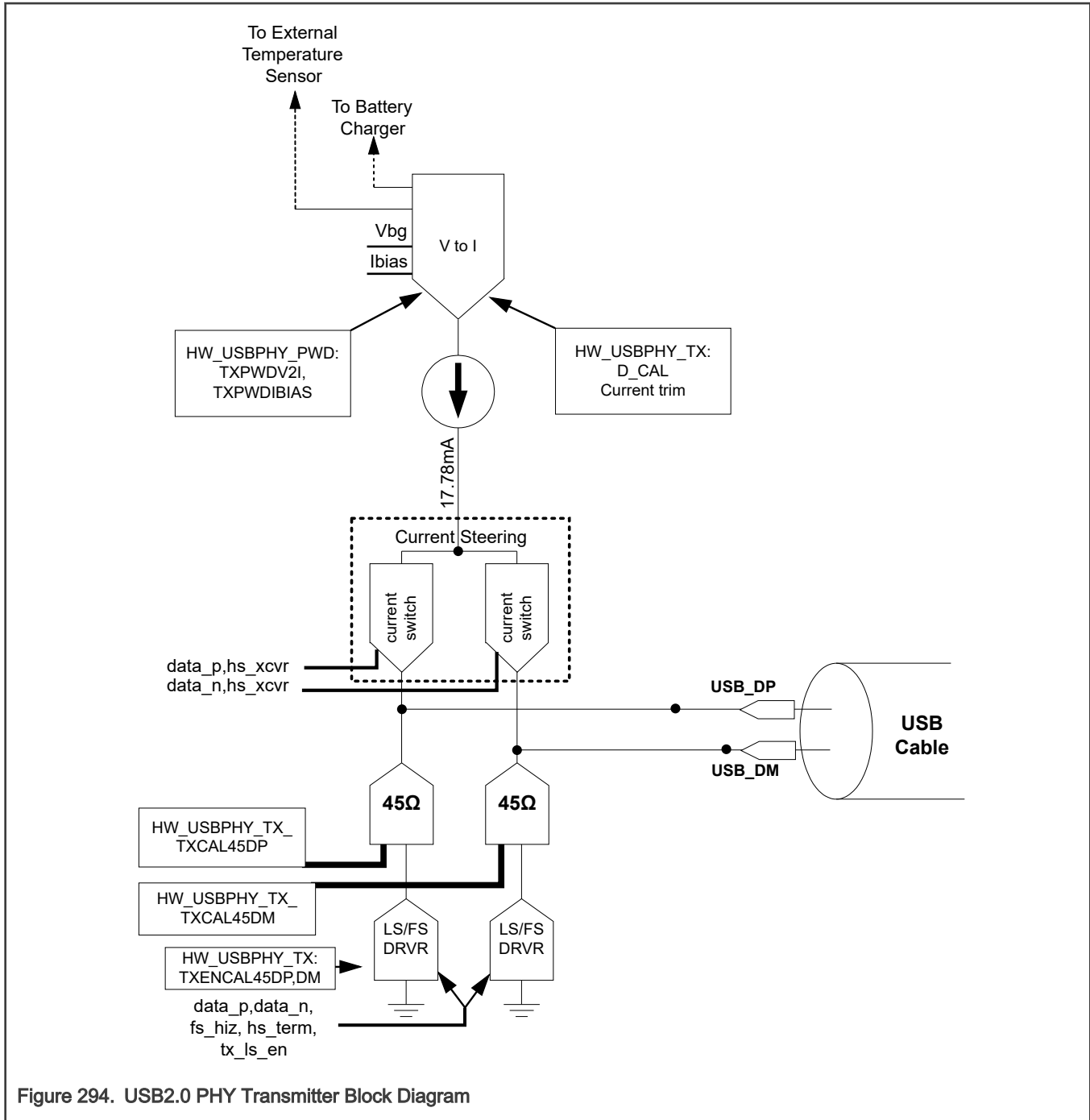
57.3.1.5.4 Switchable 1.5 K Ω USB_DP Pullup Resistor

The USB_DP signal has a switchable 1.5 K Ω pullup resistor.

The resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. The resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

57.3.1.5.5 Switchable 15 K Ω USB_DP and USB_DM Pulldown Resistors

The 15 K Ω pulldown resistors on USB_DP and USB_DM signals are enabled in host mode to indicate to the device controller on the far end of the USB connection that a host is present. These resistors are also used during certain Battery Charging detector functions.



57.3.1.6 Recommended Register Configuration for USB Certification

The USB2.0 PHY can be programmed to adjust several transceiver parameters.

- The TX HS driver termination impedance/FS driver output impedance and HS driver output currents can be adjusted through bit fields of the TX register. The ability to override the default parametric trim bit fields using the TX register has to be enabled by setting bit fields in the TRIM_OVERRIDE register.
- The HS RX thresholds for the Envelope Detector/Squelch comparator and the Host Disconnect comparator can be adjusted through bit fields of the RX register.

The default values of the bit fields for the transceiver parametric trims are centered with no changes needed for USB Certification testing when the product is used with boards optimized for signal integrity on the HS USB port.

In other cases, such as when external components are inserted between the USB DP/DM pins and the USB connector or other compromises are made on USB DP/DM signal routing, changes to the parametric trim bit fields may be useful.

57.4 Memory Map and Register Definition

The USB PHY registers are located in the AHB clock domain. They can be accessed directly by the processor.

57.4.1 USB2.0 HS PHY register descriptions

57.4.1.1 USBHSPHY memory map

USBPHY0 base address: 4013_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Power Down (PWD)	32	RW	001E_1C00h
4h	Power Down Register Set (PWD_SET)	32	RW	001E_1C00h
8h	Power Down Register Clear (PWD_CLR)	32	RW	001E_1C00h
Ch	Power Down Register Toggle (PWD_TOG)	32	RW	001E_1C00h
10h	TX Control (TX)	32	RW	0007_0707h
14h	TX Control Set (TX_SET)	32	RW	0007_0707h
18h	TX Control Clear (TX_CLR)	32	RW	0007_0707h
1Ch	TX Control Toggle (TX_TOG)	32	RW	0007_0707h
20h	RX Control (RX)	32	RW	0000_0000h
24h	RX Control Set (RX_SET)	32	RW	0000_0000h
28h	RX Control Clear (RX_CLR)	32	RW	0000_0000h
2Ch	RX Control Toggle (RX_TOG)	32	RW	0000_0000h
30h	General Purpose Control (CTRL)	32	RW	C000_0000h
34h	General Purpose Control Set (CTRL_SET)	32	RW	C000_0000h
38h	General Purpose Control Clear (CTRL_CLR)	32	RW	C000_0000h
3Ch	General Purpose Control Toggle (CTRL_TOG)	32	RW	C000_0000h
40h	Status (STATUS)	32	R	0000_0000h
50h	Debug 0 (DEBUG0)	32	RW	7F18_0000h
54h	Debug 0 Set (DEBUG0_SET)	32	RW	7F18_0000h
58h	Debug Clear (DEBUG0_CLR)	32	RW	7F18_0000h
5Ch	Debug Toggle (DEBUG0_TOG)	32	RW	7F18_0000h
70h	UTMI Debug 1 (DEBUG1)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
74h	UTMI Debug 1 Set (DEBUG1_SET)	32	RW	0000_0000h
78h	UTMI Debug 1 Clear (DEBUG1_CLR)	32	RW	0000_0000h
7Ch	UTMI Debug 1 Toggle (DEBUG1_TOG)	32	RW	0000_0000h
80h	Version (VERSION)	32	R	0500_0000h
A0h	PLL Control/Status (PLL_SIC)	32	RW	00D1_2000h
A4h	PLL Control/Status Set (PLL_SIC_SET)	32	RW	00D1_2000h
A8h	PLL Control/Status Clear (PLL_SIC_CLR)	32	RW	00D1_2000h
ACH	PLL Control/Status Toggle (PLL_SIC_TOG)	32	RW	00D1_2000h
C0h	VBUS detect (USB1_VBUS_DETECT)	32	RW	0070_0004h
C4h	VBUS detect Set (USB1_VBUS_DETECT_SET)	32	RW	0070_0004h
C8h	VBUS detect Clear (USB1_VBUS_DETECT_CLR)	32	RW	0070_0004h
CCh	VBUS detect Toggle (USB1_VBUS_DETECT_TOG)	32	RW	0070_0004h
D0h	VBUS Detect Status (USB1_VBUS_DET_STAT)	32	R	0000_0000h
E0h	Charger Detect Control (USB1_CHRG_DETECT)	32	RW	0000_0000h
E4h	Charger Detect Control Set (USB1_CHRG_DETECT_SET)	32	RW	0000_0000h
E8h	Charger Detect Control Clear (USB1_CHRG_DETECT_CLR)	32	RW	0000_0000h
ECh	Charger Detect Control Toggle (USB1_CHRG_DETECT_TOG)	32	RW	0000_0000h
F0h	Charge Detect Status (USB1_CHRG_DET_STAT)	32	R	0000_0000h
100h	Analog Control (ANACTRL)	32	RW	0000_0400h
104h	Analog Control Set (ANACTRL_SET)	32	RW	0000_0400h
108h	Analog Control Clear (ANACTRL_CLR)	32	RW	0000_0400h
10Ch	Analog Control Toggle (ANACTRL_TOG)	32	RW	0000_0400h
110h	USB PHY Loopback Control/Status (USB1_LOOPBACK)	32	RW	0055_0000h
114h	USB PHY Loopback Control/Status Set (USB1_LOOPBACK_SET)	32	RW	0055_0000h
118h	USB PHY Loopback Control/Status Clear (USB1_LOOPBACK_CLR)	32	RW	0055_0000h
11Ch	USB PHY Loopback Control/Status Toggle (USB1_LOOPBACK_TOG)	32	RW	0055_0000h
120h	Loopback Packet Number Select (USB1_LOOPBACK_HSFSCNT)	32	RW	0004_0010h
124h	USB PHY Loopback Packet Number Select Set (USB1_LOOPBACK_HSFSCNT_SET)	32	RW	0004_0010h
128h	USB PHY Loopback Packet Number Select Clear (USB1_LOOPBACK_HSFSCNT_CLR)	32	RW	0004_0010h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
12Ch	USB PHY Loopback Packet Number Select Toggle (USB1_LOOPBACK_HSFSCNT_TOG)	32	RW	0004_0010h
130h	Trim Override Enable (TRIM_OVERRIDE_EN)	32	RW	0000_007Fh
134h	Trim Set (TRIM_OVERRIDE_EN_SET)	32	RW	0000_007Fh
138h	Trim Clear (TRIM_OVERRIDE_EN_CLR)	32	RW	0000_007Fh
13Ch	Trim Toggle (TRIM_OVERRIDE_EN_TOG)	32	RW	0000_007Fh

57.4.1.2 Power Down (PWD)

Offset

Register	Offset
PWD	0h

Function

The USB PHY Power-Down Register provides overall control of the PHY power state.

Before programming this register, the PHY clocks must be enabled in the CTRL and PLL_SIC registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved												RXPW	RXPW	RXPW	RXPW	Reserved
W													DRX	DDI...	D1P...	DENV	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserved				TXPW	TXPW	TXPW	Reserved									
W					DV2I	DIB...	DFS										
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31-21	Reserved
—	

Table continues on the next page...

Table continued from the previous page...

Field	Function
20 RXPWDRX	Power down USB PHY receiver except the FS differential. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] is enabled. 0b - Normal operation 1b - Power-down the entire USB PHY receiver block except for the full-speed differential receiver
19 RXPWDDIFF	Power down USB HS differential receiver. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] is enabled. 0b - Normal operation 1b - Power down the USB high-speed differential receiver
18 RXPWD1PT1	Power down USB FS differential receiver. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] is enabled. 0b - Normal operation 1b - Power down the USB full-speed differential receiver
17 RXPWDENV	Power down USB HS receiver envelope detector. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] is enabled. 0b - Normal operation 1b - Power down the USB high-speed receiver envelope detector (squellch signal)
16-13 —	Reserved
12 TXPWDV2I	Power down USB PHY V-I converter and current mirror. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] is enabled. These circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down. 0b - Normal operation 1b - Power-down the USB PHY transmit V-to-I converter and the current mirror
11 TXPWDIBIAS	Power down USB PHY current bias block. This bit will be auto cleared if there is USB wakeup event while CTRL[ENAUTOCLR_PHY_PWD] bit of is enabled. The current bias block is shared with the battery charge circuit. Setting this bit to 1 does not powerdown the circuit unless the corresponding bit in the battery charge control is also set for power-down. 0b - Normal operation 1b - Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path.
10	Power down USB FS drivers.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXPWDFS	This bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of CTRL is enabled. 0b - Normal operation 1b - Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output
9-0 —	Reserved

57.4.1.3 Power Down Register Set (PWD_SET)

Offset

Register	Offset
PWD_SET	4h

Function

Writing a 1 to a bit position in this register sets the corresponding bit in PWD register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved												RXPW	RXPW	RXPW	RXPW	Reserved
W													DRX	DDI...	D1P...	DENV	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				TXPW	TXPW	TXPW	Reserved								
W					DV2I	DIB...	DFS									
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20 RXPWDRX	Power down USB PHY receiver except the FS differential. 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - Sets the corresponding PWD bit
19 RXPWDDIFF	Power down USB HS differential receiver. 0b - No effect 1b - Sets the corresponding PWD bit
18 RXPWD1PT1	Power down USB FS differential receiver. 0b - No effect 1b - Sets the corresponding PWD bit
17 RXPWDENV	Power down USB HS receiver envelope detector. 0b - No effect 1b - Sets the corresponding PWD bit
16-13 —	Reserved
12 TXPWDV2I	Power down USB PHY V-I converter and current mirror. 0b - No effect 1b - Sets the corresponding PWD bit
11 TXPWDIBIAS	Power down USB PHY current bias block. 0b - No effect 1b - Sets the corresponding PWD bit
10 TXPWDFS	Power down USB FS drivers. 0b - No effect 1b - Sets the corresponding PWD bit
9-0 —	Reserved

57.4.1.4 Power Down Register Clear (PWD_CLR)

Offset

Register	Offset
PWD_CLR	8h

Function

Writing a 1 to a bit position in this register clears the corresponding bit in PWD.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												RXPW	RXPW	RXPW	RXPW
W													DRX	DDI...	D1P...	DENV
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				TXPW	TXPW	TXPW	Reserved								
W					DV2I	DIB...	DFS									
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-21 —	Reserved
20 RXPWDRX	Power down USB PHY receiver except the FS differential. 0b - No effect 1b - Clears the corresponding PWD bit
19 RXPWDDIFF	Power down USB HS differential receiver. 0b - No effect 1b - Clears the corresponding PWD bit
18 RXPWD1PT1	Power down USB FS differential receiver. 0b - No effect 1b - Clears the corresponding PWD bit
17 RXPWDENV	Power down USB HS receiver envelope detector. 0b - No effect 1b - Clears the corresponding PWD bit
16-13 —	Reserved
12 TXPWDV2I	Power down USB PHY V-I converter and current mirror. 0b - No effect 1b - Clears the corresponding PWD bit
11 TXPWDIBIAS	Power down USB PHY current bias block. 0b - No effect 1b - Clears the corresponding PWD bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 TXPWDFS	Power down USB FS drivers. 0b - No effect 1b - Clears the corresponding PWD bit
9-0 —	Reserved

57.4.1.5 Power Down Register Toggle (PWD_TOG)

Offset

Register	Offset
PWD_TOG	Ch

Function

Writing a 1 to a bit position in this register toggles (inverts) the corresponding position in PWD register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved												RXPW	RXPW	RXPW	RXPW	Reserv ed
W													DRX	DDI...	D1P...	DENV	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	Reserved				TXPW	TXPW	Reserved										
W					DV2I	DIB...											DFS
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31-21 —	Reserved
20 RXPWDRX	Power down USB PHY receiver except the FS differential. 0b - No effect 1b - Toggles the corresponding PWD bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 RXPWDDIFF	Power down USB HS differential receiver. 0b - No effect 1b - Toggles the corresponding PWD bit
18 RXPWD1PT1	Power down USB FS differential receiver. 0b - No effect 1b - Toggles the corresponding PWD bit
17 RXPWDENV	Power down USB HS receiver envelope detector. 0b - No effect 1b - Toggles the corresponding PWD bit
16-13 —	Reserved
12 TXPWDV2I	Power down USB PHY V-I converter and current mirror. 0b - No effect 1b - Toggles the corresponding PWD bit
11 TXPWDIBIAS	Power down USB PHY current bias block. 0b - No effect 1b - Toggles the corresponding PWD bit
10 TXPWDFS	Power down USB FS drivers.
9-0 —	Reserved

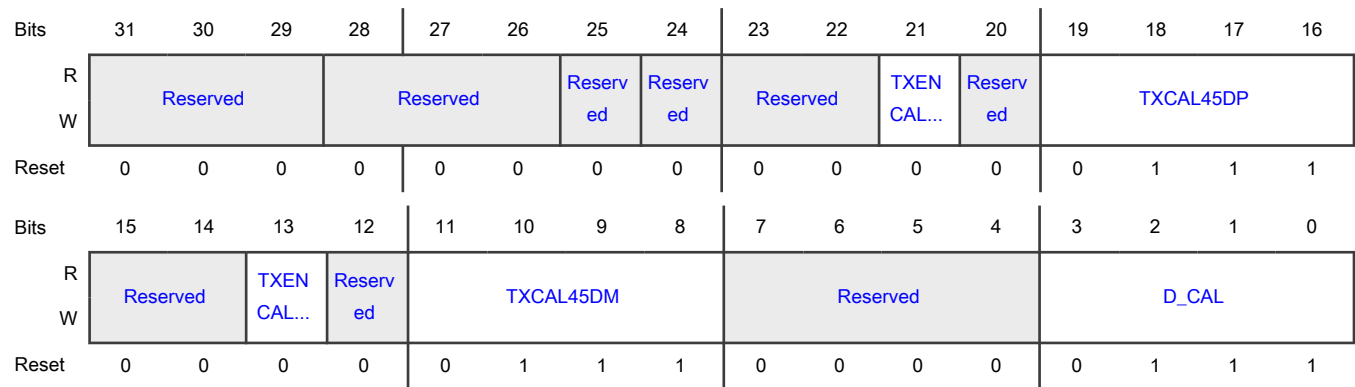
57.4.1.6 TX Control (TX)

Offset

Register	Offset
TX	10h

Function

The USB PHY Transmitter Control Register handles the transmit controls.

Diagram**Fields**

Field	Function
31-29 —	Reserved
28-26 —	Reserved
25 —	Reserved This bit field must remain at value 1'b0.
24 —	Reserved This bit field must remain at value 1'b0.
23-22 —	Reserved
21 TXENCAL45DP	DP series resistance calibration Enable resistance calibration on DN.
20 —	Reserved
19-16 TXCAL45DP	DP series termination resistance trim. Decode to trim the nominal 45Ω series termination resistance to the USB_DM output pin. Maximum resistance = 0000. Resistance is centered by design at 1000. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.
15-14 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 TXENCAL45DN	DN series Resistance calibration. Enable resistance calibration on DN.
12 —	Reserved
11-8 TXCAL45DM	DM series termination resistance trim. Decode to trim the nominal 45Ω series termination resistance to the USB_DM output pin. Maximum resistance = 0000. Resistance is centered by design at 1000. Trimming this resistance will impact both the overshoot/undershoot of the Full Speed TX output and the amplitude of the High Speed TX output.
7-4 —	Reserved
3-0 D_CAL	Current Trim decode. Decode to trim the nominal 17.78mA current source for the High Speed TX drivers on USB_DP and USB_DM. This current is directly proportional to the amplitude of the High Speed TX eye diagram. • 0000 - Maximum current, approximately 19% above nominal. • 0111 - Nominal • 1111 - Minimum current, approximately 19% below nominal.

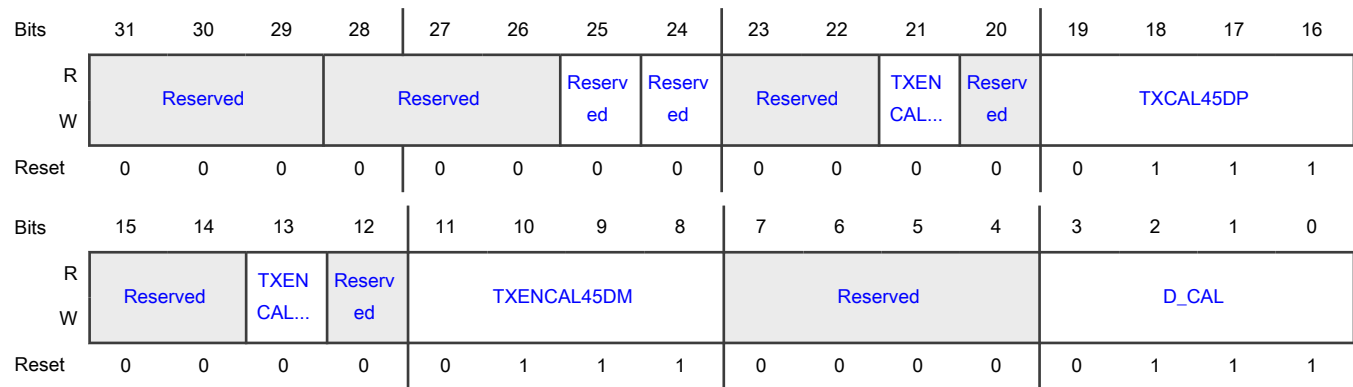
57.4.1.7 TX Control Set (TX_SET)

Offset

Register	Offset
TX_SET	14h

Function

Writing a 1 to a bit in this register sets the corresponding bit in TX register

Diagram**Fields**

Field	Function
31-29 —	Reserved
28-26 —	Reserved
25 —	Reserved
24 —	This bit field must remain at value 1'b0.
23-22 —	Reserved
21 TXENCAL45DP	Enable resistance calibration on DP 0b - No effect 1b - Sets the corresponding TX bit
20 —	Reserved
19-16 TXCAL45DP	DP series termination resistance trim. 0000b - No effect 0001b - Sets the corresponding TX bit
15-14 —	Reserved
13	Enable resistance calibration on DN.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TXENCAL45DN	0b - No effect 1b - Sets the corresponding TX bit
12 —	Reserved
11-8 TXENCAL45DM	DM series termination resistance trim. 0000b - No effect 0001b - Sets the corresponding TX bit
7-4 —	Reserved
3-0 D_CAL	Current Trim decode. 0000b - No effect 0001b - Sets the corresponding TX bit

57.4.1.8 TX Control Clear (TX_CLR)

Offset

Register	Offset
TX_CLR	18h

Function

Writing a 1 to a bit in this register clears the corresponding bit in TX register

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserv ed	Reserv ed	Reserved		TXEN CAL...	Reserv ed	TXCAL45DP	
W	Reserved				Reserved				Reserv ed	Reserv ed	Reserved		TXEN CAL...	Reserv ed	TXCAL45DP	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		TXEN CAL...	Reserv ed	TXENCAL45DM				Reserved				D_CAL			
W	Reserved		TXEN CAL...	Reserv ed	TXENCAL45DM				Reserved				D_CAL			
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Fields

Field	Function
31-29 —	Reserved
28-26 —	Reserved
25 —	Reserved
24 —	This bit field must remain at value 1'b0.
23-22 —	Reserved
21 TXENCAL45DP	Enable resistance calibration on DP 0b - No effect 1b - Clears the corresponding TX bit
20 —	Reserved
19-16 TXCAL45DP	DP series termination resistance trim. 0000b - No effect 0001b - Clears the corresponding TX bit
15-14 —	Reserved
13 TXENCAL45DN	Clears Enable resistance calibration on DN. 0b - No effect 1b - Clears the corresponding TX bit
12 —	Reserved
11-8 TXENCAL45D M	DM series termination resistance trim. 0000b - No effect 0001b - Clears the corresponding TX bit
7-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 D_CAL	Current Trim decode. 0000b - No effect 0001b - Clears the corresponding TX bit

57.4.1.9 TX Control Toggle (TX_TOG)

Offset

Register	Offset
TX_TOG	1Ch

Function

Writing a 1 to a bit in this register the corresponding bit in TX register

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				Reserved				Reserv ed	Reserv ed	Reserved		TXEN CAL...	Reserv ed	TXCAL45DP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		TXEN CAL...	Reserv ed	TXENCAL45DM				Reserved				D_CAL			
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Fields

Field	Function
31-29 —	Reserved
28-26 —	Reserved
25 —	Reserved
24	This bit field must remain at value 1'b0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23-22 —	Reserved
21 TXENCAL45DP	Enable resistance calibration on DP 0b - No effect 1b - Toggles the corresponding TX bit
20 —	Reserved
19-16 TXCAL45DP	DP series termination resistance trim. 0000b - No effect 0001b - Toggles the corresponding TX bit
15-14 —	Reserved
13 TXENCAL45DN	Enable resistance calibration on DN. 0b - No effect 1b - Toggles the corresponding TX bit
12 —	Reserved
11-8 TXENCAL45D M	DM series termination resistance trim. 0000b - No effect 0001b - Toggles the corresponding TX bit
7-4 —	Reserved
3-0 D_CAL	Current Trim decode. 0000b - No effect 0001b - Toggles the corresponding TX bit

57.4.1.10 RX Control (RX)

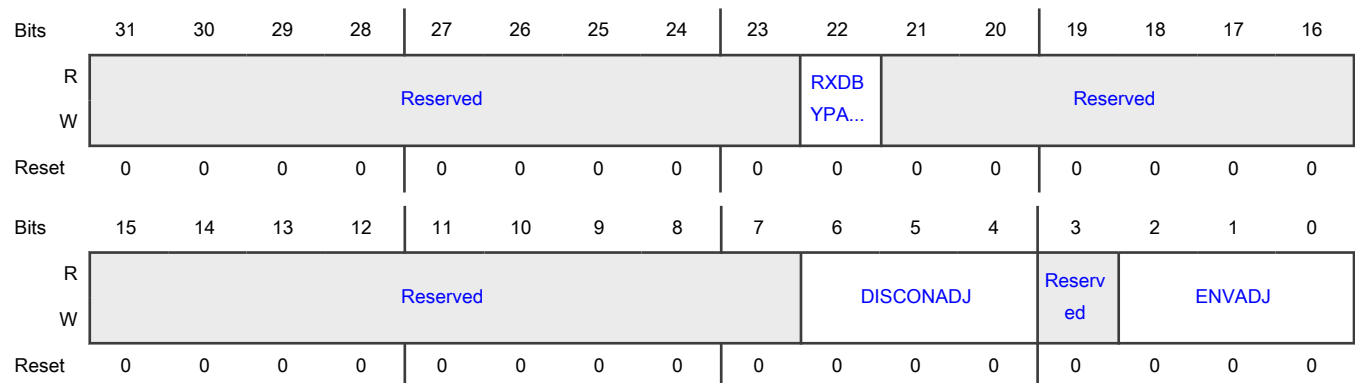
Offset

Register	Offset
RX	20h

Function

The USB PHY Receive Control Register handles receive path controls.

Diagram



Fields

Field	Function
31-23 —	Reserved
22 RXDBYPASS	DM bypass This test mode is intended for lab use only, replace FS differential receiver with DP single ended receiver. 0b - Normal operation 1b - Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver
21-7 —	Reserved
6-4 DISCONADJ	Disconnect detector trip point. The DISCONADJ field adjusts the trip point for the disconnect detector. 000b - Trip-Level Voltage is 0.56875 V 001b - Trip-Level Voltage is 0.55000 V

Table continues on the next page...

Table continued from the previous page...

Field	Function
	010b - Trip-Level Voltage is 0.58125 V 011b - Trip-Level Voltage is 0.60000 V 100b - reserved 101b - reserved 110b - reserved 111b - reserved
3 —	Reserved
2-0 ENVADJ	Envelope detector trip point. The ENVADJ field adjusts the trip point for the envelope detector. Values shown below are nominal DC settings to indicate effect of changing these bits. AC values measured during compliance testing will be somewhat higher. 000b - Trip-Level Voltage is 0.1000 V 001b - Trip-Level Voltage is 0.1125 V 010b - Trip-Level Voltage is 0.1250 V 011b - Trip-Level Voltage is 0.0875 V 100b-111b - Reserved

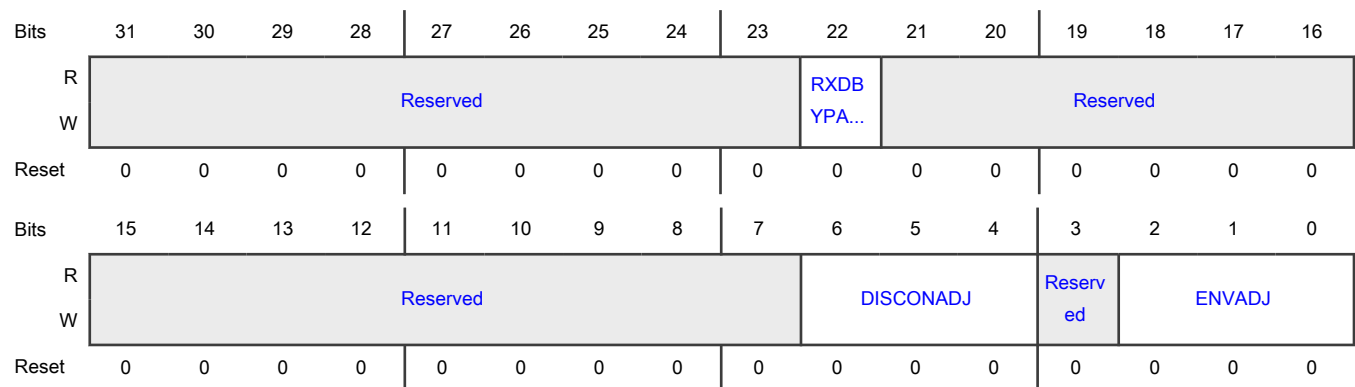
57.4.1.11 RX Control Set (RX_SET)

Offset

Register	Offset
RX_SET	24h

Function

Writing a 1 to a bit position in this register sets the corresponding position in RX register.

Diagram**Fields**

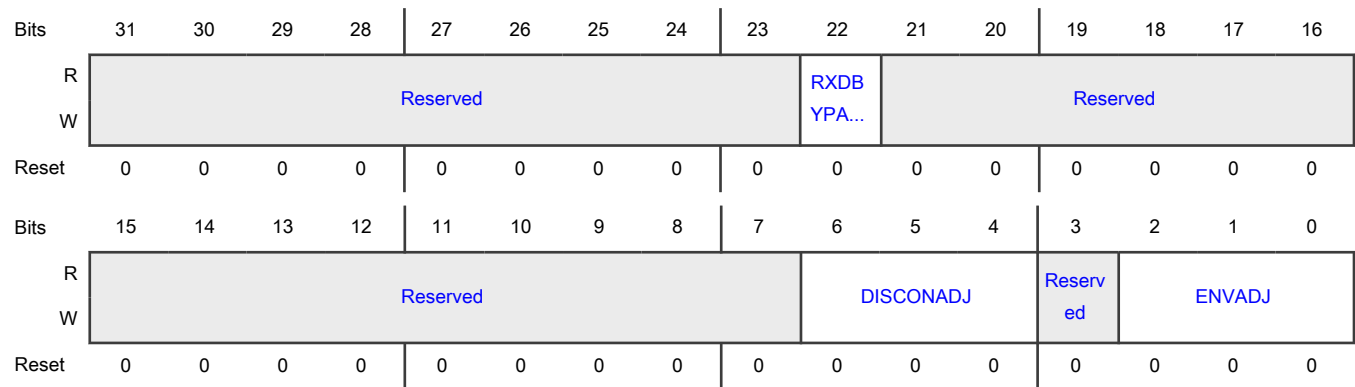
Field	Function
31-23 —	Reserved
22 RXDBYPASS	DM bypass 0b - No effect 1b - Sets the corresponding TX bit
21-7 —	Reserved
6-4 DISCONADJ	Disconnect detector trip point. 000b - No effect 001b - Sets the corresponding TX bit
3 —	Reserved
2-0 ENVADJ	Envelope detector trip point. 000b - No effect 001b - Sets the corresponding TX bit

57.4.1.12 RX Control Clear (RX_CLR)**Offset**

Register	Offset
RX_CLR	28h

Function

Writing a 1 to a bit position in this register clears the corresponding position in RX register.

Diagram**Fields**

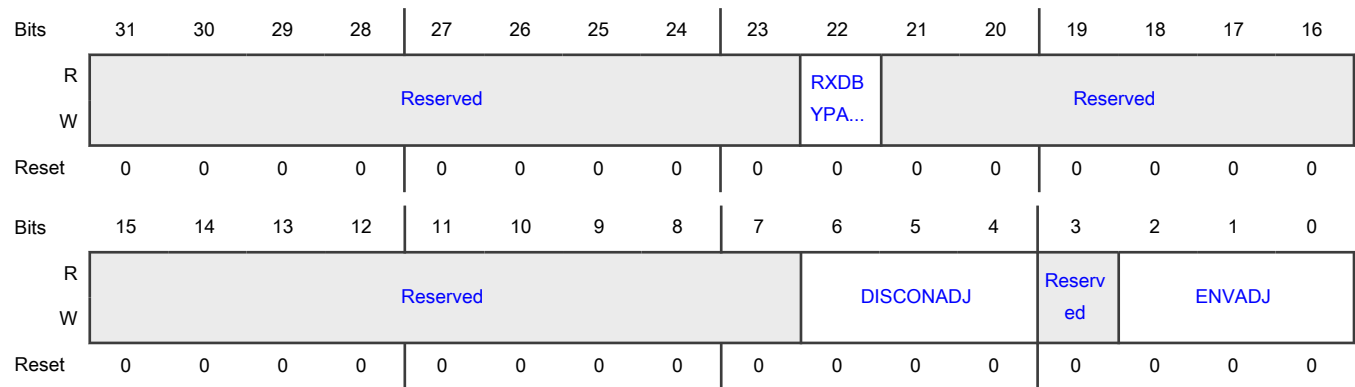
Field	Function
31-23 —	Reserved
22 RXDBYPASS	DM bypass 0b - No effect 1b - Clears the corresponding TX bit
21-7 —	Reserved
6-4 DISCONADJ	Disconnect detector trip point. 000b - No effect 001b - Clears the corresponding TX bit
3 —	Reserved
2-0 ENVADJ	Envelope detector trip point. 000b - No effect 001b - Clears the corresponding TX bit

57.4.1.13 RX Control Toggle (RX_TOG)**Offset**

Register	Offset
RX_TOG	2Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in RX register.

Diagram**Fields**

Field	Function
31-23 —	Reserved
22 RXDBYPASS	DM bypass 0b - No effect 1b - Toggles the corresponding TX bit
21-7 —	Reserved
6-4 DISCONADJ	Disconnect detector trip point. 000b - No effect 001b - Toggles the corresponding TX bit
3 —	Reserved
2-0 ENVADJ	Envelope detector trip point. 000b - No effect 001b - Toggles the corresponding TX bit

57.4.1.14 General Purpose Control (CTRL)**Offset**

Register	Offset
CTRL	30h

Function

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTR	CLKG	UTMI_	HOST	Reserv	Reserv	Reserv	FSDLL	Reserv	Reserv	Reserv	ENAU	ENAU	AUTO	Reserv	Reserv
W	ST	ATE	SU...	_FO...	ed	ed	ed	_R...	ed	ed	ed	TOC...	TOC...	RES...	ed	ed
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUT	ENUT	Reserv	DEVP	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	ENDE	HOST	Reserv	ENHO	Reserv
W	MIL...	MIL...	ed	LUG...	ed	ed	ed	ed	ed	ed	ed	VPL...	DIS...	ed	STD...	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SFTRST	Software reset Writing a 1 to this bit will soft-reset the PWD, TX, RX, and CTRL registers. Set to 0 to release the PHY from reset.
30 CLKGATE	UTMI clock gate Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wakeup event when USB is suspended while ENAUTOCLR_CLKGATE bit of CTRL is enabled.
29 UTMI_SUSPENDM	UTMI suspend Used by the PHY to indicate a powered-down state. If all the power-down bits in the PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28 HOST_FORCE_LS_SE0	FS EOP low-speed timing Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with DEBUG_HOST_RESUME_DEBUG.
27 —	Reserved
26 —	Reserved. This bit field must remain at value 1'b0.

Table continues on the next page...

Table continued from the previous page...

Field	Function
25 —	Reserved
24 FSDLL_RST_EN	Reset FSDLL lock Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.
23 —	Reserved. This bit field must remain at value 1'b0.
22 —	Reserved. This bit field must remain at value 1'b0.
21 —	Reserved. This bit field must remain at value 1'b0.
20 ENAUTOCLR_PWD	Autoclear PWD register bits. Enables the feature to auto-clear the PWD register bits in PWD if there is wakeup event while USB is suspended. This should be enabled if needed to support auto wakeup without software interaction.
19 ENAUTOCLR_CLKGATE	Autoclear clock gate. Enables the feature to auto-clear the CLKGATE bit if there is wakeup event while USB is suspended. This should be enabled if needed to support auto wakeup without software interaction.
18 AUTORESUME_EN	Enable autoresume Enable the auto resume feature. When set, HW will use 32KHz clock to send Resume to respond to the device remote wakeup (for host mode only). It's useful when PLL is off and reference clock is also powered down. NOTE To use the autoresume feature, software needs to: • Detect suspend entry (falling edge of needclk) • Set autoresume_en • Detect auto resume (rising edge of needclk) • Clears autoresume_en
17 —	Reserved. This bit field must remain at value 1'b0.
16 —	Reserved. This bit field must remain at value 1'b0.
15	Enable level 2 operation

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENUTMILEVEL 3	Enables UTMI+ Level 3 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support an external FS Hub with a LS device connected.
14 ENUTMILEVEL 2	Enable level 2 operation Enables UTMI+ Level 2 operation for the USB HS PHY. This should be enabled if an Embedded Host use case needs to support a LS device.
13 —	Reserved
12 DEVPLUGIN_I RQ	Device connected indicator Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 ENDEVPLUGIN DET	Enables non-standard resistive plugged-in detection. This bit field controls connection of nominal 200kΩ resistors to both the USB_DP and USB_DM pins as one method of detecting when a USB cable is attached in device mode. This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the USB Battery Charger Specification Revision 1.2 or any other detection mechanism for USB cable plugin. The results of this detection method are reported in STATUS[6].

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disables 200kΩ pullup resistors on USB_DP and USB_DM pins (Default) 1b - Enables 200kΩ pullup resistors on USB_DP and USB_DM pins (Default)
3 HOSTDISCON DETECT_IRQ	Device disconnect indication. Indicates that the device has disconnected in High-Speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
2 —	Reserved
1 ENHOSTDISC ONDETECT	Disconnect detect. For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. It shall be set after HS device is connected.
0 —	Reserved

57.4.1.15 General Purpose Control Set (CTRL_SET)

Offset

Register	Offset
CTRL_SET	34h

Function

Writing a 1 to a bit position in this register sets the corresponding position in CTRL register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTR	CLKG	UTMI_	HOST	Reserv	Reserv	Reserv	FSDLL	Reserv	Reserv	Reserv	ENAU	ENAU	AUTO	Reserv	Reserv
W	ST	ATE	SU...	_FO...	ed	ed	ed	_R...	ed	ed	ed	TOC...	TOC...	RES...	ed	ed
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUT	ENUT	Reserv	DEVP	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	ENDE	HOST	Reserv	ENHO	Reserv
W	MIL...	MIL...	ed	LUG...	ed	ed	ed	ed	ed	ed	ed	VPL...	DIS...	ed	STD...	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SFTRST	Software reset 0b - No effect 1b - Sets the corresponding CTRL bit
30 CLKGATE	UTMI clock gate 0b - No effect 1b - the corresponding CTRL bit
29 UTMI_SUSPENDM	UTMI suspend Used by the PHY to indicate a powered-down state. If all the power-down bits in the PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification. 0b - No effect 1b - Sets the corresponding CTRL bit
28 HOST_FORCE_LS_SE0	FS EOP low-speed timing
27 —	Reserved
26 —	Reserved. This bit field must remain at value 1'b0.
25 —	Reserved
24 FSDLL_RST_EN	Reset FSDLL lock 0b - No effect 1b - Sets the corresponding CTRL bit
23 —	Reserved. This bit field must remain at value 1'b0.
22 —	Reserved
21 —	Reserved
20	Autoclear PWD register bits.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENAUTOCLR_PHY_PWD	0b - No effect 1b - Sets the corresponding CTRL bit
19 ENAUTOCLR_CLKGATE	Autoclear clock gate. 0b - No effect 1b - the corresponding CTRL bit
18 AUTORESUME_EN	Enable autoresume 0b - No effect 1b - Sets the corresponding CTRL bit
17 —	Reserved
16 —	Reserved
15 ENUTMILEVEL3	Enable level 2 operation 0b - No effect 1b - the corresponding CTRL bit
14 ENUTMILEVEL2	Enable level 2 operation 0b - No effect 1b - Sets the corresponding CTRL bit
13 —	Reserved
12 DEVPLUGIN_IRQ	Device connected indicator 0b - No effect 1b - Sets the corresponding CTRL bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
7 —	Reserved
6 —	Reserved
5 —	Reserved 0b - No effect 1b - Sets the corresponding CTRL bit
4 ENDEVPLUGIN DET	Enables non-standard resistive plugged-in detection. 0b - No effect 1b - Sets the corresponding CTRL bit
3 HOSTDISCON DETECT_IRQ	Device disconnect indication. 0b - No effect 1b - Sets the corresponding CTRL bit
2 —	Reserved 0b - No effect 1b - Sets the corresponding CTRL bit
1 ENHOSTDISC ONDETECT	Disconnect detect. 0b - No effect 1b - Sets the corresponding CTRL bit
0 —	Reserved

57.4.1.16 General Purpose Control Clear (CTRL_CLR)

Offset

Register	Offset
CTRL_CLR	38h

Function

Writing a 1 to a bit position in this register clears the corresponding position in CTRL register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTR	CLKG	UTMI_	HOST	Reserv	Reserv	Reserv	FSDLL	Reserv	Reserv	Reserv	ENAU	ENAU	AUTO	Reserv	Reserv
W	ST	ATE	SU...	_FO...	ed	ed	ed	_R...	ed	ed	ed	TOC...	TOC...	RES...	ed	ed
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUT	ENUT	Reserv	DEVP	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	ENDE	HOST	Reserv	ENHO	Reserv
W	MIL...	MIL...	ed	LUG...	ed	ed	ed	ed	ed	ed	ed	VPL...	DIS...	ed	STD...	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SFTRST	Software reset 0b - No effect 1b - Clears the corresponding CTRL bit
30 CLKGATE	UTMI clock gate 0b - No effect 1b - Clears the corresponding CTRL bit
29 UTMI_SUSPENDM	UTMI suspend Used by the PHY to indicate a powered-down state. If all the power-down bits in the PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification. 0b - No effect 1b - Clears the corresponding CTRL bit
28 HOST_FORCE_LS_SE0	FS EOP low-speed timing
27 —	Reserved
26 —	Reserved. This bit field must remain at value 1'b0.
25 —	Reserved
24	Reset FSDLL lock

Table continues on the next page...

Table continued from the previous page...

Field	Function
FSDLL_RST_EN	0b - No effect 1b - Clears the corresponding CTRL bit
23 —	Reserved. This bit field must remain at value 1'b0.
22 —	Reserved
21 —	Reserved
20 ENAUTOCLR_PHY_PWD	Autoclear PWD register bits. 0b - No effect 1b - Clears the corresponding CTRL bit
19 ENAUTOCLR_CLKGATE	Autoclear clock gate. 0b - No effect 1b - Clears the corresponding CTRL bit
18 AUTORESUME_EN	Enable autoresume 0b - No effect 1b - Clears the corresponding CTRL bit
17 —	Reserved
16 —	Reserved
15 ENUTMILEVEL3	Enable level 2 operation 0b - No effect 1b - Clears the corresponding CTRL bit
14 ENUTMILEVEL2	Enable level 2 operation 0b - No effect 1b - Clears the corresponding CTRL bit
13 —	Reserved
12	Device connected indicator

Table continues on the next page...

Table continued from the previous page...

Field	Function
DEVPLUGIN_I RQ	0b - No effect 1b - Clears the corresponding CTRL bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved 0b - No effect 1b - Clears the corresponding CTRL bit
4 ENDEVP LUGIN DET	Enables non-standard resistive plugged-in detection. 0b - No effect 1b - Clears the corresponding CTRL bit
3 HOSTDISCON DETECT_IRQ	Device disconnect indication. 0b - No effect 1b - Clears the corresponding CTRL bit
2 —	Reserved 0b - No effect 1b - Clears the corresponding CTRL bit
1 ENHOSTDISC ONDETECT	Disconnect detect. 0b - No effect 1b - Clears the corresponding CTRL bit
0	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	

57.4.1.17 General Purpose Control Toggle (CTRL_TOG)

Offset

Register	Offset
CTRL_TOG	3Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in CTRL register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTR	CLKG	UTMI_	HOST	Reserv	Reserv	Reserv	FSDLL	Reserv	Reserv	Reserv	ENAU	ENAU	AUTO	Reserv	Reserv
W	ST	ATE	SU...	_FO...	ed	ed	ed	_R...	ed	ed	ed	TOC...	TOC...	RES...	ed	ed
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUT	ENUT	Reserv	DEVP	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	Reserv	ENDE	HOST	Reserv	ENHO	Reserv
W	MIL...	MIL...	ed	LUG...	ed	ed	ed	ed	ed	ed	ed	VPL...	DIS...	ed	STD...	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 SFTRST	Software reset 0b - No effect 1b - Toggles the corresponding CTRL bit
30 CLKGATE	UTMI clock gate 0b - No effect 1b - Toggles the corresponding CTRL bit
29 UTMI_SUSPENDM	UTMI suspend Used by the PHY to indicate a powered-down state. If all the power-down bits in the PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1 when USB controller entering into Suspend mode. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No effect 1b - Toggles the corresponding CTRL bit
28 HOST_FORCE _LS_SE0	FS EOP low-speed timing
27 —	Reserved
26 —	Reserved. This bit field must remain at value 1'b0.
25 —	Reserved
24 FSDLL_RST_E N	Reset FSDLL lock 0b - No effect 1b - Toggles the corresponding CTRL bit
23 —	Reserved. This bit field must remain at value 1'b0.
22 —	Reserved
21 —	Reserved
20 ENAUTOCLR_ PHY_PWD	Autoclear PWD register bits. 0b - No effect 1b - Toggles the corresponding CTRL bit
19 ENAUTOCLR_ CLKGATE	Autoclear clock gate. 0b - No effect 1b - the corresponding CTRL bit
18 AUTORESUME _EN	Enable autoresume 0b - No effect 1b - Toggles the corresponding CTRL bit
17	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
16 —	Reserved
15 ENUTMILEVEL 3	Enable level 2 operation 0b - No effect 1b - Toggles the corresponding CTRL bit
14 ENUTMILEVEL 2	Enable level 2 operation 0b - No effect 1b - Toggles the corresponding CTRL bit
13 —	Reserved
12 DEVPLUGIN_I RQ	Device connected indicator 0b - No effect 1b - Toggles the corresponding CTRL bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved 0b - No effect 1b - Toggles the corresponding CTRL bit
4	Enables non-standard resistive plugged-in detection.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENDEVPLUGIN DET	0b - No effect 1b - Toggles the corresponding CTRL bit
3 HOSTDISCON DETECT_IRQ	Device disconnect indication. 0b - No effect 1b - Toggles the corresponding CTRL bit
2 —	Reserved 0b - No effect 1b - Toggles the corresponding CTRL bit
1 ENHOSTDISC ONDETECT	Disconnect detect. 0b - No effect 1b - Toggles the corresponding CTRL bit
0 —	Reserved

57.4.1.18 Status (STATUS)

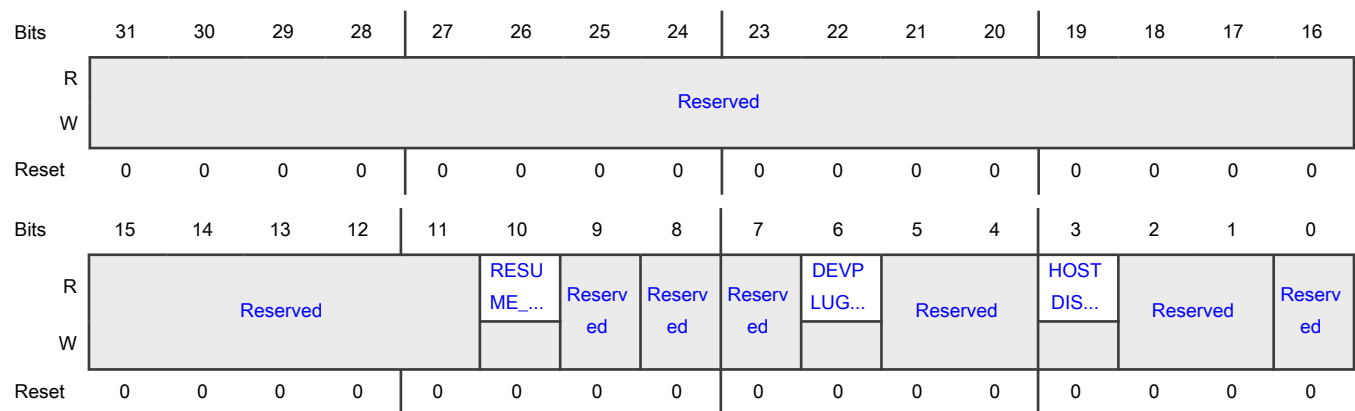
Offset

Register	Offset
STATUS	40h

Function

The USB PHY Status Register holds results of IRQ and other detects.

Diagram



Fields

Field	Function
31-11 —	Reserved
10 RESUME_STATUS	Resume status Indicates that the host is sending a wake-up after Suspend and has triggered an interrupt.
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 DEVPLUGIN_STATUS	Status indicator for non-standard resistive plugged-in detection. Indicates that the device has been connected on the USB_DP and USB_DM lines using the nonstandard resistive plugged-in detection method controlled by CTRL[4]. When a USB cable attached to a remote host is attached to the local device, the 15kΩ host pulldowns will override the high value resistors used in this detection method. 0b - No attachment to a USB host is detected 1b - Cable attachment to a USB host is detected
5-4 —	Reserved
3 HOSTDISCON_DETECT_STATUS	Host disconnect status Indicates at the local host (downstream) port that the remote device has disconnected while in High-Speed mode. 0b - USB cable disconnect has not been detected at the local host 1b - USB cable disconnect has been detected at the local host
2-1 —	Reserved
0 —	Reserved

57.4.1.19 Debug 0 (DEBUG0)

Offset

Register	Offset
DEBUG0	50h

Function

This register is used to debug the USB PHY.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	CLKGATE	HOST_RESET	SQUELCHRESETLENGTH				ENSQUELCHRESET	Reserved				SQUELCHRESETCOUNT			
W																
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			ENTX2_RX	TX2RXCOUNT				Reserved		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG_I	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 CLKGATE	Test clock gate Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29 HOST_RESUME_DEBUG	Host resume Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28-25 SQUELCHRESETLENGTH	Squelch reset length Duration of RESET in terms of the number of 480-MHz cycles.
24	Enable squelch reset Set bit to allow squelch to reset high-speed receive.

Table continues on the next page...

Table continued from the previous page...

Field	Function
ENSQUELCHR ESET	
23-21 —	Reserved
20-16 SQUELCHRES ETCOUNT	Squelch reset count Delay in between the detection of squelch to the reset of high-speed RX.
15-13 —	Reserved
12 ENTX2RXCOU NT	Set this bit to allow a countdown to transition in between TX and RX.
11-8 TX2RXCOUNT	TX2RXCOUNT Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7-6 —	Reserved
5-4 ENHSTPULLD OWN	Enable Host pulldown This bit field selects host pulldown overdrive mode. Set bit 5 to value 1'b1 to override the control of the USB_DP 15kΩ pulldown. Set bit 4 to value 1'b1 to override the control of the USB_DP 15kΩ pulldown. Clear both bits to value 2'b00 to disable the host pulldown overdrive mode. When in host pulldown overdrive mode, the connection of the individual pulldown resistors is further controlled by the DEBUG[3:2] bit field.
3-2 HSTPULLDOW N	HS DP/DM pulldown resistance select. This bit field selects whether to connect pulldown resistors on the USB_DP/USB_DM pins if the corresponding pulldown overdrive mode is enabled through DEBUG[5:4] Set bit 3 to value 1'b1 to connect the 15Ω pulldown on USB_DP line. Set bit 2 to value 1'b1 to connect the 15Ω pulldown on the USB_DM line. Clear both bits to value 2'b00 to disconnect the pulldowns in override mode.
1 DEBUG_INTER FACE_HOLD	Debug interface Use holding registers to assist in timing for external UTMI interface.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 —	Reserved

57.4.1.20 Debug 0 Set (DEBUG0_SET)

Offset

Register	Offset
DEBUG0_SET	54h

Function

Writing a 1 to a bit position in this register sets the corresponding position in DEBUG0 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	CLKGATE	HOST_RESUME_DEBUG	SQUELCHRESETLENGTH				ENSQUELCHRESETCOUNT	Reserved				SQUELCHRESETCOUNT			
W																
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			ENTX2RXCOUNT	TX2RXCOUNT				Reserved		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG0_SET	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 CLKGATE	Test clock gate 0b - No effect 1b - Sets the corresponding CTRL bit
29 HOST_RESUME_DEBUG	Host resume 0b - No effect 1b - Sets the corresponding CTRL bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
28-25 SQUELCHRES ETLENGTH	Squelch reset length 0000b - No effect 0001b - Sets the corresponding CTRL bit
24 ENSQUELCHR ESET	Enable squelch reset 0b - No effect 1b - Sets the corresponding CTRL bit
23-21 —	Reserved
20-16 SQUELCHRES ETCOUNT	Squelch reset count 0_0000b - No effect 0_0001b - Sets the corresponding CTRL bit
15-13 —	Reserved
12 ENTX2RXCOU NT	ENTX2RXCOUNT 0b - No effect 1b - Sets the corresponding CTRL bit
11-8 TX2RXCOUNT	TX2RXCOUNT 0000b - No effect 0001b - Sets the corresponding CTRL bit
7-6 —	Reserved
5-4 ENHSTPULLD OWN	Enable Host pulldown 00b - No effect 01b - Sets the corresponding CTRL bit
3-2 HSTPULLDOW N	HS DP/DM pulldown resistance select. 00b - No effect 01b - Sets the corresponding CTRL bit
1 DEBUG_INTER FACE_HOLD	Debug interface 0b - No effect 1b - Sets the corresponding CTRL bit
0	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	

57.4.1.21 Debug Clear (DEBUG0_CLR)

Offset

Register	Offset
DEBUG0_CLR	58h

Function

Writing a 1 to a bit position in this register clears the corresponding position in DEBUG0 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	CLKG	HOST	SQUELCHRESETLENGTH				ENSQ	Reserved			SQUELCHRESETCOUNT				
W	ed	ATE	_RE...					UEL...								
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			ENTX2	TX2RXCOUNT				Reserved		ENHSTPULLD	HSTPULLDOWN		DEBU	Reserv	
W				RX...							OWN			G_I...	ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 CLKGATE	Test clock gate 0b - No effect 1b - Clears the corresponding CTRL bit
29 HOST_RESUME_DEBUG	Host resume 0b - No effect 1b - Clears the corresponding CTRL bit
28-25	Squelch reset length

Table continues on the next page...

Table continued from the previous page...

Field	Function
SQUELCHRES ETLENGTH	0000b - No effect 0001b - Clears the corresponding CTRL bit
24 ENSQUELCHRE ESET	Enable squelch reset 0b - No effect 1b - Clears the corresponding CTRL bit
23-21 —	Reserved
20-16 SQUELCHRES ETCOUNT	Squelch reset count 0_0000b - No effect 0_0001b - Clears the corresponding CTRL bit
15-13 —	Reserved
12 ENTX2RXCOU NT	ENTX2RXCOUNT 0b - No effect 1b - Clears the corresponding CTRL bit
11-8 TX2RXCOUNT	TX2RXCOUNT 0000b - No effect 0001b - Clears the corresponding CTRL bit
7-6 —	Reserved
5-4 ENHSTPULLD OWN	Enable Host pulldown 00b - No effect 01b - Clears the corresponding CTRL bit
3-2 HSTPULLDOW N	HS DP/DM pulldown resistance select. 00b - No effect 01b - Clears the corresponding CTRL bit
1 DEBUG_INTER FACE_HOLD	Debug interface 0b - No effect 1b - Clears the corresponding CTRL bit
0 —	Reserved

57.4.1.22 Debug Toggle (DEBUG0_TOG)

Offset

Register	Offset
DEBUG0_TOG	5Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in DEBUG0 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	CLKGATE	HOST_RESET	SQUELCHRESETLENGTH				ENSQUEL	Reserved				SQUELCHRESETCOUNT			
W																
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			ENTX2_RX	TX2RXCOUNT				Reserved		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG_I	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 CLKGATE	Test clock gate 0b - No effect 1b - Toggles the corresponding CTRL bit
29 HOST_RESUME_DEBUG	Host resume 0b - No effect 1b - Toggles the corresponding CTRL bit
28-25 SQUELCHRESETLENGTH	Squelch reset length 0000b - No effect 0001b - the corresponding CTRL bit
24 ENSQUELCHRESET	Enable squelch reset 0b - No effect 1b - Toggles the corresponding CTRL bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-21 —	Reserved
20-16 SQUELCHRES ETCOUNT	Squelch reset count 0_0000b - No effect 0_0001b - Toggles the corresponding CTRL bit
15-13 —	Reserved
12 ENTX2RXCOU NT	ENTX2RXCOUNT 0b - No effect 1b - Toggles the corresponding CTRL bit
11-8 TX2RXCOUNT	TX2RXCOUNT 0000b - No effect 0001b - Toggles the corresponding CTRL bit
7-6 —	Reserved
5-4 ENHSTPULLD OWN	Enable Host pulldown 00b - No effect 01b - Toggles the corresponding CTRL bit
3-2 HSTPULLDOW N	HS DP/DM pulldown resistance select. 00b - No effect 01b - Toggles the corresponding CTRL bit
1 DEBUG_INTER FACE_HOLD	Debug interface 0b - No effect 1b - Toggles the corresponding CTRL bit
0 —	Reserved

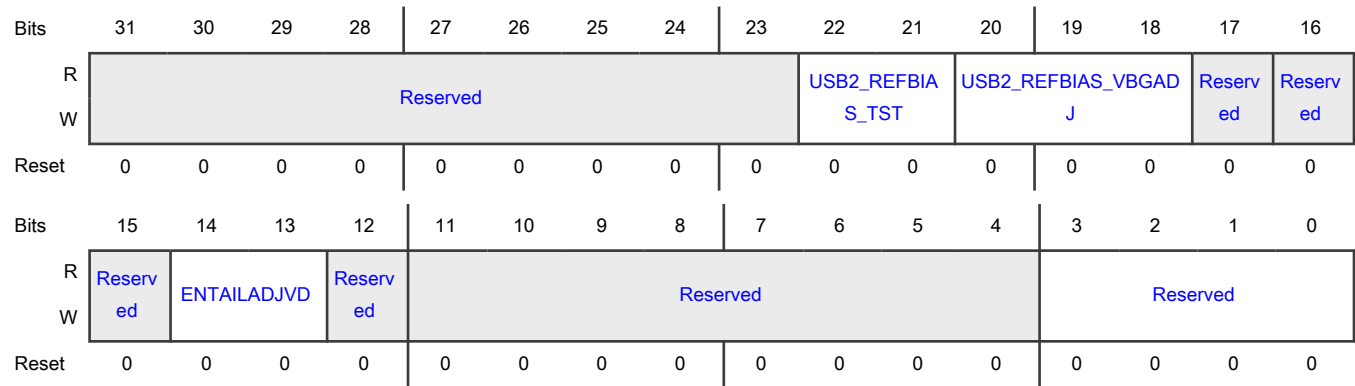
57.4.1.23 UTMI Debug 1 (DEBUG1)

Offset

Register	Offset
DEBUG1	70h

Function

Chooses the muxing of the debug register to be shown in DEBUG0_STATUS register.

Diagram**Fields**

Field	Function
31-23 —	Reserved
22-21 USB2_REFBIA S_TST	Bias current control Bias current control for usb2_phy and usb_PLL
20-18 USB2_REFBIA S_VBGADJ	Bandgap adjustment Adjustment bits on bandgap
17 —	Reserved. This field must remain at 1'b0.
16 —	Reserved. This field must remain at 1'b0.
15 —	Reserved. This field must remain at 1'b0.
14-13 ENTAILADJVD	Enable delay increment Delay increment of the rise of squelch: 00b - Delay is nominal 01b - Delay is +20% 10b - Delay is -20%

Table continues on the next page...

Table continued from the previous page...

Field	Function
	11b - Delay is -40%
12 —	Reserved
11-4 —	Reserved
3-0 —	Reserved

57.4.1.24 UTMI Debug 1 Set (DEBUG1_SET)

Offset

Register	Offset
DEBUG1_SET	74h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the DEBUG1 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								USB2_REFBIA		USB2_REFBIAS_VBGAD		Reserv		Reserv	
W	Reserved								S_TST		J		ed		ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	ENTAILADJVD		Reserv	Reserved								Reserved			
W	ed			ed												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-23 —	Reserved
22-21	Bias current control

Table continues on the next page...

Table continued from the previous page...

Field	Function
USB2_REFBIA S_TST	00b - No effect 01b - Sets the corresponding DEBUG1 bit
20-18 USB2_REFBIA S_VBGADJ	Bandgap adjustment 000b - No effect 001b - Sets the corresponding DEBUG1 bit
17 —	Reserved. This field must remain at 1'b0.
16 —	Reserved. This field must remain at 1'b0.
15 —	Reserved. This field must remain at 1'b0.
14-13 ENTAILADJVD	Enable delay increment 00b - No effect 01b - Sets the corresponding DEBUG1 bit
12 —	Reserved
11-4 —	Reserved
3-0 —	Reserved

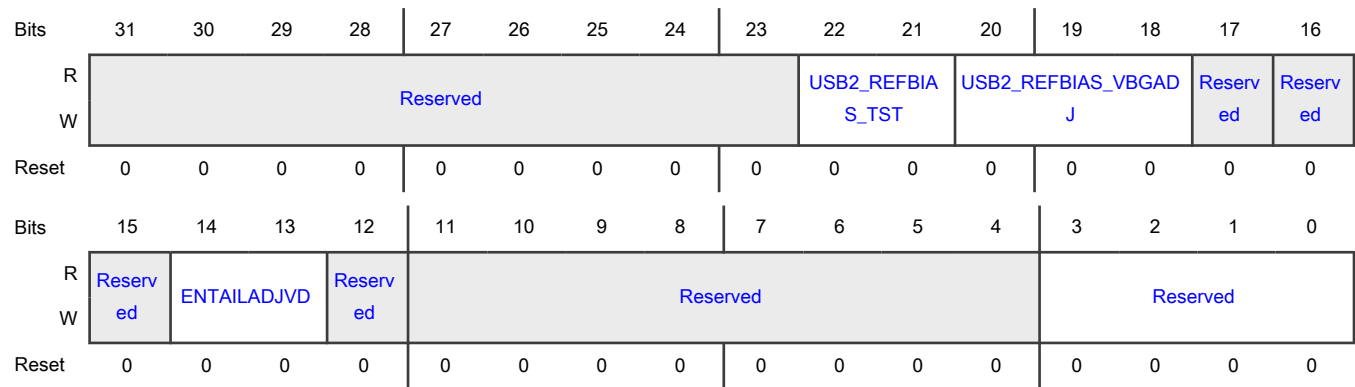
57.4.1.25 UTMI Debug 1 Clear (DEBUG1_CLR)

Offset

Register	Offset
DEBUG1_CLR	78h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the DEBUG1 register.

Diagram**Fields**

Field	Function
31-23 —	Reserved
22-21 USB2_REFBIA S_TST	Bias current control 00b - No effect 01b - Clears the corresponding DEBUG1 bit
20-18 USB2_REFBIA S_VBGADJ	Bandgap adjustment 000b - No effect 001b - Clears the corresponding DEBUG1 bit
17 —	Reserved. This field must remain at 1'b0.
16 —	Reserved. This field must remain at 1'b0.
15 —	Reserved. This field must remain at 1'b0.
14-13 ENTAILADJVD	Enable delay increment 00b - No effect 01b - Clears the corresponding DEBUG1 bit
12 —	Reserved
11-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 —	Reserved

57.4.1.26 UTMI Debug 1 Toggle (DEBUG1_TOG)

Offset

Register	Offset
DEBUG1_TOG	7Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the DEBUG1 register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								USB2_REFBIA S_TST		USB2_REFBIA_VBGAD J		Reserv ed		Reserv ed	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	ENTAILADJVD		Reserv ed	Reserved								Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-23 —	Reserved
22-21 USB2_REFBIA S_TST	Bias current control 00b - No effect 01b - Toggles the corresponding DEBUG1 bit
20-18 USB2_REFBIA S_VBGADJ	Bandgap adjustment 000b - No effect 001b - Toggles the corresponding DEBUG1 bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 —	Reserved. This field must remain at 1'b0.
16 —	Reserved. This field must remain at 1'b0.
15 —	Reserved. This field must remain at 1'b0.
14-13 ENTAILADJVD	Enable delay increment 00b - No effect 01b - Clears the corresponding DEBUG1 bit
12 —	Reserved
11-4 —	Reserved
3-0 —	Reserved

57.4.1.27 Version (VERSION)

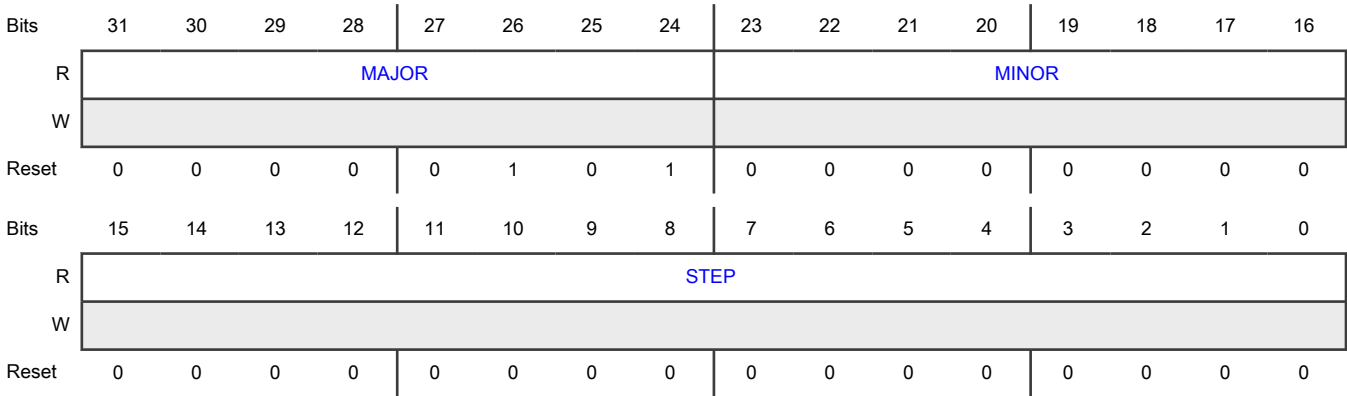
Offset

Register	Offset
VERSION	80h

Function

The version of USBPHY.

Diagram



Fields

Field	Function
31-24 MAJOR	MAJOR
23-16 MINOR	MINOR
15-0 STEP	STEP

57.4.1.28 PLL Control/Status (PLL_SIC)

Offset

Register	Offset
PLL_SIC	A0h

Function

This register configures the 480 MHz USB PHY PLL.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLL_LOCK	Reserved	Reserved				PLL_DIV_SEL				PLL_REFCLK_SEL	REFBIAS_OVERRIDE	REFBIAS_OVERRIDE	Reserved	Reserved	PLL_BYPASS
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		PLL_ENABLE	PLL_POWERDOWN	Reserved	Reserved	Reserved	Reserved	Reserved	PLL_ENABLE	Reserved					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PLL_LOCK	USB PLL lock status indicator 0 - PLL is not currently locked 1 - PLL is currently locked
30 —	Reserved
29-25 —	Reserved
24-22 PLL_DIV_SEL	PLL Divider value This field controls the USB PLL feedback loop divider. Valid range for divider values: 54-108. $F_{out} = F_{in} * \text{div\_select}/2.0$. The USB PLL is designed to produce a 480MHz output clock. This bit field allows use of different frequency signals for the PLL reference clock input connected to the OSCCLK signal from the system oscillator. When override is enabled through TRIM_OVERRIDE_EN[0], the USB PLL will use this register value. 000b - Divide by 13 001b - Divide by 15 010b - Divide by 16 011b - Divide by 20 100b - Divide by 22 101b - Divide by 25 110b - Divide by 30 111b - Divide by 240
21	Enable PLL regulator

Table continues on the next page...

Table continued from the previous page...

Field	Function
PLL_REG_ENA BLE	This field controls the USB PLL regulator, set to enable the regulator. SW must set this bit 15 μ s before setting PLL_POWER to avoid glitches on PLL output clock.
20 REFBIAS_PWD	Power down Reference bias This bit is only used when REFBIAS_PWD_SEL is set to 1.
19 REFBIAS_PWD _SEL	Reference bias power control Reference bias power down select. 0b - Selects PLL_POWER to control the reference bias 1b - Selects REFBIAS_PWD to control the reference bias
18 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
17 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
16 PLL_BYPASS	Bypass USB PLL 0b - Use USB PLL 1b - Bypass USB PLL
15-14 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
13 PLL_ENABLE	PLL enable Enables the clock output from the USB PLL. The real PLL output enable signal is also controlled by PLL power up control. Hardware will disable the PLL output before power down PLL, and enable the PLL output after power up PLL. The software only needs to set it when initializing the PLL.
12 PLL_POWER	Power PLL Power up the USB PLL. The real PLL power up is also controlled by hardware. Hardware will power down PLL when USB is suspended and the device doesn't use it. NOTE The description for this bit field is only true if PLL_SIC[5] is high, which is not its default condition.
11 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
10 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
9	Reserved. This bit field must remain at value 1'b0 for normal operation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
8 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
7 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
6 PLL_EN_USB_CLKS	PLL clock enable Enables the USB clock from PLL to USB PHY.
5-0 —	Reserved. This bit field must remain at value 1'b0 for normal operation.

57.4.1.29 PLL Control/Status Set (PLL_SIC_SET)

Offset

Register	Offset
PLL_SIC_SET	A4h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the PLL_SIC register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLL_LOCK	Reserved	Reserved				PLL_DIV_SEL				PLL_REG...	REFBI_AS...	REFBI_AS...	Reserved	Reserved	PLL_BYP...
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		PLL_ENA...	PLL_POW...	Reserved	Reserved	Reserved	Reserved	Reserved	PLL_EN...	Reserved					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PLL_LOCK	PLL_LOCK 0b - No effect 1b - Sets the corresponding bit
30 —	Reserved
29-25 —	Reserved
24-22 PLL_DIV_SEL	PLL_DIV_SEL 000b - No effect 001b - Sets the corresponding bit
21 PLL_REG_ENA BLE	PLL_REG_ENABLE 0b - No effect 1b - Sets the corresponding bit
20 REFBIAS_PWD	REFBIAS_PWD 0b - No effect 1b - Sets the corresponding bit
19 REFBIAS_PWD _SEL	REFBIAS_PWD_SEL 0b - No effect 1b - Sets the corresponding bit
18 —	Reserved
17 —	Reserved
16 PLL_BYPASS	Bypass USB PLL 0b - No effect 1b - Sets the corresponding bit
15-14 —	Reserved
13 PLL_ENABLE	ENABLE 0b - No effect 1b - Sets the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 PLL_POWER	POWER 0b - No effect 1b - Sets the corresponding bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
7 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
6 PLL_EN_USB_CLKS	PLL_EN_USB_CLKS 0b - No effect 1b - Sets the corresponding bit
5-0 —	Reserved

57.4.1.30 PLL Control/Status Clear (PLL_SIC_CLR)

Offset

Register	Offset
PLL_SIC_CLR	A8h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the PLL_SIC register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLL_LOCK	Reserved	Reserved				PLL_DIV_SEL				PLL_REG_ENABLE	REFBIAS_PWD	REFBIAS_PWD_SEL	Reserved	Reserved	PLL_BYPASS
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		PLL_ENA	PLL_POW	Reserved	Reserved	Reserved	Reserved	Reserved	PLL_ENA	Reserved					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PLL_LOCK	PLL_LOCK 0b - No effect 1b - Clears the corresponding bit
30 —	Reserved
29-25 —	Reserved
24-22 PLL_DIV_SEL	PLL_DIV_SEL 000b - No effect 001b - Clears the corresponding bit
21 PLL_REG_ENABLE	PLL_REG_ENABLE 0b - No effect 1b - Clears the corresponding bit
20 REFBIAS_PWD	REFBIAS_PWD 0b - No effect 1b - Clears the corresponding bit
19 REFBIAS_PWD_SEL	REFBIAS_PWD_SEL 0b - No effect 1b - Clears the corresponding bit
18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
17 —	Reserved
16 PLL_BYPASS	Bypass USB PLL 0b - No effect 1b - Clears the corresponding bit
15-14 —	Reserved
13 PLL_ENABLE	ENABLE 0b - No effect 1b - Clears the corresponding bit
12 PLL_POWER	POWER 0b - No effect 1b - Clears the corresponding bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
7 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
6 PLL_EN_USB_CLKS	PLL_EN_USB_CLKS 0b - No effect 1b - Clears the corresponding bit
5-0 —	Reserved

57.4.1.31 PLL Control/Status Toggle (PLL_SIC_TOG)

Offset

Register	Offset
PLL_SIC_TOG	ACh

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the PLL_SIC register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PLL_LOCK	Reserved	Reserved				PLL_DIV_SEL				PLL_REG_ENABLE	REFBIAS	REFBIAS	Reserved	Reserved	PLL_BYPASS
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		PLL_ENABLE	PLL_POWDOWN	Reserved	Reserved	Reserved	Reserved	Reserved	PLL_ENABLE	Reserved					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 PLL_LOCK	PLL_LOCK 0b - No effect 1b - Toggles the corresponding bit
30 —	Reserved
29-25 —	Reserved
24-22 PLL_DIV_SEL	PLL_DIV_SEL 000b - No effect 001b - Toggles the corresponding bit
21 PLL_REG_ENABLE	PLL_REG_ENABLE 0b - No effect 1b - Toggles the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
20 REFBIAS_PWD	REFBIAS_PWD 0b - No effect 1b - Toggles the corresponding bit
19 REFBIAS_PWD_SEL	REFBIAS_PWD_SEL 0b - No effect 1b - Toggles the corresponding bit
18 —	Reserved
17 —	Reserved
16 PLL_BYPASS	Bypass USB PLL 0b - No effect 1b - Toggles the corresponding bit
15-14 —	Reserved
13 PLL_ENABLE	PLL_ENABLE 0b - No effect 1b - Toggles the corresponding bit
12 PLL_POWER	POWER 0b - No effect 1b - Toggles the corresponding bit
11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved. This bit field must remain at value 1'b0 for normal operation.
7 —	Reserved. This bit field must remain at value 1'b0 for normal operation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
6 PLL_EN_USB_CLKS	PLL_EN_USB_CLKS 0b - No effect 1b - Toggles the corresponding bit
5-0 —	Reserved

57.4.1.32 VBUS detect (USB1_VBUS_DETECT)

Offset

Register	Offset
USB1_VBUS_DETECT	C0h

Function

This register defines controls for USB VBUS detect and some additional out of band signaling functions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN_C	0			Reserv	DISCH	0			Reserved			PWRU	Reserv	VBUS	0
W	HAR...				ed	AR...							P_C...	ed	VAL...	
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				VBUS_SOURC			VBUS	VBUS	AVALI	BVALI	SESS	VBUS_	VBUSVALID_THRESH		
W					E_SEL			VAL...	VAL...	D_...	D_...	END...	OV...			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31 EN_CHARGER_RESISTOR	Enables resistors used for an older method of resistive battery charger detection This bit field enables a nominal 125kΩ pullup on DP and a nominal 375kΩ pulldown on DN. This allows resistive battery charger detection per the <i>USB Battery Charging Specification Revision 1.0</i> which is now obsolete. Detection of DP and DN pin states using this method is done with the CHRG_DET_STAT[3:2] bit fields.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This bit field must remain at a value of 1'b0 for normal USB data communication, or when using the USBHSDCD module for battery charger detection per the <i>USB Battery Charger Specification Revision 1.2</i> or any other detection mechanism for USB cable plugin. 0b - Disable resistive charger detection resistors on DP and DP 1b - Enable resistive charger detection resistors on DP and DP
30-28 —	Reserved
27 —	Reserved This bit field has no effect for this module.
26 DISCHARGE_V BUS	Controls VBUS discharge resistor This bit field controls a nominal 22kΩ resistor between the USB1_VBUS pin and ground. It can be used to accelerate the fall of the VBUS signal at the end of a session. 0b - VBUS discharge resistor is disabled (Default) 1b - VBUS discharge resistor is enabled
25-23 —	Reserved
22-21 —	Reserved This bit field has no effect for this module.
20 PWRUP_CMPS	Enables the VBUS_VALID comparator Powers up the comparator used for the VBUS_VALID detector. This bit field can be reset to value 1'b0 to save power if the internal VBUS_VALID comparator is not used. 0b - Powers down the VBUS_VALID comparator 1b - Enables the VBUS_VALID comparator (default)
19 —	Reserved This bit field has no effect for this module.
18 VBUSVALID_T O_SESSVALID	Selects the comparator used for VBUS_VALID This bit field controls the comparator used to report the VBUS_VALID results in USB1_VBUS_DETECT[3] between the VBUS_VALID comparator and the Session Valid comparator. The VBUS_VALID comparator is the most accurate and has a programmable threshold set by USB_VBUS_DETECT[2:0]. The Session Valid comparator may be useful in systems using non-standard VBUS voltages. The mux selection in this bit field happens before any VBUS_VALID selection controlled by the USB1_VBUS_DETECT[10:8] bits.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Use the VBUS_VALID comparator for VBUS_VALID results 1b - Use the Session End comparator for VBUS_VALID results. The Session End threshold is >0.8V and <4.0V.
17-11 —	Reserved
10-9 VBUS_SOURCE_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selections in this bit field only take effect if both USB1_VBUS_DETECT[8] and USB1_VBUS_DETECT[3] each have the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USB1_VBUS_DET_STAT[3]. 00b - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default) 01b - Use the Session Valid comparator results for signal reported to the USB controller 10b - Use the Session Valid comparator results for signal reported to the USB controller 11b - Reserved, do not use
8 VBUSVALID_SELECTOR	Selects the source of the VBUS_VALID signal reported to the USB controller This is one of the bit fields that selects the source of the VBUS_VALID signal reported to the USB controller. The VBUS_VALID source selection in this bit field only takes effect if USB1_VBUS_DETECT[3] has the value 1'b0. This bit field does not impact the VBUS_VALID value reported in USB1_VBUS_DET_STAT[3]. 0b - Use the VBUS_VALID comparator results for signal reported to the USB controller (Default) 1b - Use the VBUS_VALID_3V detector results for signal reported to the USB controller
7 VBUSVALID_OVERRIDE	Override value for VBUS_VALID signal sent to USB controller The bit field provides the value for VBUS_VALID reported to the USB controller if the value of USB1_VBUS_DETECT[3] is set to 1'b1. The value of this bit field does not affect the value of USB1_VBUS_DET_STAT[3].
6 AVALID_OVERRIDE	Override value for A-Device Session Valid The bit field provides the value for USB1_VBUS_DET_STAT[2] if USB_VBUS_DETECT[3] is set to value 1'b1.
5 BVALID_OVERRIDE	Override value for B-Device Session Valid The bit field provides the value for USB1_VBUS_DET_STAT[1] if USB_VBUS_DETECT[3] is set to value 1'b1.
4	Override value for SESEND

Table continues on the next page...

Table continued from the previous page...

Field	Function
SESSEND_OV ERRIDE	The bit field provides the value for USB1_VBUS_DET_STAT[0] if USB_VBUS_DETECT[3] is set to value 1'b1.
3 VBUS_OVERRI DE_EN	VBUS detect signal override enable This bit field allows SW to override the results from the VBUS_VALID and Session Valid comparators using the values in USB1_VBUS_DETECT[7:4]. The VBUS_VALID, AVALID, BVALID, and SESSEND signals sent to the USB controller are each affected by these bit selections. The values reported for AVALID, BVALID, and SESSEND in USB1_VBUS_DET_STAT[2:0] are also affected but the value reported for VBUS_VALID in USB1_VBUS_DET_STAT[3] is not affected. This override method may be useful if VBUS detection is not done with the internal VBUS_VALID or Session End comparators. 0b - Use the results of the internal VBUS_VALID and Session Valid comparators for VBUS_VALID, AVALID, BVALID, and SESSEND (Default) 1b - Use the override values for VBUS_VALID, AVALID, BVALID, and SESSEND
2-0 VBUSVALID_T HRESH	VBUS comparator threshold Sets the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hysteresis to minimize the need for software debounce of the detection. This comparator has ~50mV of hysteresis to prevent chattering at the comparator trip point. • 000 - 4.0 V • 001 - 4.1 V • 010 - 4.2 V • 011 - 4.3 V • 100 - 4.4 V (Default) • 101 - 4.5 V • 110 - 4.6 V • 111 - 4.7 V

57.4.1.33 VBUS detect Set (USB1_VBUS_DETECT_SET)

Offset

Register	Offset
USB1_VBUS_DETECT_ SET	C4h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN_C	Reserved			Reserv	DISCH	Reserved					PWRU	Reserv	VBUS	Reserved	
W	HAR...				ed	AR...						P_C...	ed	VAL...		
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	Reserv	Reserv	Reserv	Reserv	VBUS_SOURC	VBUS	VBUS	VBUS	AVALI	BVALI	SESS	VBUS_	VBUSVALID_THRESH		
W	ed	ed	ed	ed	ed	E_SEL	VAL...	VAL...	VAL...	D_...	D_...	END...	OV...			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31 EN_CHARGER _RESISTOR	Enables resistors used for an older method of resistive battery charger detection 0b - No effect 1b - Sets the corresponding bit
30-28 —	Reserved
27 —	Reserved
26 DISCHARGE_V BUS	Controls VBUS discharge resistor 0b - No effect 1b - Sets the corresponding bit
25-21 —	Reserved
20 PWRUP_CMPS	Enables the VBUS_VALID comparator 0b - No effect 1b - Sets the corresponding bit
19 —	Reserved
18 VBUSVALID_T O_SESSVALID	Selects the comparator used for VBUS_VALID 0b - No effect 1b - Sets the corresponding bit
17-15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10-9 VBUS_SOURCE_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller
8 VBUSVALID_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller 0b - No effect 1b - Sets the corresponding bit
7 VBUSVALID_OVERRIDE	Override value for VBUS_VALID signal sent to USB controller 0b - No effect 1b - Sets the corresponding bit
6 AVALID_OVERRIDE	Override value for A-Device Session Valid 0b - No effect 1b - Sets the corresponding bit
5 BVALID_OVERRIDE	Override value for B-Device Session Valid 0b - No effect 1b - Sets the corresponding bit
4 SESSEND_OVERRIDE	Override value for SESSEND 0b - No effect 1b - Sets the corresponding bit
3 VBUS_OVERRIDE_EN	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1'b0. 0b - No effect 1b - Sets the corresponding bit
2-0	VBUS comparator threshold

Table continues on the next page...

Table continued from the previous page...

Field	Function
VBUSVALID_T HRESH	000b - No effect 001b - Sets the corresponding bit

57.4.1.34 VBUS detect Clear (USB1_VBUS_DETECT_CLR)

Offset

Register	Offset
USB1_VBUS_DETECT_CLR	C8h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN_C	Reserved			Reserv	DISCH	Reserved				PWRU	Reserv	VBUS	Reserved		
W	HAR...				ed	AR...					P_C...	ed	VAL...			
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	Reserv	Reserv	Reserv	Reserv	VBUS_SOURC	VBUS	VBUS	AVALI	BVALI	SESS	VBUS_	VBUSVALID_THRESH			
W	ed	ed	ed	ed	ed	E_SEL	VAL...	VAL...	D_...	D_...	END...	OV...				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31 EN_CHARGER _RESISTOR	Enables resistors used for an older method of resistive battery charger detection 0b - No effect 1b - Sets the corresponding bit
30-28 —	Reserved
27 —	Reserved
26	Controls VBUS discharge resistor

Table continues on the next page...

Table continued from the previous page...

Field	Function
DISCHARGE_VBUS	0b - No effect 1b - clears the corresponding bit
25-21 —	Reserved
20 PWRUP_CMPS	Enables the VBUS_VALID comparator 0b - No effect 1b - clears the corresponding bit
19 —	Reserved
18 VBUSVALID_T O_SESSVALID	Selects the comparator used for VBUS_VALID 0b - No effect 1b - clears the corresponding bit
17-15 —	Reserved
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10-9 VBUS_SOURCE_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller
8 VBUSVALID_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller 0b - No effect 1b - clears the corresponding bit
7 VBUSVALID_OVERRIDE	Override value for VBUS_VALID signal sent to USB controller 0b - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1b - clears the corresponding bit
6 AVALID_OVERRIDE	Override value for A-Device Session Valid 0b - No effect 1b - clears the corresponding bit
5 BVALID_OVERRIDE	Override value for B-Device Session Valid 0b - No effect 1b - clears the corresponding bit
4 SESSEND_OVERRIDE	Override value for SESSEND 0b - No effect 1b - clears the corresponding bit
3 VBUS_OVERRIDE_EN	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1'b0. 0b - No effect 1b - clears the corresponding bit
2-0 VBUSVALID_THRESHOLD	VBUS comparator threshold 000b - No effect 001b - clears the corresponding bit

57.4.1.35 VBUS detect Toggle (USB1_VBUS_DETECT_TOG)

Offset

Register	Offset
USB1_VBUS_DETECT_TOG	CCh

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN_C	Reserved			Reserv	DISCH	Reserved				PWRU	Reserv	VBUS	Reserved		
W	HAR...				ed	AR...					P_C...	ed	VAL...			
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv	Reserv	Reserv	Reserv	Reserv	VBUS_SOURC	VBUS	VBUS	AVALI	BVALI	SESS	VBUS_	VBUSVALID_THRESH			
W	ed	ed	ed	ed	ed	E_SEL	VAL...	VAL...	D_...	D_...	END...	OV...				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Function
31 EN_CHARGER _RESISTOR	Enables resistors used for an older method of resistive battery charger detection 0b - No effect 1b - Toggles the corresponding bit
30-28 —	Reserved
27 —	Reserved
26 DISCHARGE_V BUS	Controls VBUS discharge resistor 0b - No effect 1b - Toggles the corresponding bit
25-21 —	Reserved
20 PWRUP_CMPS	Enables the VBUS_VALID comparator 0b - No effect 1b - Toggles the corresponding bit
19 —	Reserved
18 VBUSVALID_T O_SESSVALID	Selects the comparator used for VBUS_VALID 0b - No effect 1b - Toggles the corresponding bit
17-15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 —	Reserved
10-9 VBUS_SOURCE_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller
8 VBUSVALID_SEL	Selects the source of the VBUS_VALID signal reported to the USB controller 0b - No effect 1b - Toggles the corresponding bit
7 VBUSVALID_OVERRIDE	Override value for VBUS_VALID signal sent to USB controller 0b - No effect 1b - Toggles the corresponding bit
6 AVALID_OVERRIDE	Override value for A-Device Session Valid 0b - No effect 1b - Toggles the corresponding bit
5 BVALID_OVERRIDE	Override value for B-Device Session Valid 0b - No effect 1b - Toggles the corresponding bit
4 SESEND_OVERRIDE	Override value for SESSEND 0b - No effect 1b - Toggles the corresponding bit
3 VBUS_OVERRIDE_EN	VBUS detect signal override. This bit is used when EXT_VBUS_OVERRIDE_EN = 1'b0. 0b - No effect 1b - Toggles the corresponding bit
2-0	VBUS comparator threshold

Table continues on the next page...

Table continued from the previous page...

Field	Function
VBUSVALID_T HRESH	000b - No effect 001b - Toggles the corresponding bit

57.4.1.36 VBUS Detect Status (USB1_VBUS_DET_STAT)

Offset

Register	Offset
USB1_VBUS_DET_STAT	D0h

Function

This register allows observation of status for USB VBUS detect functions.

The values reported in this register are synchronized by the apb_clk.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											VBUS_ VA...	VBUS_ VA...	AVALI D	BVALI D	SESS END
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4 VBUS_VALID_3V	VBUS_VALID_3V detector status The VBUS_VALID_3V detector has a lower threshold for the voltage on the USB1_VBUS pin than either the Session Valid or VBUS_VALID comparators. This signal may be useful for applications using a non-standard VBUS voltage.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - VBUS voltage is below VBUS_VALID_3V threshold 1b - VBUS voltage is above VBUS_VALID_3V threshold
3 VBUS_VALID	VBUS voltage status This bit field shows the result of VBUS_VALID detection for the USB1_VBUS pin. The VBUS_VALID comparator used is selected by USB1_VBUS_DETECT[18]. The VBUS_VALID source is not affected by the values of USB1_VBUS_DETECT[7, 3] and cannot be overwritten by SW. 0b - VBUS is below the comparator threshold 1b - VBUS is above the comparator threshold
2 AVALID	A-Device Session Valid status A-Device Session Valid status, determined by the Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the VBUS_DETECT[6, 3] bit fields. 0b - The VBUS voltage is below the Session Valid threshold 1b - The VBUS voltage is above the Session Valid threshold
1 BVALID	B-Device Session Valid status B-Device Session Valid status, determined by the Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the VBUS_DETECT[5, 3] bit fields. 0b - The VBUS voltage is below the Session Valid threshold 1b - The VBUS voltage is above the Session Valid threshold
0 SESSEND	Session End indicator Session End status, value inverted from Session Valid comparator. The default value of this bit is determined by the voltage on the USB1_VBUS pin. This bit can be overwritten by SW using the VBUS_DETECT[4, 3] bit fields. 0b - The VBUS voltage is above the Session Valid threshold 1b - The VBUS voltage is below the Session Valid threshold

57.4.1.37 Charger Detect Control (USB1_CHRG_DETECT)

Offset

Register	Offset
USB1_CHRG_DETECT	E0h

Function

It defines controls for USB Battery Charging detection functions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserved								BGR_I	Reserved		Reserv	Reserv	Reserved	
W	ed									Bl...			ed	ed		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												PULLU	Reserv	Reserv	
W													P...	ed	ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-24 —	Reserved
23 BGR_IBIAS	USB charge detector bias current reference This bit determines the reference for the bias current of the USB charge detector. This bit should always be set to 1. 0b - Bias current is derived from the USB PHY internal current generator. 1b - Bias current is derived from the reference generator of the bandgap.
22-21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17-3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 PULLUP_DP	PULLUP_DP This bit is used to pull up DP, for digital charge detect.
1 —	Reserved
0 —	Reserved. The bit field must remain at value 1'b0 to avoid interfering with USB Battery Charging detection.

57.4.1.38 Charger Detect Control Set (USB1_CHRG_DETECT_SET)

Offset

Register	Offset
USB1_CHRG_DETECT_SET	E4h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the VBUS_DET_STAT register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserved								BGR_I	Reserved		Reserv	Reserv	Reserv	Reserved
W	ed									Bl...			ed	ed	ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												Reserv	PULLU	Reserv	Reserv
W													ed	P_...	ed	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
23 BGR_IBIAS	BGR_IBIAS 0b - No effect 1b - Sets the corresponding bit
22-21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17-4 —	Reserved
3 —	Reserved
2 PULLUP_DP	PULLUP_DP 0b - No effect 1b - Sets the corresponding bit
1 —	Reserved
0 —	Reserved. The bit field must remain at value 1'b0 to avoid interfering with USB Battery Charging detection.

57.4.1.39 Charger Detect Control Clear (USB1_CHRG_DETECT_CLR)

Offset

Register	Offset
USB1_CHRG_DETECT_CLR	E8h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the VBUS_DET_STAT register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserved								BGR_I	Reserved		Reserv	Reserv	Reserved	
W	ed									Bl...			ed	ed		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												Reserv	PULLU	Reserv	Reserv
W													ed	P_...	ed	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-24 —	Reserved
23 BGR_IBIAS	BGR_IBIAS 0b - No effect 1b - Clears the corresponding bit
22-21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17-4 —	Reserved
3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 PULLUP_DP	PULLUP_DP 0b - No effect 1b - Clears the corresponding bit
1 —	Reserved
0 —	Reserved. The bit field must remain at value 1'b0 to avoid interfering with USB Battery Charging detection.

57.4.1.40 Charger Detect Control Toggle (USB1_CHRG_DETECT_TOG)

Offset

Register	Offset
USB1_CHRG_DETECT_TOG	ECh

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the VBUS_DET_STAT register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserved								BGR_I	Reserved		Reserv	Reserv	Reserv	Reserved
W	ed									Bl...			ed	ed	ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												Reserv	PULLU	Reserv	Reserv
W													ed	P_...	ed	ed
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
30-24 —	Reserved
23 BGR_IBIAS	BGR_IBIAS 0b - No effect 1b - Toggles the corresponding bit
22-21 —	Reserved
20 —	Reserved
19 —	Reserved
18 —	Reserved
17-4 —	Reserved
3 —	Reserved
2 PULLUP_DP	PULLUP_DP 0b - No effect 1b - Toggles the corresponding bit
1 —	Reserved
0 —	Reserved. The bit field must remain at value 1'b0 to avoid interfering with USB Battery Charging detection.

57.4.1.41 Charge Detect Status (USB1_CHRG_DET_STAT)

Offset

Register	Offset
USB1_CHRG_DET_STAT	F0h

Function

This register allows observation of status for USB Charge detect functions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SECD ET_...	DP_ST ATE	DM_S TATE	CHRG _DE...	PLUG_ CO...			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-5 —	Reserved
4 SECDDET_DCP	Battery Charging Secondary Detection phase output During the USB Battery Charging Secondary Detection phase using the USBHSDCD module, this bit field indicates which kind of Charging Port was detected. 0b - Charging Downstream Port (CDP) has been detected 1b - Downstream Charging Port (DCP) has been detected
3 DP_STATE	DP_STATE Single ended receiver output for the DP pin, from charger detection circuits. 0b - DP pin voltage is < 0.8V 1b - DP pin voltage is > 2.0V
2 DM_STATE	DM_STATE Single ended receiver output for the DM pin, from charger detection circuits. 0b - DM pin voltage is < 0.8V 1b - DM pin voltage is > 2.0V
1 CHRG_DETECTED	Battery Charging Primary Detection phase output During the USB Battery Charging Primary Detection phase using the USBHSDCD module, this bit field indicates whether a Standard Downstream Port or Charging Port was detected. 0b - Standard Downstream Port (SDP) has been detected 1b - Charging Port has been detected

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 PLUG_CONTACT	Battery Charging Data Contact Detection phase output During the Data Contact Detection phase per the <i>USB Battery Charging Specification Revision 1.2</i> using the USBHSDCD module, this bit field indicates whether a USB cable has been attached between the remote host and the local device. 0b - No USB cable attachment has been detected 1b - A USB cable attachment between the device and host has been detected

57.4.1.42 Analog Control (ANACTRL)

Offset

Register	Offset
ANACTRL	100h

Function

This register allows an additional type of control of 15 kohm pulldown resistors on both USB_DP and USB_DM pins.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	Reserved			Reserved											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				DEV_P UL...	Reserved	Reserved				Reserved		Reserved	Reserved		
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-28 —	Reserved
27-11	The value of this these bit fields must remain at 1'b0 for normal operation.

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
10 DEV_PULLDOWN	Device Pull-down Setting this field to 1'b1 will enable the 15kΩ pulldown resistors on both USB_DP and USB_DM pins. This feature can be used in device mode while the USB cable is disconnected to keep the data pins at known values, avoiding unnecessary interrupts from the single ended receivers. This bit must be reset to 1'b0 during normal USB data communication in device mode, or while battery charger detection using the USBHSDCD module is used. • 0 - The 15kΩ nominal pulldowns on the USB_DP and USB_DM pins are disabled in device mode. • 1 - The 15kΩ nominal pulldowns on the USB_DP and USB_DM pins are enabled in device mode.
9 —	Reserved
8-4 —	Reserved
3-2 —	The value of this bit field must remain at 2'b00 for normal operation.
1 —	Reserved
0 —	Reserved. The value of this bit field must remain at 1'b0 for normal operation.

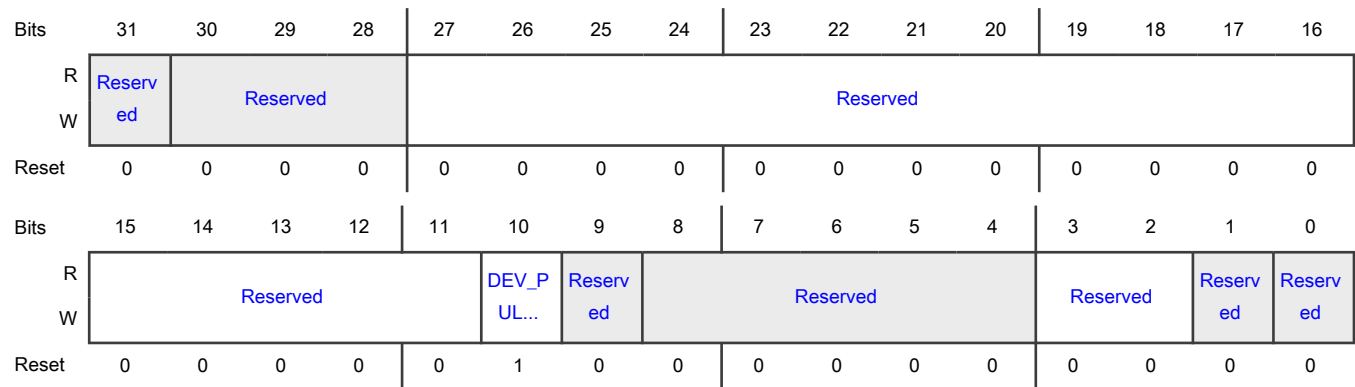
57.4.1.43 Analog Control Set (ANACTRL_SET)

Offset

Register	Offset
ANACTRL_SET	104h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the ANACTRL register.

Diagram**Fields**

Field	Function
31 —	Reserved
30-28 —	Reserved
27-11 —	Reserved
10 DEV_PULLDOWN	Device Pull-down 0b - No effect 1b - Sets the corresponding bit
9 —	Reserved
8-4 —	Reserved
3-2 —	Reserved
1 —	Reserved
0 —	Reserved. The value of this bit field must remain at 1'b0 for normal operation.

57.4.1.44 Analog Control Clear (ANACTRL_CLR)

Offset

Register	Offset
ANACTRL_CLR	108h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the ANACTRL register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv ed	Reserved			Reserved											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				DEV_P UL...	Reserv ed	Reserved				Reserved	Reserv ed	Reserv ed			
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-28 —	Reserved
27-11 —	Reserved
10 DEV_PULLDOWN	Device Pull-down 0b - No effect 1b - Clears the corresponding bit
9 —	Reserved
8-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 —	Reserved
1 —	Reserved
0 —	Reserved. The value of this bit field must remain at 1'b0 for normal operation.

57.4.1.45 Analog Control Toggle (ANACTRL_TOG)

Offset

Register	Offset
ANACTRL_TOG	10Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the ANACTRL register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserv	Reserved			Reserved											
W	ed															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				DEV_P	UL...	Reserv	ed	Reserved		Reserved		Reserved	Reserv	ed	Reserv
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30-28 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
27-11 —	Reserved
10 DEV_PULLDOWN	Device Pull-down 0b - No effect 1b - Toggles the corresponding bit
9 —	Reserved
8-7 —	Reserved
6-4 —	Reserved
3-2 —	Reserved
1 —	Reserved
0 —	Reserved. The value of this bit field must remain at 1'b0 for normal operation.

57.4.1.46 USB PHY Loopback Control/Status (USB1_LOOPBACK)

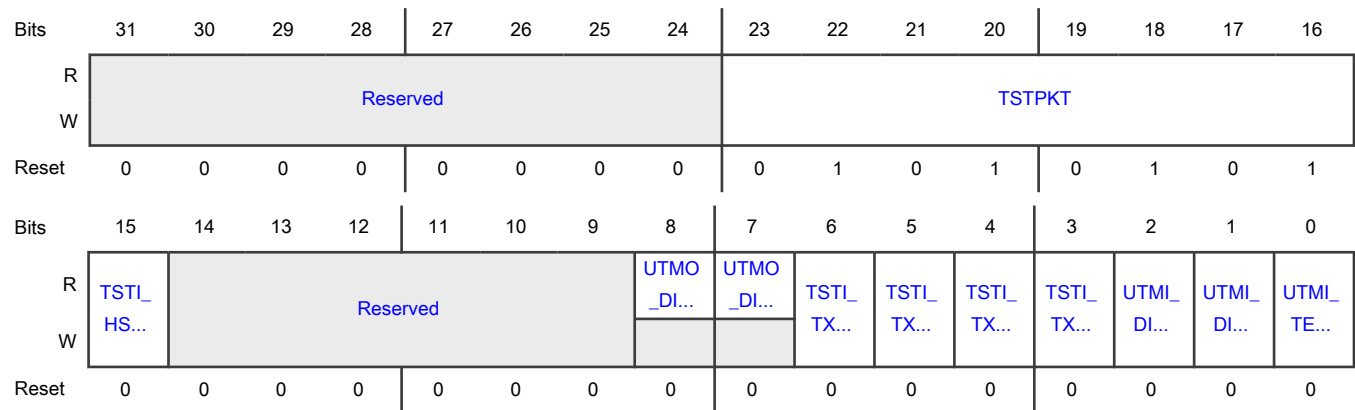
Offset

Register	Offset
USB1_LOOPBACK	110h

Function

This register controls loopback testing of the USB PHY. Loopback mode is for test purposes only. It cannot be used during normal USB data communication.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 TSTPKT	Test packet Selects the packet data byte used for USB loopback testing in Pulse mode. Pulse mode is selected by setting USB1_LOOPBACK[2:1] to a value of 2'b01. Default value produces a data inversion at each unit interval.
15 TSTL_HSFS_M ODE_EN	Setting this bit field to value 1'b1 will enable the loopback test to dynamically change the packet speed. It will send a number of High Speed packets determined by USB1_LOOPBACK_HSFSCNT[TSTL_HS_NUMBER], followed by a number of Full Speed packets determined by USB1_LOOPBACK_HSFSCNT[TSTL_FS_NUMBER].
14-9 —	Reserved
8 UTMO_DIG_TS T1	Status bit for USB loopback test. Value = 1'b1 indicates a passing result. Test results are only valid while UTMI_TESTSTART is set to value 1'b1.
7 UTMO_DIG_TS T0	Status bit for USB loopback test. Value = 1'b0 indicates a passing result. Test results are only valid while UTMI_TESTSTART is set to value 1'b1.
6 TSTL_TX_HIZ	Sets TX Hi-Z for USB loopback test.
5 TSTL_TX_EN	Enable TX for USB loopback test.
4	Select HS or FS mode for USB loopback testing.

Table continues on the next page...

Table continued from the previous page...

Field	Function
TSTI_TX_LS_MODE	Set to value 1'b1 to choose LS for USB loopback testing, set to value 1'b0 to choose HS or FS mode which is defined by TSTI_TX_HS_MODE.
3 TSTI_TX_HS_MODE	Select HS or FS mode for USB loopback testing. Set to value 1'b1 to choose HS for USB loopback testing, set to value 1'b0 to choose FS mode.
2 UTMI_DIG_TST1	Mode control for USB loopback test. Setting this bit to a value of 1'b1 while UTMI_DIG_TST0 remains at a value of 1'b0 selects Psuedorandom modes for the loopback test.
1 UTMI_DIG_TST0	Mode control for USB loopback test. Setting this bit to a value of 1'b1 while UTMI_DIG_TST1 remains at a value of 1'b0 selects Pulse mode for the loopback test.
0 UTMI_TESTSTART	USB loopback test. This bit enables the USB loopback test.

57.4.1.47 USB PHY Loopback Control/Status Set (USB1_LOOPBACK_SET)

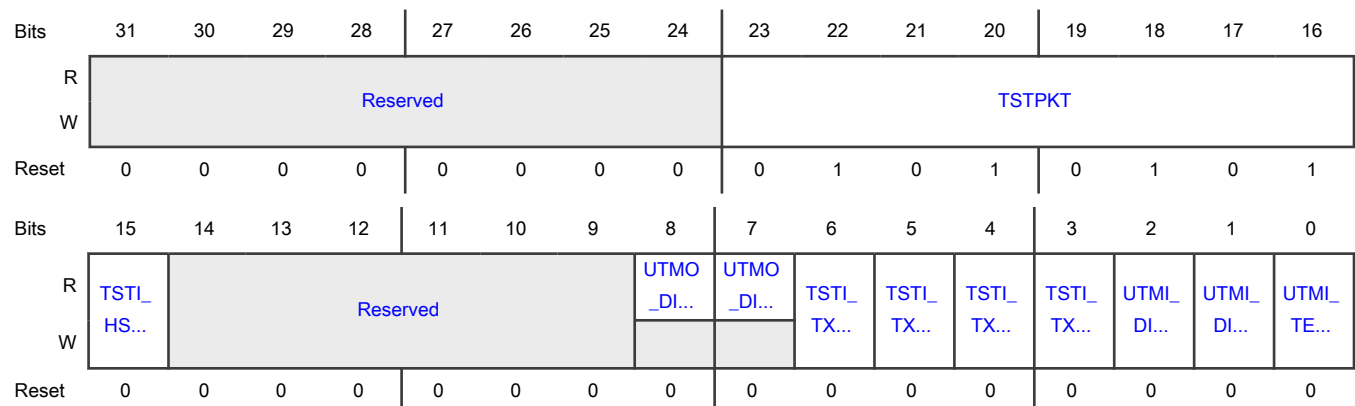
Offset

Register	Offset
USB1_LOOPBACK_SET	114h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the LOOPBACK_SET register. Only zeros should be written to reserved bits.

Diagram



Fields

Field	Function
31-24 —	Reserved
23-16 TSTPKT	Test packet 0000_0000b - No effect 0000_0001b - Sets the corresponding bit
15 TSTI_HSFS_M ODE_EN	TSTI_HSFS_MODE_EN 0b - No effect 1b - Sets the corresponding bit
14-9 —	Reserved
8 UTMO_DIG_TS T1	Status bit for USB loopback test. 0b - No effect 1b - Sets the corresponding bit
7 UTMO_DIG_TS T0	Status bit for USB loopback test. 0b - No effect 1b - Sets the corresponding bit
6 TSTI_TX_HIZ	Sets TX Hi-Z for USB loopback test. 0b - No effect 1b - Sets the corresponding bit
5 TSTI_TX_EN	Enable TX for USB loopback test. 0b - No effect 1b - Sets the corresponding bit
4 TSTI_TX_LS_M ODE	Select HS or FS mode for USB loopback testing. 0b - No effect 1b - Sets the corresponding bit
3 TSTI_TX_HS_M ODE	Select HS or FS mode for USB loopback testing.
2 UTMI_DIG_TST 1	Mode control for USB loopback test. 0b - No effect 1b - Sets the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 UTMI_DIG_TST 0	Mode control for USB loopback test. 0b - No effect 1b - Sets the corresponding bit
0 UTMI_TESTST ART	USB loopback test. 0b - No effect 1b - Sets the corresponding bit

57.4.1.48 USB PHY Loopback Control/Status Clear (USB1_LOOPBACK_CLR)

Offset

Register	Offset
USB1_LOOPBACK_CLR	118h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the LOOPBACK_SET register. Only zeros should be written to reserved bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TSTPKT							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSTL_	Reserved						UTMO_	UTMO_	TSTL_	TSTL_	TSTL_	TSTL_	UTMI_	UTMI_	UTMI_
W	HS...							_DI...	_DI...	TX...	TX...	TX...	TX...	DI...	DI...	TE...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-16	Test packet

Table continues on the next page...

Table continued from the previous page...

Field	Function
TSTPKT	0000_0000b - No effect 0000_0001b - Clears the corresponding bit
15 TSTI_HSFS_M ODE_EN	TSTI_HSFS_MODE_EN 0b - No effect 1b - Clears the corresponding bit
14-9 —	Reserved
8 UTMO_DIG_TS T1	Status bit for USB loopback test. 0b - No effect 1b - Clears the corresponding bit
7 UTMO_DIG_TS T0	Status bit for USB loopback test. 0b - No effect 1b - Clears the corresponding bit
6 TSTI_TX_HIZ	Sets TX Hi-Z for USB loopback test. 0b - No effect 1b - Clears the corresponding bit
5 TSTI_TX_EN	Enable TX for USB loopback test. 0b - No effect 1b - Clears the corresponding bit
4 TSTI_TX_LS_M ODE	Select HS or FS mode for USB loopback testing. 0b - No effect 1b - Clears the corresponding bit
3 TSTI_TX_HS_M ODE	Select HS or FS mode for USB loopback testing.
2 UTMI_DIG_TST 1	Mode control for USB loopback test. 0b - No effect 1b - Clears the corresponding bit
1 UTMI_DIG_TST 0	Mode control for USB loopback test. 0b - No effect 1b - Clears the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
0	USB loopback test.
UTMI_TESTST ART	0b - No effect 1b - Clears the corresponding bit

57.4.1.49 USB PHY Loopback Control/Status Toggle (USB1_LOOPBACK_TOG)

Offset

Register	Offset
USB1_LOOPBACK_TO G	11Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the LOOPBACK_SET register. Only zeros should be written to reserved bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TSTPKT							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSTI_ HS...	Reserved						UTMO _DI...	UTMO _DI...	TSTI_ TX...	TSTI_ TX...	TSTI_ TX...	TSTI_ TX...	UTMI_ DI...	UTMI_ DI...	UTMI_ TE...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-24 —	Reserved
23-16 TSTPKT	Test packet 0000_0000b - No effect 0000_0001b - Toggles the corresponding bit
15	TSTI_HSFS_MODE_EN

Table continues on the next page...

Table continued from the previous page...

Field	Function
TSTI_HSFS_M ODE_EN	0b - No effect 1b - Toggles the corresponding bit
14-9 —	Reserved
8 UTMO_DIG_TS T1	Status bit for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
7 UTMO_DIG_TS T0	Status bit for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
6 TSTI_TX_HIZ	Sets TX Hi-Z for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
5 TSTI_TX_EN	Enable TX for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
4 TSTI_TX_LS_M ODE	Select HS or FS mode for USB loopback testing. 0b - No effect 1b - Toggles the corresponding bit
3 TSTI_TX_HS_M ODE	Select HS or FS mode for USB loopback testing.
2 UTMI_DIG_TST 1	Mode control for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
1 UTMI_DIG_TST 0	Mode control for USB loopback test. 0b - No effect 1b - Toggles the corresponding bit
0 UTMI_TESTST ART	USB loopback test. 0b - No effect 1b - Toggles the corresponding bit

57.4.1.50 Loopback Packet Number Select (USB1_LOOPBACK_HSFSCNT)

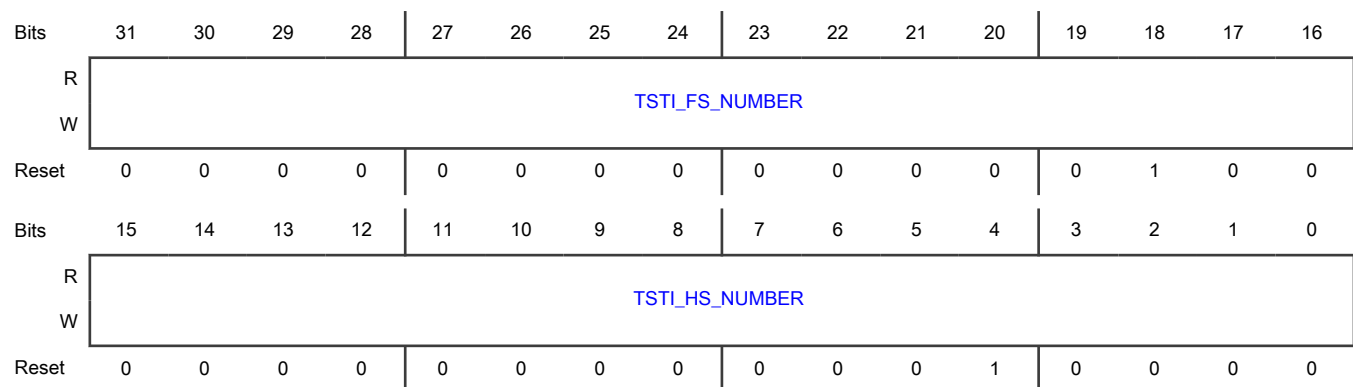
Offset

Register	Offset
USB1_LOOPBACK_HSFSCNT	120h

Function

It provides the number of packets in dynamic loopback test based on the mode.

Diagram



Fields

Field	Function
31-16 TSTI_FS_NUMBER	USB loopback test FS CNT. Full speed packet number, used when USB1_LOOPBACK[TSTI_HSFS_MODE_EN] is set to value 1'b1.
15-0 TSTI_HS_NUMBER	USB loopback test HS CNT. High speed packet number, used when USB1_LOOPBACK[TSTI_HSFS_MODE_EN] is set to value 1'b1.

57.4.1.51 USB PHY Loopback Packet Number Select Set (USB1_LOOPBACK_HSFSCNT_SET)

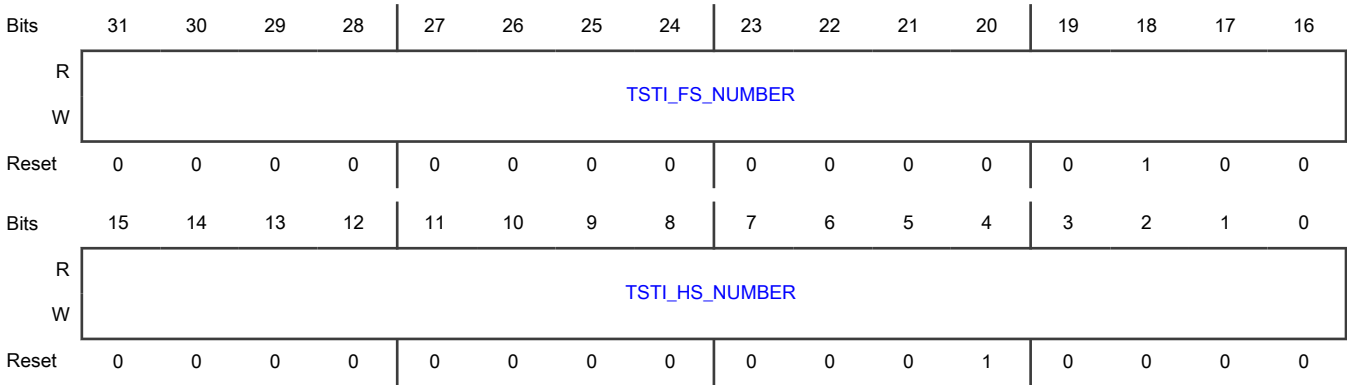
Offset

Register	Offset
USB1_LOOPBACK_HSFSCNT_SET	124h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the LOOPBACK_HSFSCNT register.

Diagram



Fields

Field	Function
31-16 TSTI_FS_NUMBER	USB loopback test FS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Sets the corresponding bit
15-0 TSTI_HS_NUMBER	USB loopback test HS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Sets the corresponding bit

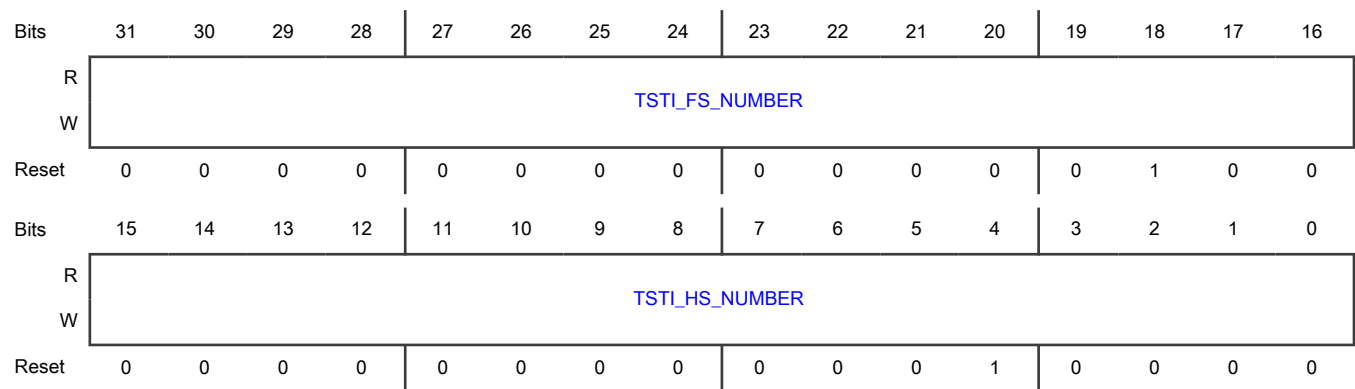
57.4.1.52 USB PHY Loopback Packet Number Select Clear (USB1_LOOPBACK_HSFSCNT_CLR)

Offset

Register	Offset
USB1_LOOPBACK_HSFSCNT_CLR	128h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the LOOPBACK_HSFSCNT register.

Diagram**Fields**

Field	Function
31-16 TSTI_FS_NUMBER	USB loopback test FS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Clears the corresponding bit
15-0 TSTI_HS_NUMBER	USB loopback test HS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Clears the corresponding bit

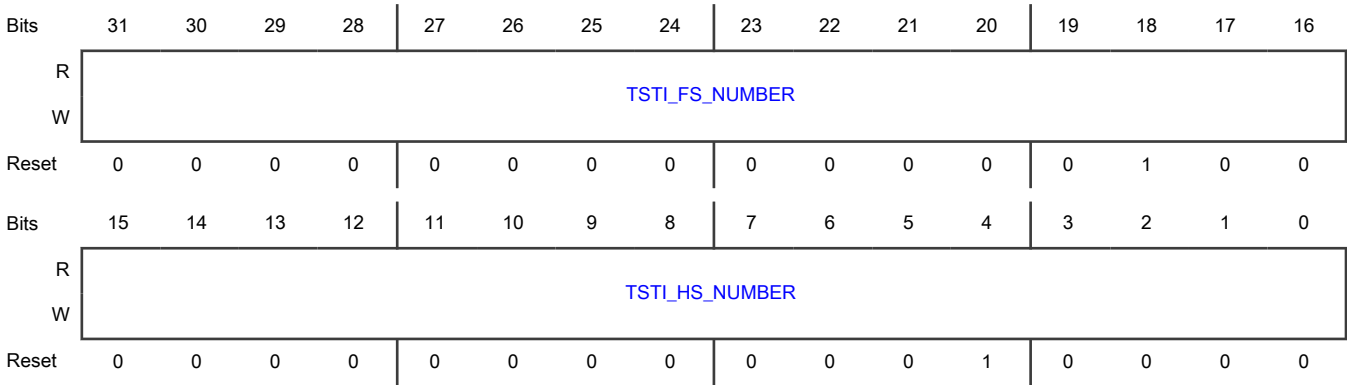
57.4.1.53 USB PHY Loopback Packet Number Select Toggle (USB1_LOOPBACK_HSFSCNT_TOG)**Offset**

Register	Offset
USB1_LOOPBACK_HSFSCNT_TOG	12Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the LOOPBACK_HSFSCNT register.

Diagram



Fields

Field	Function
31-16 TSTI_FS_NUMBER	USB loopback test FS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Toggles the corresponding bit
15-0 TSTI_HS_NUMBER	USB loopback test HS CNT. 0000_0000_0000_0000b - No effect 0000_0000_0000_0001b - Toggles the corresponding bit

57.4.1.54 Trim Override Enable (TRIM_OVERRIDE_EN)

Offset

Register	Offset
TRIM_OVERRIDE_EN	130h

Function

The bit fields in this register allow observation of the default OTP setting of the Phy parameter for PLL_DIV_SEL. Additional bit fields also determine whether those values can be overridden by settings in the TX, DEBUG1, and PLL_SIC registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBPHY_TX_CAL45DN				USBPHY_TX_CAL45DP				USBPHY_TX_D_CAL				USB_REG_ENV_TAI...		PLL_CTRL0_DIV_S...	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PLL_C TR...	USB2_REFBIA S_TST	USB2_REFBIAS_VBGA DJ		Reserved				REFBI AS...	REFBI AS...	TX_CA L4...	TX_CA L4...	TX_D_ CA...	ENV_T AI...	DIV_S EL...	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-28 USBPHY_TX_C AL45DN	Default value of TX_CAL45DM.
27-24 USBPHY_TX_C AL45DP	Default value of TX_CAL45DP.
23-20 USBPHY_TX_D _CAL	Default value of TX_D_CAL.
19-18 USB_REG_EN V_TAIL_ADJ_V D	Default value of ENV_TAIL_ADJ.
17-15 PLL_CTRL0_DI V_SEL	Default value of PLL_DIV_SEL.
14-13 USB2_REFBIA S_TST	Bias current control for usb2_phy and usb_PLL
12-10 USB2_REFBIA S_VBGADJ	Adjustment bits for bandgap
9-7	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
6 REFBIAS_TST_OVERRIDE	Override enable for bias current control. When this field is set, the register value in DEBUG1[22:21] will be used.
5 REFBIAS_VBG_ADJ_OVERRIDE	Override enable for bandgap adjustment. When this field is set, the register value in DEBUG1[20:18] will be used.
4 TX_CAL45DM_OVERRIDE	TX_CAL45DM_OVERRIDE Override enable for TX_CAL45DM, when set, the register value in TX[11:8] will be used.
3 TX_CAL45DP_OVERRIDE	TX_CAL45DP_OVERRIDE Override enable for TX_CAL45DP, when set, the register value in TX[19:16] will be used.
2 TX_D_CAL_OVERRIDE	TX_D_CAL_OVERRIDE Override enable for TX_D_CAL, when set, the register value in TX[3:0] will be used.
1 ENV_TAIL_ADJ_VD_OVERRIDE	ENV_TAIL_ADJ_VD_OVERRIDE Override enable for ENV_TAIL_ADJ, when set, the register value in DEBUG1[14:13] will be used.
0 DIV_SEL_OVERRIDE	DIV_SEL_OVERRIDE Override enable for PLL_DIV_SEL, when set, the register value in PLL_SIC[1:0] will be used.

57.4.1.55 Trim Set (TRIM_OVERRIDE_EN_SET)

Offset

Register	Offset
TRIM_OVERRIDE_EN_SET	134h

Function

Writing a 1 to a bit position in this register sets the corresponding position in the TRIM0 register. Only zeros should be written to reserved bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBPHY_TX_CAL45DN				USBPHY_TX_CAL45DP				USBPHY_TX_D_CAL				USB_REG_ENV_TAI...		PLL_CTRL0_DIV_S...	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PLL_CTRL0_DIV_SEL...	USB2_REFBIAS_TST	USB2_REFBIAS_VBGA_ADJ		Reserved				REFBIAS...	REFBIAS...	TX_CAL4...	TX_CAL4...	TX_DIV_SEL...	ENV_TAIL_ADJ...	DIV_SEL...	
W									REFBIAS...	REFBIAS...	TX_CAL4...	TX_CAL4...	TX_DIV_SEL...	ENV_TAIL_ADJ...	DIV_SEL...	
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-28 USBPHY_TX_CAL45DN	Default value of TX_CAL45DM. 0000b - No effect 0001b - Sets the corresponding bit
27-24 USBPHY_TX_CAL45DP	Default value of TX_CAL45DP. 0000b - No effect 0001b - Sets the corresponding bit
23-20 USBPHY_TX_D_CAL	Default value of TX_D_CAL. 0000b - No effect 0001b - Sets the corresponding bit
19-18 USB_REG_ENV_TAIL_ADJ_V_D	Default value of ENV_TAIL_ADJ. 00b - No effect 01b - Sets the corresponding bit
17-15 PLL_CTRL0_DIV_SEL	Default value of PLL_DIV_SEL. 000b - No effect 001b - Sets the corresponding bit
14-13 USB2_REFBIAS_TST	Bias current control for usb2_phy and usb_PLL 00b - No effect 01b - Sets the corresponding bit
12-10 USB2_REFBIAS_VBGADJ	Adjustment bits for bandgap 000b - No effect 001b - Sets the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-7 —	Reserved
6 REFBIAS_TST_ OVERRIDE	Override enable for bias current control. 0b - No effect 1b - Sets the corresponding bit
5 REFBIAS_VBG ADJ_OVERRID E	Override enable for bandgap adjustment. 0b - No effect 1b - Sets the corresponding bit
4 TX_CAL45DM_ OVERRIDE	TX_CAL45DM_OVERRIDE 0b - No effect 1b - Sets the corresponding bit
3 TX_CAL45DP_ OVERRIDE	TX_CAL45DP_OVERRIDE 0b - No effect 1b - Sets the corresponding bit
2 TX_D_CAL_OV ERRIDE	TX_D_CAL_OVERRIDE 0b - No effect 1b - Sets the corresponding bit
1 ENV_TAIL_ADJ _VD_OVERRID E	ENV_TAIL_ADJ_VD_OVERRIDE 0b - No effect 1b - Sets the corresponding bit
0 DIV_SEL_OVE RRIDE	DIV_SEL_OVERRIDE 0b - No effect 1b - Sets the corresponding bit

57.4.1.56 Trim Clear (TRIM_OVERRIDE_EN_CLR)

Offset

Register	Offset
TRIM_OVERRIDE_EN_ CLR	138h

Function

Writing a 1 to a bit position in this register clears the corresponding position in the TRIM0 register. Only zeros should be written to reserved bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBPHY_TX_CAL45DN				USBPHY_TX_CAL45DP				USBPHY_TX_D_CAL				USB_REG_ENV_TAI...		PLL_CTRL0_DIV_S...	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PLL_C TR...	USB2_REFBIA S_TST	USB2_REFBIAS_VBGA DJ		Reserved				REFBI AS...	REFBI AS...	TX_CA L4...	TX_CA L4...	TX_D_ CA...	ENV_T AI...	DIV_S EL...	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-28 USBPHY_TX_CAL45DN	Default value of TX_CAL45DM. 0000b - No effect 0001b - Clears the corresponding bit
27-24 USBPHY_TX_CAL45DP	Default value of TX_CAL45DP. 0000b - No effect 0001b - Clears the corresponding bit
23-20 USBPHY_TX_D_CAL	Default value of TX_D_CAL. 0000b - No effect 0001b - Clears the corresponding bit
19-18 USB_REG_ENV_TAIL_ADJ_V D	Default value of ENV_TAIL_ADJ. 00b - No effect 01b - Clears the corresponding bit
17-15 PLL_CTRL0_DIV_SEL	Default value of PLL_DIV_SEL. 000b - No effect 001b - Clears the corresponding bit
14-13 USB2_REFBIAS_TST	Bias current control for usb2_phy and usb_PLL 00b - No effect 01b - Clears the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
12-10 USB2_REFBIAS_VBGADJ	Adjustment bits for bandgap 000b - No effect 001b - Clears the corresponding bit
9-7 —	Reserved
6 REFBIAS_TST_OVERRIDE	Override enable for bias current control. 0b - No effect 1b - Clears the corresponding bit
5 REFBIAS_VBGADJ_OVERRIDE	Override enable for bandgap adjustment. 0b - No effect 1b - Clears the corresponding bit
4 TX_CAL45DM_OVERRIDE	TX_CAL45DM_OVERRIDE 0b - No effect 1b - Clears the corresponding bit
3 TX_CAL45DP_OVERRIDE	TX_CAL45DP_OVERRIDE 0b - No effect 1b - Clears the corresponding bit
2 TX_D_CAL_OVERRIDE	TX_D_CAL_OVERRIDE 0b - No effect 1b - Clears the corresponding bit
1 ENV_TAIL_ADJ_VD_OVERRIDE	ENV_TAIL_ADJ_VD_OVERRIDE 0b - No effect 1b - Clears the corresponding bit
0 DIV_SEL_OVERRIDE	DIV_SEL_OVERRIDE 0b - No effect 1b - Clears the corresponding bit

57.4.1.57 Trim Toggle (TRIM_OVERRIDE_EN_TOG)

Offset

Register	Offset
TRIM_OVERRIDE_EN_TOG	13Ch

Function

Writing a 1 to a bit position in this register toggles the corresponding position in the TRIM0 register. Only zeros should be written to reserved bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBPHY_TX_CAL45DN				USBPHY_TX_CAL45DP				USBPHY_TX_D_CAL				USB_REG_ENV_TAI...		PLL_CTRL0_DIV_S...	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PLL_C TR...	USB2_REFBIA S_TST		USB2_REFBIAS_VBGA DJ		Reserved				REFBI AS...	REFBI AS...	TX_CA L4...	TX_CA L4...	TX_D_ CA...	ENV_T AI...	DIV_S EL...
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Fields

Field	Function
31-28 USBPHY_TX_CAL45DN	Default value of TX_CAL45DM. 0000b - No effect 0001b - Toggles the corresponding bit
27-24 USBPHY_TX_CAL45DP	Default value of TX_CAL45DP. 0000b - No effect 0001b - Toggles the corresponding bit
23-20 USBPHY_TX_D_CAL	Default value of TX_D_CAL. 0000b - No effect 0001b - Toggles the corresponding bit
19-18 USB_REG_ENV_TAIL_ADJ_V D	Default value of ENV_TAIL_ADJ. 00b - No effect 01b - Toggles the corresponding bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-15 PLL_CTRL0_DIV_SEL	Default value of PLL_DIV_SEL. 000b - No effect 001b - Toggles the corresponding bit
14-13 USB2_REFBIAS_TST	Bias current control for usb2_phy and usb_PLL 00b - No effect 01b - Toggles the corresponding bit
12-10 USB2_REFBIAS_VBGADJ	Adjustment bits for bandgap 000b - No effect 001b - Toggles the corresponding bit
9-7 —	Reserved
6 REFBIAS_TST_OVERRIDE	Override enable for bias current control. 0b - No effect 1b - Toggles the corresponding bit
5 REFBIAS_VBG_ADJ_OVERRIDE	Override enable for bandgap adjustment. 0b - No effect 1b - Toggles the corresponding bit
4 TX_CAL45DM_OVERRIDE	TX_CAL45DM_OVERRIDE 0b - No effect 1b - Toggles the corresponding bit
3 TX_CAL45DP_OVERRIDE	TX_CAL45DP_OVERRIDE 0b - No effect 1b - Toggles the corresponding bit
2 TX_D_CAL_OVERRIDE	TX_D_CAL_OVERRIDE 0b - No effect 1b - Toggles the corresponding bit
1 ENV_TAIL_ADJ_VD_OVERRIDE	ENV_TAIL_ADJ_VD_OVERRIDE 0b - No effect 1b - Toggles the corresponding bit
0	DIV_SEL_OVERRIDE

Table continues on the next page...

Table continued from the previous page...

Field	Function
DIV_SEL_OVE RRIDE	0b - No effect 1b - Toggles the corresponding bit

Chapter 58

Analog-to-Digital Converter (ADC)

58.1 Chip-specific ADC information

Table 589. Reference links to related information

Topic	Related module	Reference
Full description	ADC	ADC
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

A 12-bit SAR ADC with up to 16 channels and 1 uSec conversion time will be used in conjunction with the associated ADC Controller. A temperature sensor will also be provided, included with the ADC. This is accessible only by the M33 CPU and both DMA controllers. The ADC will support both single-ended and differential operation. Up to sixteen ADC input pins will be provided which can either be used to drive 16 single-ended channels or 8 differential channels. RT500 provides two built-in temperature sensors, one from PMC and the other from ADC. The temperature sensors are connected internally to the ADC channel 7. Select the desired temp sensor through the TEMPSSENSORCTL register (See SYSCTL).

58.1.1 Module instances

This device has one instance of the ADC module, ADC0.

58.1.2 Chip-specific ADC Signal Descriptions

The ADC module supports analog channel inputs with differential and single-ended conversion options for all channels. The module requires specific supply, ground, and reference connections.

Table 590. Chip-specific ADC signal descriptions

Signal	Description
VDDA_ADC1V8	1.8 V analog supply voltage for ADC and comparator.
VDDA_BIAS	Bias voltage rail for maximum input voltage of ADC channels and analog comparator. VDDA_BIAS should be connected to the same supply source as used by the GPIO used for analog channel inputs. For chips that support multiple I/O supply levels through implementation of pad segments, then the supply for VDDA_BIAS should be the highest of the I/O supply levels associated with ADC analog channels used in the application.
VREFP	ADC positive reference voltage.
VREFN	ADC negative reference voltage.

Table 591. ADC configuration register

7:6 REFSEL	Description
0	Default setting. Uses the VREFP device pin as the reference.
1	Uses the VDDA_ADC1V8 device pin as the reference.
2	Uses the VDDA_ADC1V8 device pin as the reference.
3	Reserved

Table 592. ADC channel mapping

Port Pin	ADC Single Ended channel ¹	Differential channel pair
P0_5	CH0A	CH0P
P0_6	CH0B	CH0N
P0_12	CH1A	CH1P
P0_13	CH1B	CH1N
P0_19	CH2A	
P4_18	CH6A	

1. The CMDLa[ADCH], CMDLa[DIFF] and CMDLa[ABSEL] bitfields control selection of paired or individual input channels. Each ADC command independently makes a channel selection. Each ADCH channel selection has an associated A side and an associated B side input.

Each ADCH pair can optionally be converted in a differential mode but only limited pairs are intended to be converted as differential channels that is adjacent pins that have been designed with matched impedance.

Table 593. ADC channels for Analog Signals

Analog Signal	ADC Channels
Temp Sensor (+)	CH7A
Temp Sensor (-)	CH7B

58.1.3 Hardware Triggers

Hardware triggers are available from a number of sources, including pin interrupts, timer match outputs, the comparator output, the ARM CPU TX event, and the GPIO pin interrupt function.

Table 594. Hardware trigger sources

Trigger	Trigger source
0	GPIO_INT0_IRQ0
1	GPIO_INT0_IRQ1
2	SCT0_OUT4
3	SCT0_OUT5
4	SCT0_OUT9

Table continues on the next page...

Table 594. Hardware trigger sources (continued)

Trigger	Trigger source
5	CTIMER0_MAT3
6	CTIMER1_MAT3
7	CTIMER2_MAT3
8	CTIMER3_MAT3
9	CTIMER4_MAT3
10	CMP0_OUT
11	ARM_TXEV
12	SmartDMA
13	GPIOINT_BMATCH

58.1.4 Initialization

Initial configuration of the ADC can be accomplished as follows:

- Enable the clock to ADC in the PSCCTL1 (See CLKCTL). This enables the register interface and the peripheral function clock.
- Select a clock source for the ADC using the ADC0FCLKSEL1 register.
- Select a clock divide for the ADC using the ADC0FCLKDIV register.
- Clear the ADC peripheral reset in the PRSTCTL1 register by writing to the PRSTCTL1_CLR register (See RSTCTL).
- Enable the analog function of the ADC using the PDRUNCFG0 register (See SYSCTL).
- The ADC provides an interrupt to the NVIC. To allow interrupts to wake-up the device from deep-sleep mode, enable this in the STARTEN0 register (See SYSCTL).
- Configure the pins used by the ADC with IOPCTL configuration register. The ADC DMA request lines are connected to the DMA trigger inputs via the DMACn_ITRIG_SEL registers.

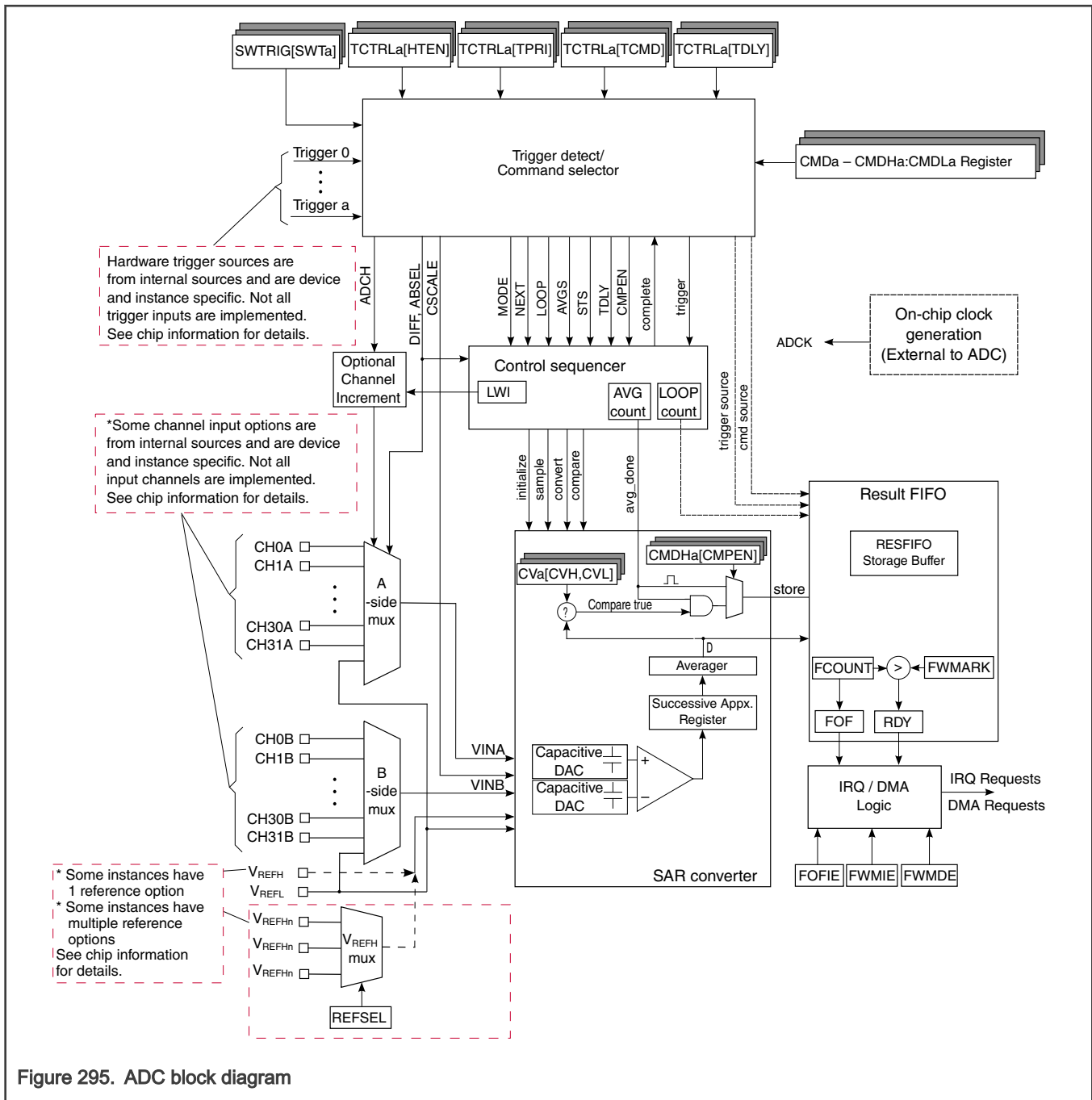
58.2 Overview

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-a-chip.

NOTE

For chip-specific modes of operation, see the power management information for the device.

58.2.1 Block diagram



58.2.2 Features

ADC has these features:

- Linear successive approximation algorithm
 - Differential operation with 13-bit resolution
 - Single-ended operation with 12-bit resolution
- Channel scaling, to allow input voltage levels higher than the ADC reference voltage
- Configurable analog input sample time

- Configurable speed options to accommodate operation in low-power modes of SoC
- Trigger detection with up to 16 trigger sources with priority level configuration. Software or hardware trigger option for each.
- 15 command buffers, to allow independent options selection and channel sequence scanning
- Automatic comparisons for less-than, greater-than, within range, or out-of-range with "store on true" and "repeat until true" options
- 2 independent result FIFOs, each containing 16 entries. Each FIFO has configurable watermark and overflow detection.
- Interrupt, DMA, or polled operation

58.3 Functional description

ADC performs analog-to-digital conversions on any of the software-selectable analog input channels via a successive approximation algorithm.

The ADC module can average the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting `CMDHn[AVGS]` to a non-zero value. The function operates in any conversion mode or configuration.

ADC can compare the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range, or outside-range detection. The compare function operates in any conversion mode or configuration.

When the conversion and averaging loops finish, the resulting data is placed in one of 2 available FIFO data buffers. The data includes tag information associated with the result. When the number of stored data words exceeds the setting, a configurable watermark level supports interrupts or DMA requests. Interrupts can also be enabled to indicate when FIFO overflow errors occur.

The module initializes to its lowest power state during reset.

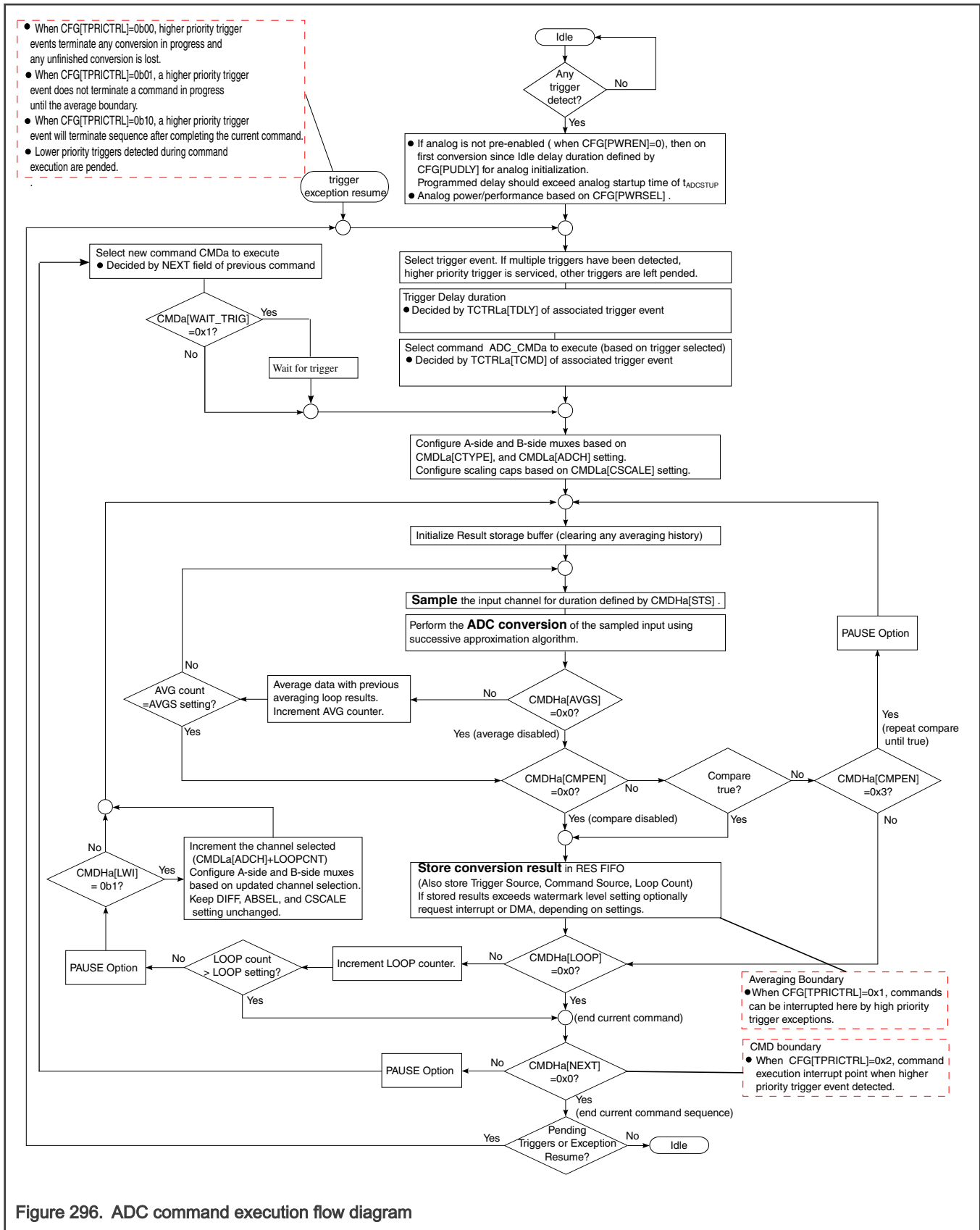
The ADC analog circuits can be pre-enabled to begin conversions sooner at the expense of higher idle currents. Conversions are initiated by selectable trigger events from software or hardware sources.

The trigger-detection logic includes a configurable enable and priority scheme for available trigger sources.

ADC includes multiple command buffers to provide flexibility for channel scanning and independent channel selections for different trigger sources. These command buffers can be configured for:

- Input scaling factor
- Differential or single-ended conversion
- Sample time
- Averaging on a per-channel basis

The sequencing of an ADC command is summarized in the flow diagram below.



58.3.1 Power control mode

By default, the ADC analog circuits are disabled while ADC is in its idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled. These circuits require a period of initialization before the first conversion cycle.

The value of [CFG\[PUDLY\]](#) should be set to incur a delay longer than $t_{ADCSTUP}$. The accuracy of initial conversions after activation is degraded if the value of [CFG\[PUDLY\]](#) is too low.

You can achieve faster conversion startup times by setting [CFG\[PWREN\]](#) to pre-enable the analog circuits of ADC. This faster conversion consumes extra power, even while ADC is in an idle state. When [CFG\[PWREN\]](#) is 1, the Power Enable timer is activated. The timer enforces the minimum time required (configured by [CFG\[PUDLY\]](#)) before detected triggers can initiate ADC conversions.

ADC also has options for controlling power and performance summarized in the table below. See the device data sheet for specification of the available power modes.

Table 595. Power option settings

CFG[PWRSEL]	Description
00b (Default)	Slowest speed and lowest power
01b	Faster speed and higher power
10b	Even faster speed and even higher power
11b	Fastest speed and highest power

In the two highest power mode settings ([CFG\[PWRSEL\]](#) = 2h and [CFG\[PWRSEL\]](#) = 3h) the ADC delays sampling for 16 ADC clock cycles after leaving the idle state. This delay is in addition to any configured power-up delay or trigger delay.

58.3.2 Operations

58.3.2.1 ADC modes of operation

The ADC module supports the chip low power modes described in the following table. See section [Clock operation](#) for more information.

Table 596. Chip modes supported by the ADC module

Chip mode	ADC Operation
Run	Normal operation
Stop/Wait	When the Doze Enable bit (CTRL[DOZEN]) is 0, the ADC can continue to operate and the module is using an external or internal clock source which remains operating during Stop/Wait modes. ADC is allowed to continue operation while the system is transitioned to the low power state. Any conversion in progress is not disrupted. External hardware trigger detect and active conversions are operational. When the DOZEN bit is 1, the module waits for the current averaging iteration/ FIFO storage to complete before acknowledging stop or wait mode entry. There is no associated ack/handshake and the system transitions to low power state without ADC interaction. ADC operation should terminate after completion of any conversion in progress.
Low Leakage Stop	The Doze Enable (CTRL[DOZEN]) bit is ignored and the module waits for the current transfer to complete any pending operation before acknowledging low leakage mode entry.

58.3.2.2 Voltage reference

The voltage reference high (V_{REFH}) used by ADC is supplied from either an on-chip voltage reference source or from an off-chip source via external pins. V_{REFL} is always from an external pin and must be at the same voltage as V_{SSA} .

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via [CFG\[REFSEL\]](#)).

See the chip configuration information on the voltage reference options specific to this packaged device.

58.3.2.3 Trigger detect and command execution

See [Figure 296](#) for a flow diagram of command execution sequencing.

ADC command execution is initiated from up to 16 trigger sources. Each trigger can be software-generated by writing 1b to the corresponding [SWTRIG\[SWTn\]](#) field. Alternatively, asynchronous input sources at the periphery of the module can generate hardware triggers. The number and sources of hardware triggers implemented is device-specific. See the chip-specific ADC information for descriptions of available hardware trigger sources for this device.

Each hardware trigger source is enabled by setting the associated enable field ([TCTRLn\[HTEN\]](#)). Each trigger source is assigned a priority via the associated priority control field ([TCTRLn\[TPRI\]](#)). Each of the trigger sources is associated with a command buffer via the associated command select field ([TCTRLn\[TCMD\]](#)).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising edge of the associated hardware trigger source.

Each trigger source has an associated priority field [TCTRLn\[TPRI\]](#) which allows arbitration between trigger sources. Arbitration selects two things: which trigger sequence to execute next, and how to handle a trigger exception. Trigger exceptions are defined as allowing a higher-priority trigger sequence to interrupt operation of a lower-priority sequence. When a trigger exception occurs, programmable arbitration allows the configurable stop and resume points for low-priority sequences. The fields affecting arbitration are [CFG\[HPT_EXDI\]](#), [CFG\[TCMDRES\]](#), [CFG\[TRES\]](#), and [CFG\[TPRCTRL\]](#).

1. When [CFG\[HPT_EXDI\]](#) = 1b, trigger exceptions are disabled and any higher priority triggers are left pending until the current sequence completes. New triggers are accepted based on priority.
2. When [CFG\[HPT_EXDI\]](#) = 0b (default), exceptions are enabled and the higher-priority sequence begins executing at a user-specified breakpoint.

Breakpoint locations are determined by [CFG\[TPRCTRL\]](#), which affects latency for accepting trigger exceptions.

1. When [CFG\[TPRCTRL\]](#) = 0h, a higher-priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started.
2. When [CFG\[TPRCTRL\]](#) = 1h, the current conversion is allowed to complete (including averaging) before the higher-priority exception starts. In this mode, if the command is running through a series of averages, this series completes. However, there is no requirement to finish the entire command before being interrupted. For example, if the command consists of four loop iterations, there is no requirement to complete all four iterations before the interrupt occurs.
3. When [CFG\[TPRCTRL\]](#) = 2h, a higher-priority trigger begins once the current command is completed. If a command consists of five loop iterations each containing eight averages, then all 40 conversions must be completed before accepting the trigger exception.

[CFG\[TCMDRES\]](#) and [CFG\[TRES\]](#) determine what ADC does after accepting a trigger exception. The module can be programmed to resume commands after returning from a trigger exception.

1. When [CFG\[TRES\]](#) = 0h, commands are not automatically resumed after being stopped by an exception. However, an interrupt is set to indicate that this case has occurred. The flag [TSTAT\[TEXC_NUM\]](#) can be used to resolve which trigger was stopped by the exception.
2. When [CFG\[TRES\]](#) = 1h, ADC automatically resumes commands after they were stopped by an exception.

Using [CFG\[TRES\]](#) with [CFG\[TCMDRES\]](#), the module can be programmed to resume commands at one of two possible locations.

1. When [CFG\[TCMDRES\]](#) = 0h, the trigger which was stopped by an exception is resumed from the beginning of its associated command sequence. Triggers that are waiting to be resumed take the same priority programmed to [TCTRLn\[TPRI\]](#).

- When `CFG[TCMDRES] = 1h`, the trigger is resumed from the command that it was executing before being interrupted by an exception.

If a trigger occurs with priority lower than the priority of the currently executing command, trigger detection is left pending until the current command sequence finishes. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or command sequence) completes.

When a conversion is completed (including hardware averaging when `CMDHn[AVGS]` is non-zero), the result is placed in a RESFIFO buffer. When an ADC command selects looping (when `CMDHn[LOOP]` is non-zero) a command stores multiple conversion results to the FIFO during execution of that command.

At the end of command execution, `CMDHn[NEXT]` selects the next command to be executed. Multiple commands can be executed sequentially by configuring the `CMDHn[NEXT]` field of each command. Setting the next command to 0h causes conversions to terminate at the completion of the current command. Unending circular command execution is allowed by setting the `CMDHn[NEXT]` field of the last command in a sequence to the first command in the sequence.

By default, command sequences execute automatically in the order in which `CMDHn[NEXT]` fields are programmed. However, by using `CMDHn[WAIT_TRIG]`, command execution can be stalled and launched based on trigger inputs. For example, if `TRIGGER2` is programmed to start the command sequence `CMD1`, `CMD2`, `CMD3`, then this sequence is run once unconditionally to completion upon receiving `TRIGGER2`. If `CMDH2[WAIT_TRIG] = 1h`, however, the sequence pauses after `CMD1` until `TRIGGER2` is received again. In this way, sequences can be stalled until a trigger assertion is received.

Disabling ADC by writing 0b to `CTRL[ADCEN]` terminates any active ADC command processing. Writing 0b to `CTRL[ADCEN]` causes the current command (or command sequence) to terminate, clears any pending triggers, and sends the ADC module to an idle state.

58.3.2.4 Pause option

When an application does not require the maximum conversion rate, the effective conversion rate can be reduced:

- By implementing periodic trigger events to initiate ADC conversions, or
- By selecting a reduced-frequency clock as the `ADCK` source.

Both options are chip-specific and are dependent on ADC triggering and clocking options external to the ADC module. The latency associated with ADC analog power-up delays limits the maximum conversion rate when using periodic triggering.

Conversion rates can also be reduced by inserting a pause of a programmable duration:

- Between loop iterations
- Between commands in a sequence
- Between conversions, when a command is executing in the Compare Until True configuration

When `PAUSE[PAUSEEN] = 1`, `PAUSE[PAUSEDLY]` controls the duration of pausing during command execution sequencing. The pause delay is a count of $(\text{PAUSE[PAUSEDLY]} * 4)$ `ADCK` cycles.

NOTE

Do not change the `PAUSE` register while `CTRL[ADCEN] = 1`. Writes to the `PAUSE` register while `CTRL[ADCEN]` is 1 can lead to metastable operation.

See [Figure 296](#) for the places during command execution sequencing where the pause can be inserted.

58.3.2.5 Resync functionality

Any trigger source (SW or HW) can be configured to act as a resync trigger. Trigger based resync functionality is used to interrupt a running trigger (resync target) and clear the FIFO it's writing to. This can either be used to abort a running sequence, or restart a running sequence depending on the configuration of `CFG[TRES]`. If `CFG[TRES] = 0b1` then the target sequence is aborted, the FIFO cleared, and the sequence restarted after the resync occurs. If `CFG[TRES] = 0b0` then the target sequence is aborted and the FIFO is cleared after the `RESYNC` occurs. Note, the FIFO(s) cleared are based on the resync target `TCTRLm[FIFO_SEL_A]` and `TCTRLm[FIFO_SEL_B]`. To only clear one FIFO, `TCTRLm[FIFO_SEL_A] == TCTRLm[FIFO_SEL_B]`. Any results not

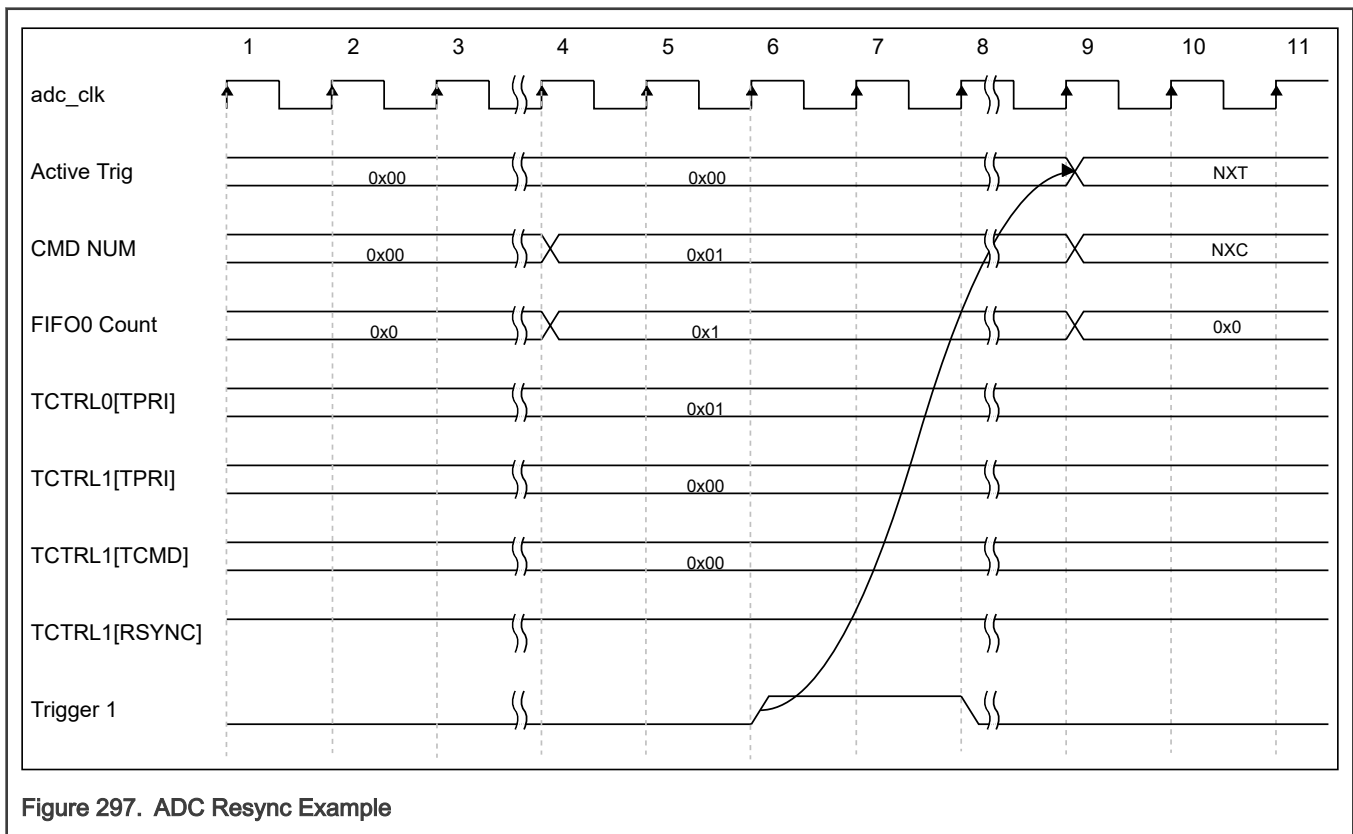
associated with the resync target are lost if they are stored in either TCTRLm[FIFO_SEL_A] or TCTRLm[FIFO_SEL_B] at the time of the resync.

A resync trigger needs to have a specific target. The resync only occurs if the resync target is running at the time of the trigger. For the following description, let n be the resync trigger number, let m be the resync target number. According to these variables, trigger n should resync trigger m. To enable a trigger source to act as a resync trigger, the following conditions must be satisfied:

1. The resync trigger TCTRLn[RSYNC] must be set to 0b1.
2. The resync trigger must have higher priority than the resync target (TCTRLn[TPRI] must be less than TCTRLm[TPRI]).
3. The resync target is specified using TCTRLn[TCMD]. In this case the resync target, m, must be equal to TCTRLn[TCMD].
4. The resync target, m, must be executing commands when the resync trigger, n, is asserted.
5. Trigger m must have at least one conversion left to begin when trigger n is received.

If a trigger source n has TCTRLn[RSYNC] set to 0b1, but some of the above conditions aren't met then the trigger source n is ignored.

The following figure illustrates a resync trigger sequence executing. Note, in this example, trigger source 1 is configured to resync trigger source 0.



In this figure, trigger source 0 is executing a sequence of commands when trigger source 1 is asserted. Notice that trigger source 0 is stopped when trigger source 1 is asserted (after some synchronization delay). In addition, the FIFO being written to by trigger source 0 is cleared (FIFO0 in this example). After the trigger 0 sequence is stopped, the ADC runs the next trigger pending with the highest priority. This is marked as NXT, for next trigger. If resume functionality is enabled, and trigger source 0 had the highest priority pending, then NXT = 0x00.

58.3.2.6 Temperature sensor

The ADC module includes a temperature sensor whose output is connected to one of the analog channel inputs.

NOTE

The temperature sensor may only be connected to individual instances of the ADC module and may not be available on all instances of ADC. See device-specific analog channel information for an explanation of which channel is used to convert the temperature sensor data.

The following equation is an approximate transfer function of the temperature sensor.

$$Temp = 25 - \left(\frac{V_{TEMP} - V_{TEMP25}}{m} \right)$$

Figure 298. Approximate transfer function of the temperature sensor

Where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25 °C.
- m is temperature sensor slope.

In application code, the user executes a conversion on the temperature sensor channel and converts the result to a voltage (V_{TEMP}) based on the voltage reference. The ambient temperature is then calculated using the above equation. V_{TEMP25} and the temperature sensor slope values are specified constant values from the ADC electrical information in the device data sheet.

When converting the temperature sensor channel data, these command selections must be configured:

- Differential mode ($CMDLn[DIFF] = 1h$)
- Maximum averaging ($CMDHn[AVGS] = 7h$)
- Maximum sample time ($CMDHn[STS] = 7h$)

58.3.2.7 Result FIFO operation

ADC includes 2 16-entry FIFOs in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command, and the loop count are also stored with the data. $FCTRLn[FCOUNT]$ indicates how many valid data words are stored in each RESFIFO.

A programmable watermark threshold supports configurable notification of data availability. When $FCTRLn[FCOUNT]$ is greater than $FCTRLn[FWMARK]$, the associated RDY flag is asserted. When $IE[FWMIE] = 1$, a watermark interrupt request is issued. When $DE[FWMDEn] = 1$, a DMA request is issued. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements $FCTRLn[FCOUNT]$. When $FCTRLn[FCOUNT]$ falls equal to or below $FCTRLn[FWMARK]$, the RDY flag is cleared.

Each FIFO can be emptied by successive reads of RESFIFO. When $RESFIFO[VALID]$ is 1, the associated FIFO entry is valid. Reading RESFIFO when the FIFO is empty (when $RESFIFO[VALID] = 0$ and $FCTRLn[FCOUNT] = 0h$) provides an undefined data word. All FIFOs are reset by writing 1b to $CTRL[RSTFIFO]$.

If ADC attempts to store a data word to the FIFO when the FIFO is full, the FIFO overflow flag ($FCTRLn[FOF]$) is set. When $IE[FOFIE] = 1$, an overflow interrupt request is issued. The FOF flag is cleared by writing 1 to $STAT[FOFn]$. When overflow events occur, no new data is stored and the data associated with the storage event that triggered the overflow is lost.

Conversion results can be steered to any FIFO in the design. $TCTRLn[FIFO\_SEL\_A]$ and $TCTRLn[FIFO\_SEL\_B]$ determine into which FIFO the final result is written. Depending on which trigger is executing, the results can be steered to different locations. Depending on the type of conversion selected, the FIFO destination register fields are interpreted differently. During either differential or single-ended mode ($CMDLn[CTYPE] \neq 3h$) only one result is produced. The destination during these modes is determined from $TCTRLn[FIFO\_SEL\_A]$. In dual-single-ended mode, both $TCTRLn[FIFO\_SEL\_A]$ and $TCTRLn[FIFO\_SEL\_B]$ determine the Channel A and Channel B destinations respectively.

58.3.2.8 Compare Function

After the input is sampled and converted and any averaging iterations are performed, $CMDHn[CMPE]$ determines:

- Whether to use the automatic compare function, storing the conversion result when true.

- Whether to repeat the channel acquisition until the automatic compare function returns a true result.

The command-sequencing options related to the compare function are summarized in the table below.

NOTE

Not all command buffers have an associated compare value register. The compare function is only available on command buffers that have a corresponding compare value register.

Table 597. Compare modes

CMDHn[CMPEP]	Compare function	Description
00b	Compare disabled	Do not perform compare operation. Always store the conversion result to the FIFO.
01b	Reserved	
10b	Store on true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Regardless of the comparison result, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required.
11b	Repeat compare until true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Once a comparison is true, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required. When a comparison is false, the conversion is repeated without considering the LOOP setting, and the LOOP counter is not incremented.

The compare operation checks the result based on the values of CVn[CVH] and CVn[CVL], as shown in the following table.

Table 598. Compare operations

CVn[CVL] vs. CVn[CVH]	Operation	Description
set CVn[CVL] < CVn[CVH]	Outside range (General form)	True if the result is less than CVn[CVL] or greater than CVn[CVH].
set CVn[CVH] to maximum value set CVn[CVL] to compare point	Less than	True if the result is less than CVn[CVL].
set CVn[CVL] to minimum value set CVn[CVH] to compare point	Greater than	True if the result is greater than CVn[CVH].
set CVn[CVL] > CVn[CVH]	Inside range	True if the result is less than CVn[CVL] and greater than CVn[CVH].

NOTE

When ADC continues operating in a low-power mode, the compare function can monitor the voltage and wake the device only when the compare condition is met.

58.3.3 Clock operation

The ADC operates from the ADCK clock input provided from an on-chip clock select block and is used by the SAR conversion control sequencing logic and the FIFO storage buffer. The ADCK frequency must be within the specified frequency range for ADCK and varies based on configuration of CFG[PWRSEL]. Refer to the device datasheet for supported frequency ranges.

The ADC continues operating in stop and wait modes provided the Doze Enable bit (CTRL[DOZEN]) is clear and the on-chip clock select block continues to supply an ADCK clock source. Note, in stop mode with CTRL[DOZEN] == 0b0, the bus clock can be shut off, and asynchronous interrupts and DMA requests can be configured. In addition, the ADC continues processing commands and writing data to the internal FIFO. The ADC has four sources for async interrupts during stop mode: watermark, FIFO overflow, TCOMP, and TEXC. To enable them, properly configure the bits IE[FWMIE], IE[FOFIE], IE[TCOMP_IE], and IE[TEXC_IE] before entering stop mode.

When the DOZEN bit is set in stop and wait modes, the ADC waits for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry. Any pending triggers are dropped when a stop/wait mode request is made with DOZEN set. The ADC is forced into its lowest power setting after acknowledging the DOZEN stop/wait mode request. The same behavior is observed when entering a Low Leakage Stop Mode.

58.4 External signals

The ADC module supports analog channel inputs with differential and single-ended conversion options for all channels. ADC requires supply and ground connections.

Table 599. ADC signal descriptions

Signal	Description	I/O
V _{DDA}	Analog Power Supply	I
V _{SSA}	Analog Ground	I
VDDA_BIAS	Analog Supply Bias. See note below discussing in more detail.	I
CHnA - CH0A ¹	A-side Analog Channel Inputs	I
CHnB - CH0B ¹	B-side Analog Channel Inputs	I

- where n is the maximum channel number supported in the chip. See the chip-specific information for the number of channels supported on your device.

NOTE

The VDDA_BIAS supply used by ADC is chip-specific. It is connected to a shared supply source in some implementations. In other implementations, VDDA_BIAS is supplied through an external pin from an off-chip source. See the chip configuration information on the VDDA_BIAS connections specific to this packaged device. When connected to an external pin, VDDA_BIAS should be connected to the same supply source as the one used by the GPIO for analog channel inputs. For chips supporting multiple I/O supply levels via pad segments, the VDDA_BIAS supply should be the highest I/O supply level associated with ADC analog channels used in the application.

58.4.1 Analog channel inputs (CHnA and CHnB)

[CMDLn\[ADCH\]](#), [CMDLn\[DIFF\]](#), and [CMDLn\[ABSEL\]](#) control selection of paired or individual input channels. Each ADC command independently selects a channel. Each [CMDLn\[ADCH\]](#) channel selection has an associated A-side and B-side input. Each [CMDLn\[ADCH\]](#) pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

Some input channels from on-chip sources, such as temperature sensors and reference voltage sources, may only be connected to individual instances of ADC. Some input channel options in the field-setting descriptions may not be available for your device. See chip-specific information for the channels supported on this device.

58.5 Initialization

58.6 ADC register descriptions

This section describes the ADC registers.

58.6.1 ADC memory map

ADC0 base address: 4013_A000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	R	0200_201Ah
4h	Parameter Register (PARAM)	32	R	0F04_1010h
10h	Control Register (CTRL)	32	RW	0000_0000h
14h	Status Register (STAT)	32	RW	0000_0000h
18h	Interrupt Enable Register (IE)	32	RW	0000_0000h
1Ch	DMA Enable Register (DE)	32	RW	0000_0000h
20h	Configuration Register (CFG)	32	RW	0080_0000h
24h	Pause Register (PAUSE)	32	RW	0000_0000h
34h	Software Trigger Register (SWTRIG)	32	RW	0000_0000h
38h	Trigger Status Register (TSTAT)	32	RW	0000_0000h
A0h - DCh	Trigger Control Register (TCTRL0 - TCTRL15)	32	RW	0000_0000h
E0h - E4h	FIFO Control Register (FCTRL0 - FCTRL1)	32	RW	0000_0000h
100h	Command Low Buffer Register (CMDL1)	32	RW	0000_2000h
104h	Command High Buffer Register (CMDH1)	32	RW	0000_0000h
108h	Command Low Buffer Register (CMDL2)	32	RW	0000_2000h
10Ch	Command High Buffer Register (CMDH2)	32	RW	0000_0000h
110h	Command Low Buffer Register (CMDL3)	32	RW	0000_2000h
114h	Command High Buffer Register (CMDH3)	32	RW	0000_0000h
118h	Command Low Buffer Register (CMDL4)	32	RW	0000_2000h
11Ch	Command High Buffer Register (CMDH4)	32	RW	0000_0000h
120h	Command Low Buffer Register (CMDL5)	32	RW	0000_2000h
124h	Command High Buffer Register (CMDH5)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
128h	Command Low Buffer Register (CMDL6)	32	RW	0000_2000h
12Ch	Command High Buffer Register (CMDH6)	32	RW	0000_0000h
130h	Command Low Buffer Register (CMDL7)	32	RW	0000_2000h
134h	Command High Buffer Register (CMDH7)	32	RW	0000_0000h
138h	Command Low Buffer Register (CMDL8)	32	RW	0000_2000h
13Ch	Command High Buffer Register (CMDH8)	32	RW	0000_0000h
140h	Command Low Buffer Register (CMDL9)	32	RW	0000_2000h
144h	Command High Buffer Register (CMDH9)	32	RW	0000_0000h
148h	Command Low Buffer Register (CMDL10)	32	RW	0000_2000h
14Ch	Command High Buffer Register (CMDH10)	32	RW	0000_0000h
150h	Command Low Buffer Register (CMDL11)	32	RW	0000_2000h
154h	Command High Buffer Register (CMDH11)	32	RW	0000_0000h
158h	Command Low Buffer Register (CMDL12)	32	RW	0000_2000h
15Ch	Command High Buffer Register (CMDH12)	32	RW	0000_0000h
160h	Command Low Buffer Register (CMDL13)	32	RW	0000_2000h
164h	Command High Buffer Register (CMDH13)	32	RW	0000_0000h
168h	Command Low Buffer Register (CMDL14)	32	RW	0000_2000h
16Ch	Command High Buffer Register (CMDH14)	32	RW	0000_0000h
170h	Command Low Buffer Register (CMDL15)	32	RW	0000_2000h
174h	Command High Buffer Register (CMDH15)	32	RW	0000_0000h
200h - 20Ch	Compare Value Register (CV1 - CV4)	32	RW	0000_0000h
300h - 304h	Data Result FIFO Register (RESFIFO0 - RESFIFO1)	32	R	0000_0000h

58.6.2 Version ID Register (VERID)

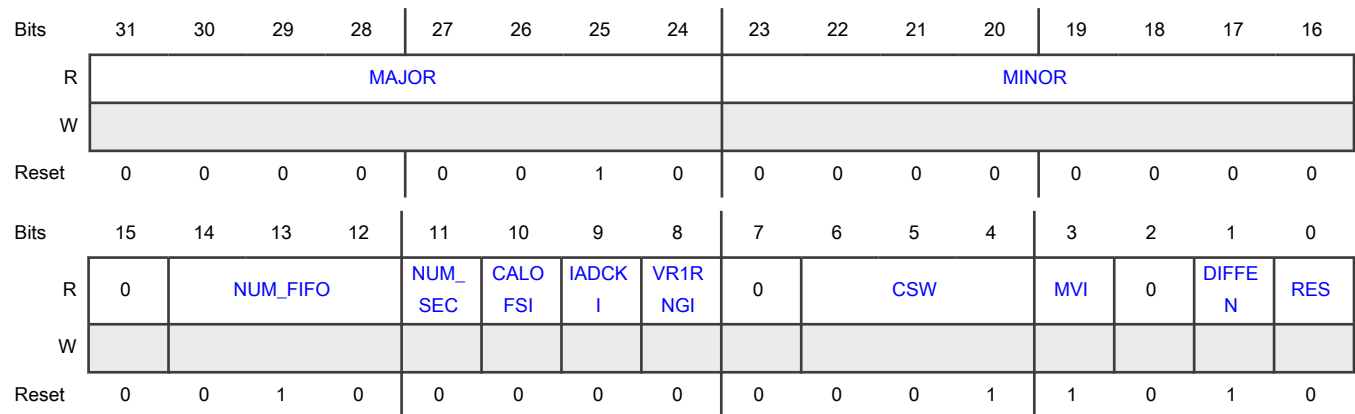
Offset

Register	Offset
VERID	0h

Function

Indicates the version integrated for this instance on the device and the inclusion of optional features.

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15 —	Reserved
14-12 NUM_FIFO	Number of FIFOs Indicates the number of result FIFOs implemented in the design. 000b - N/A 001b - One 010b - Two 011b - Three 100b - Four
11 NUM_SEC	Number of Single-Ended Outputs Supported Indicates the number of single-ended channels which can be processed simultaneously. 0b - One 1b - Two
10 CALOFSI	Calibration Function Implemented Indicates whether a calibration function and trimming control register are implemented. 0b - Not implemented 1b - Implemented

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 IADCKI	Internal ADC Clock Implemented Indicates whether this implementation of the ADC block includes an internal clock source. 0b - Not implemented 1b - Implemented
8 VR1RNGI	Voltage Reference 1 Range Control Bit Implemented Indicates whether a control bit is implemented to select the input voltage range on Voltage Reference Option 1. 0b - Range control not required. 1b - Range control required.
7 —	Reserved
6-4 CSW	Channel Scale Width Indicates whether channel scaling is supported. When supported, each command buffer has a control field (CMDLn[CSCALE]) for setting input scaling. 000b - Not supported. 001b - Supported with one-bit CSCALE control field. 110b - Supported with six-bit CSCALE control field.
3 MVI	Multiple Vref Implemented Indicates whether multiple voltage reference high (VREFH) inputs are supported. When multiple voltage references are supported, CFG[REFSEL] selects voltage reference high options. 0b - Single VREFH input supported. 1b - Multiple VREFH inputs supported.
2 —	Reserved
1 DIFFEN	Differential Supported Indicates whether differential operation is supported. When supported, each command buffer has control fields (CMDLn[DIFF] and CMDLn[ABSEL]) for configuring for differential operation and expanding the number of supported channels from 32 to 64. 0b - Not supported 1b - Supported. CMDLn[DIFF] and CMDLn[ABSEL] control fields implemented.
0 RES	Resolution Indicates the maximum accuracy supported.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Up to 13-bit differential or 12-bit single-ended resolution supported. 1b - Up to 16-bit differential or 15-bit single-ended resolution supported. CMDL _n [MODE] available for selecting the resolution of conversions for the associated command.

58.6.3 Parameter Register (PARAM)

Offset

Register	Offset
PARAM	4h

Function

The Parameter register indicates the size of several variable integration options for this instance on the device.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMD_NUM								CV_NUM							
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIFOSIZE								TRIG_NUM							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Function
31-24 CMD_NUM	Command Buffer Number Indicates number of command buffers implemented.
23-16 CV_NUM	Compare Value Number Indicates number of compare value registers implemented.
15-8 FIFOSIZE	Result FIFO Depth Indicates the maximum number of conversion data words that can be stored in the result FIFO before an overflow occurs.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0000_0001b - 2 0000_0100b - 4 0000_1000b - 8 0001_0000b - 16 0010_0000b - 32 0100_0000b - 64
7-0 TRIG_NUM	Trigger Number Number of Triggers implemented.

58.6.4 Control Register (CTRL)

Offset

Register	Offset
CTRL	10h

Function

Includes primary control bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						0	0	0		0	0	0	DOZE	RST	ADCE
W							RSTFI	RSTFI						N		N
							FO1	FO0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-10 —	Reserved
9 RSTFIFO1	Reset FIFO 1 0b - No effect. 1b - FIFO 1 is reset.
8 RSTFIFO0	Reset FIFO 0 0b - No effect. 1b - FIFO 0 is reset.
7-5 —	Reserved
4 —	Reserved
3 —	Reserved
2 DOZEN	Doze Enable Controls system transition to Stop and Wait power modes while ADC is converting. When 0, immediate entries to Stop or Wait modes are allowed and ADC conversion functions remain enabled. Any conversion in progress is not disrupted. The selected clock source provided from the on-chip clock source must be able to continue operating. When 1, the ADC waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power entry request. After entering the low-power state, ADC is kept inactive until the system exits the low-power state. When the system enters Low-Leakage Stop mode, the DOZEN bit is ignored and ADC waits for the current transfer to complete any pending operation. 0b - ADC is enabled in low-power mode. 1b - ADC is disabled in low-power mode.
1 RST	Software Reset Resets all internal logic and registers, except the CTRL register. Remains 1 until cleared by software. 0b - ADC logic is not reset. 1b - ADC logic is reset.
0 ADCEN	ADC Enable 0b - Disabled 1b - Enabled

58.6.5 Status Register (STAT)

Offset

Register	Offset
STAT	14h

Function

Provides the status of the ADC module.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				CMDACT				0				TRGACT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				ADC_ ACT...	0	TCOM P_...	TEXC_ INT	0				FOF1	RDY1	FOF0	RDY0
W							W1C	W1C					W1C		W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-28 —	Reserved
27-24 CMDACT	Command Active Indicates the command actively being processed. Commands are only shown here when they are actively being processed by the SAR routine. Use with STAT[ADC_ACTIVE] to determine which conversion is running. 0000b - No command currently in progress. 0001b - Command 1 currently being executed. 0010b - Command 2 currently being executed. 0011b-1111b - Associated command number currently being executed.
23-20 —	Reserved
19-16 TRGACT	Trigger Active

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Indicates the trigger actively being processed. This field can be used to determine which trigger is running through a trigger delay or being converted by the SAR routine. The command associated with this field is running through a conversion when STAT[CMDACT] is not zero. 0000b - Command (sequence) associated with Trigger 0 currently being executed. 0001b - Command (sequence) associated with Trigger 1 currently being executed. 0010b - Command (sequence) associated with Trigger 2 currently being executed. 0011b-1111b - Command (sequence) from the associated Trigger number currently being executed.
15-12 —	Reserved
11 ADC_ACTIVE	ADC Active Indicates whether the module is processing a conversion, or has pending triggers to service. 0b - ADC is idle. There are no pending triggers to service and no active commands are being processed. 1b - ADC is processing a conversion, running through the power-up delay, or servicing a trigger.
10 —	Reserved
9 TCOMP_INT	Interrupt Flag For Trigger Completion Indicates when a trigger sequence has been completed (all associated commands have been run). IE[TCOMP_IE] must be 1 for the specific trigger source to flag an interrupt upon completion. 0b - Either IE[TCOMP_IE] = 0, or no trigger sequences have run to completion. 1b - Trigger sequence has been completed and all data is stored in the associated FIFO.
8 TEXC_INT	Interrupt Flag For High-Priority Trigger Exception Indicates when a high-priority trigger exception has occurred and CFG[TRES] = 0. This flag only asserts if trigger exception interrupts are enabled (IE[TEXC_IE] = 1h). This flag can be used with TSTAT[TEXC_NUM] to resolve which trigger source was interrupted. 0b - No trigger exceptions have occurred. 1b - A trigger exception has occurred and is pending acknowledgment.
7-4 —	Reserved
3 FOF1	Result FIFO1 Overflow Flag Indicates that more data has been written to the Result FIFO 1 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE1]. However, an interrupt request is issued only if IE[FOFIE1] is 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - No result FIFO1 overflow has occurred since the last time that the flag was cleared. 1b - At least one result FIFO1 overflow has occurred since the last time that the flag was cleared.
2 RDY1	Result FIFO1 Ready Flag Indicates when the number of valid data words in the result FIFO 1 is greater than the level set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE1]. However, an interrupt request or DMA request occurs only when the associated control bit (IE[FWMIE1] and DE[FWMDE1]) is set. This flag is cleared when FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0b - Not above watermark 1b - Above watermark
1 FOF0	Result FIFO 0 Overflow Flag Indicates that more data has been written to the Result FIFO 0 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE0]. However, an interrupt request is issued only if IE[FOFIE0] is 1. 0b - No result FIFO 0 overflow has occurred since the last time that the flag was cleared. 1b - At least one result FIFO 0 overflow has occurred since the last time that the flag was cleared.
0 RDY0	Result FIFO 0 Ready Flag Indicates when the number of valid data words in the result FIFO 0 is greater than the watermark set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE0]. However, an interrupt request or DMA request occurs only when the associated control field (IE[FWMIE0] and DE[FWMDE0]) is 1. This flag is cleared when the FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0b - Not above watermark 1b - Above watermark

58.6.6 Interrupt Enable Register (IE)

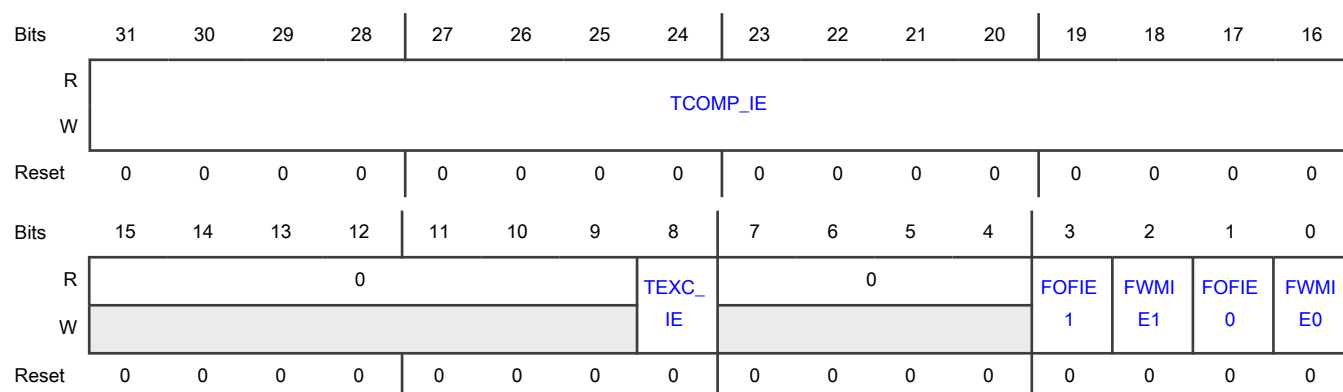
Offset

Register	Offset
IE	18h

Function

Includes system interrupt masking control bits.

Diagram



Fields

Field	Function
31-16 TCOMP_IE	Trigger Completion Interrupt Enable Enables generation of interrupt requests to indicate when complete trigger command sequences are executed. Each bit in TCOMP_IE corresponds to a trigger source. (TCOMP_IE[0] corresponds to trigger 0, and so on.) All results are stored in the FIFO when a sequence is considered complete. 0000_0000_0000_0000b - All disabled 0000_0000_0000_0001b - Trigger completion interrupts are enabled for trigger source 0 only. 0000_0000_0000_0010b - Trigger completion interrupts are enabled for trigger source 1 only. 0000_0000_0000_0011b-1111_1111_1111_1110b - Associated trigger completion interrupts are enabled. 1111_1111_1111_1111b - All enabled
15-9 —	Reserved
8 TEXC_IE	Trigger Exception Interrupt Enable Enables ADC to assert an interrupt request when a high-priority trigger exception occurs. TSTAT[TEXC_NUM] contains the value of the corresponding trigger affected by the exception. 0b - Disabled 1b - Enabled
7-4 —	Reserved
3 FOFIE1	Result FIFO1 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF1 flag is asserted. 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 FWMIE1	FIFO1 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY1] flag is asserted. 0b - Disabled 1b - Enabled
1 FOFIE0	Result FIFO 0 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF flag is asserted. 0b - Disabled 1b - Enabled
0 FWMIE0	FIFO 0 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY0] flag is asserted. 0b - Disabled 1b - Enabled

58.6.7 DMA Enable Register (DE)

Offset

Register	Offset
DE	1Ch

Function

Includes DMA request-masking control bits.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0														FWMD	FWMD
W															E1	E0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-2 —	Reserved
1 FWMDE1	FIFO1 Watermark DMA Enable Enables generation of DMA requests when STAT[RDY1] flag is asserted. 0b - Disabled 1b - Enabled
0 FWMDE0	FIFO 0 Watermark DMA Enable Enables generation of DMA requests when STAT[RDY0] flag is asserted. 0b - Disabled 1b - Enabled

58.6.8 Configuration Register (CFG)

Offset

Register	Offset
CFG	20h

Function

Controls ADC functions common to all commands. When [CTRL\[ADCEN\]](#) = 1, this register cannot be changed and writes to this register are ignored.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	PWREN	0				PUDLY							
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0				HPT_EXDI	TCMDRES	TRES	REFSEL		PWRSEL		0		TPRCTRL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	Reserved
30 —	Reserved
29 —	Reserved
28 PWREN	ADC Analog Pre-Enable Enables the ADC analog circuits. When setting this field, user code should delay for a period exceeding the analog startup time of $t_{ADCSTUP}$ before enabling ADC for operation. The module is still operational even when this field is 0, but command execution start is delayed for a period defined by CFG[PUDLY]. See the device data sheet for ADC idle and active power consumption parameters. 0b - ADC analog circuits are only enabled while conversions are active. Analog startup delays affect performance. 1b - ADC analog circuits are pre-enabled and ready to execute conversions without startup delays, at the cost of higher DC current consumption. A single power-up delay (CFG[PUDLY]) is executed immediately once PWREN is set. No detected triggers begin ADC operation until the power-up delay time has passed. After this initial delay expires, the analog circuits remain pre-enabled, and no additional delays are executed.
27-24 —	Reserved
23-16 PUDLY	Power-up Delay Defines the power-up delay executed after an initial trigger wakes ADC from its idle state. Must be programmed to a non-zero value to enable the ADC. Depending on the value of CFG[PWREN], the delay is executed in one of two modes: - When CFG[PWREN] = 0, the ADC analog circuits are only powered on while the module is active. The power-up delay allows time for the analog circuits to stabilize after being powered on. The startup delay count of (PUDLY * 4) ADCK cycles must result in a longer delay than the analog startup time of $t_{ADCSTUP}$. Accuracy of the initial conversions after activation is degraded if CFG[PUDLY] is set to too small a value. After active conversions, if no subsequent conversions are pending, the ADC analog circuits are automatically reverted to their low-power idle state. When ADC is awakened with a later trigger, the power-up delay runs again. - When CFG[PWREN] = 1 prior to ADC activation, the analog circuits are pre-enabled and the activation delay defined by CFG[PUDLY] is executed immediately. After the delay has been executed, the analog circuits remain enabled regardless of ADC activity. This configuration begins conversions immediately after trigger event detection, at the cost of increased DC power consumption in the analog circuits.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	NOTE ADC does not begin an initial conversion until the power-up delay has executed, regardless of the value of CFG[PWREN].
15 —	Reserved
14-11 —	Reserved
10 HPT_EXDI	High-Priority Trigger Exception Disable Disables high-priority trigger exceptions. See Trigger detect and command execution for a detailed description on trigger exception handling. When 1, exceptions are disabled and CFG[TCMDRES], CFG[TRES], and CFG[TPRCTRL] are ignored. 0b - Enabled 1b - Disabled
9 TCMDRES	Trigger Command Resume Determines where a trigger sequence resumes when interrupted by a high-priority trigger exception. To use this field, CFG[TRES] must be 1. 0b - Trigger sequence automatically restarted. 1b - Trigger sequence resumed from the command that was executed prior to the exception.
8 TRES	Trigger Resume Enable Determines whether trigger sequences interrupted by a high-priority exception are automatically resumed or restarted. If 1, the sequence can be resumed along command boundaries or restarted from the beginning of a sequence, as determined by CFG[TCMDRES]. If 0, interrupted triggers can be resumed via software by monitoring STAT[TEXC_INT] (or via ISR handling for trigger exception interrupt when IE[TEXC_IE] = 1). Software determines which trigger was interrupted by reading TSTAT[TEXC_NUM] and restarts the trigger by writing 1 to the corresponding field in the SWTRIG register. 0b - Not automatically resumed or restarted 1b - Automatically resumed or restarted
7-6 REFSEL	Voltage Reference Selection Selects the voltage reference high used for conversions. NOTE See the chip-specific ADC information for voltage reference options specific to this packaged device.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Option 1 01b - Option 2 10b - Option 3 11b - Reserved
5-4 PWRSEL	Power Configuration Select Configures the module for power and performance. In the highest-power setting, the highest conversion rates are possible. Refer to the device data sheet for power and performance capabilities for each setting. 00b - Lowest power 01b - Higher power than 00b 10b - Higher power than 01b 11b - Highest power
3-2 —	Reserved
1-0 TPRICTRL	ADC Trigger Priority Control Controls how higher-priority trigger exceptions are handled when they are received during command processing. See Trigger detect and command execution for a detailed explanation of trigger event handling. 00b - Current conversion is aborted and the new command specified by the trigger is started. 01b - Current command is stopped after completing the current conversion. If averaging is enabled, the averaging loop is completed. CMDHn[LOOP] is ignored and the higher-priority trigger is serviced. 10b - Current command is completed (averaging, looping, compare) before servicing the higher-priority trigger. 11b - Reserved

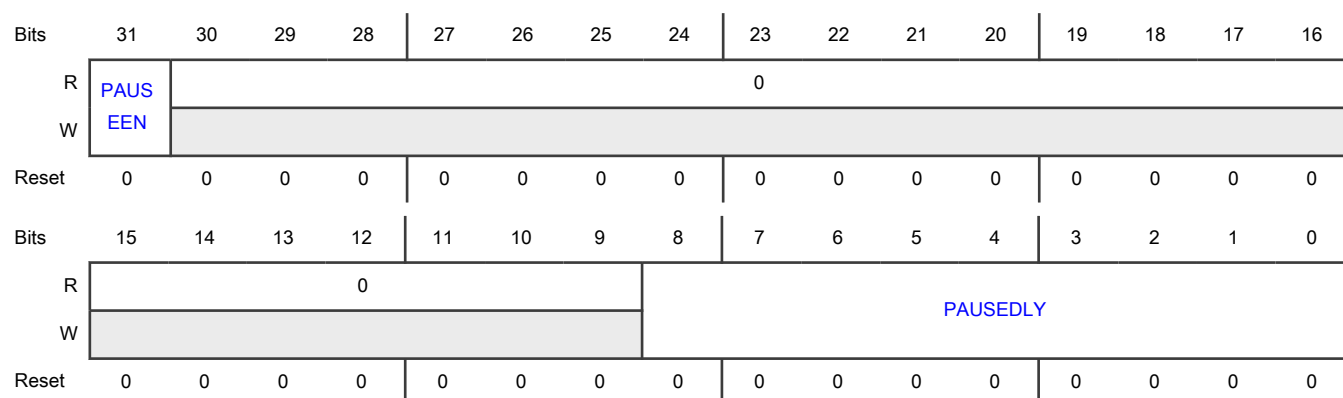
58.6.9 Pause Register (PAUSE)

Offset

Register	Offset
PAUSE	24h

Function

Controls the optional inserted delay between conversions. When CTRL[ADCEN] = 1, this register should not be modified.

Diagram**Fields**

Field	Function
31 PAUSEEN	Pause Enable Enables the ADC pausing function. When enabled, a programmable delay is inserted during command execution sequencing: - Between LOOP iterations. - Between commands in a sequence. - Between conversions, when a command is executing in compare-until-true configuration. NOTE CMDHn[WAIT_TRIG] and PAUSE are mutually exclusive. CMDHn[WAIT_TRIG] takes priority over PAUSE between commands when both are enabled. 0b - Disabled 1b - Enabled
30-9 —	Reserved
8-0 PAUSEDLY	Pause Delay Controls the duration of pauses during command execution sequencing when PAUSE[PAUSEEN] = 1. The pause delay is a count of (PAUSEDLY * 4) ADCK cycles.

58.6.10 Software Trigger Register (SWTRIG)**Offset**

Register	Offset
SWTRIG	34h

Function

Initiates software-triggered conversions when written to. Writes to this register are ignored while CTRL[ADCEN] = 0. There is an approximately three ADC-clock-cycle synchronization delay between asserting CTRL[ADCEN] and when SWTRIG can be accepted.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SWT15	SWT14	SWT13	SWT12	SWT11	SWT10	SWT9	SWT8	SWT7	SWT6	SWT5	SWT4	SWT3	SWT2	SWT1	SWT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	Reserved
15 SWT15	Software Trigger 15 Write 1 to SWT15 generates a trigger 15 event. Writing 1 to SWT15 is ignored while the trigger 15 event is being serviced or is pending. 0b - No trigger 15 event generated. 1b - Trigger 15 event generated.
14 SWT14	Software Trigger 14 Write 1 to SWT14 generates a trigger 14 event. Writing 1 to SWT14 is ignored while the trigger 14 event is being serviced or is pending. 0b - No trigger 14 event generated. 1b - Trigger 14 event generated.
13 SWT13	Software Trigger 13 Write 1 to SWT13 generates a trigger 13 event. Writing 1 to SWT13 is ignored while the trigger 13 event is being serviced or is pending. 0b - No trigger 13 event generated. 1b - Trigger 13 event generated.
12 SWT12	Software Trigger 12

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Write 1 to SWT12 generates a trigger 12 event. Writing 1 to SWT12 is ignored while the trigger 12 event is being serviced or is pending. 0b - No trigger 12 event generated. 1b - Trigger 12 event generated.
11 SWT11	Software Trigger 11 Write 1 to SWT11 generates a trigger 11 event. Writing 1 to SWT11 is ignored while the trigger 11 event is being serviced or is pending. 0b - No trigger 11 event generated. 1b - Trigger 11 event generated.
10 SWT10	Software Trigger 10 Write 1 to SWT10 generates a trigger 10 event. Writing 1 to SWT10 is ignored while the trigger 10 event is being serviced or is pending. 0b - No trigger 10 event generated. 1b - Trigger 10 event generated.
9 SWT9	Software Trigger 9 Write 1 to SWT9 generates a trigger 9 event. Writing 1 to SWT9 is ignored while the trigger 9 event is being serviced or is pending. 0b - No trigger 9 event generated. 1b - Trigger 9 event generated.
8 SWT8	Software Trigger 8 Write 1 to SWT8 generates a trigger 8 event. Writing 1 to SWT8 is ignored while the trigger 8 event is being serviced or is pending. 0b - No trigger 8 event generated. 1b - Trigger 8 event generated.
7 SWT7	Software Trigger 7 Write 1 to SWT7 generates a trigger 7 event. Writing 1 to SWT7 is ignored while the trigger 7 event is being serviced or is pending. 0b - No trigger 7 event generated. 1b - Trigger 7 event generated.
6 SWT6	Software Trigger 6 Write 1 to SWT6 generates a trigger 6 event. Writing 1 to SWT6 is ignored while the trigger 6 event is being serviced or is pending. 0b - No trigger 6 event generated. 1b - Trigger 6 event generated.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 SWT5	Software Trigger 5 Write 1 to SWT5 generates a trigger 5 event. Writing 1 to SWT5 is ignored while the trigger 5 event is being serviced or is pending. 0b - No trigger 5 event generated. 1b - Trigger 5 event generated.
4 SWT4	Software Trigger 4 Write 1 to SWT4 generates a trigger 4 event. Writing 1 to SWT4 is ignored while the trigger 4 event is being serviced or is pending. 0b - No trigger 4 event generated. 1b - Trigger 4 event generated.
3 SWT3	Software Trigger 3 Write 1 to SWT3 generates a trigger 3 event. Writing 1 to SWT3 is ignored while the trigger 3 event is being serviced or is pending. 0b - No trigger 3 event generated. 1b - Trigger 3 event generated.
2 SWT2	Software Trigger 2 Write 1 to SWT2 generates a trigger 2 event. Writing 1 to SWT2 is ignored while the trigger 2 event is being serviced or is pending. 0b - No trigger 2 event generated. 1b - Trigger 2 event generated.
1 SWT1	Software Trigger 1 Write 1 to SWT1 generates a trigger 1 event. Writing 1 to SWT1 is ignored while the trigger 1 event is being serviced or is pending. 0b - No trigger 1 event generated. 1b - Trigger 1 event generated.
0 SWT0	Software Trigger 0 Generates a trigger 0 event. Writing 1 to this field is ignored while the trigger 0 event is being serviced or is pending. 0b - No trigger 0 event generated. 1b - Trigger 0 event generated.

58.6.11 Trigger Status Register (TSTAT)

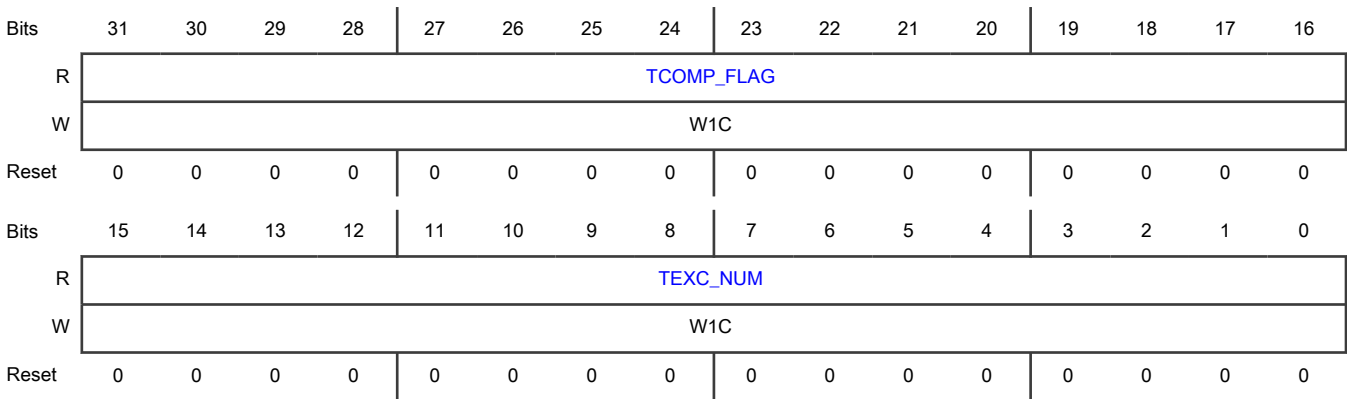
Offset

Register	Offset
TSTAT	38h

Function

Contains status flags to indicate when trigger sequences have been completed or interrupted by a high-priority trigger exception. Each field in this register is set by hardware and cleared by software.

Diagram



Fields

Field	Function
31-16 TCOMP_FLAG	Trigger Completion Flag Indicates which triggers have been completed. Each bit in this field corresponds to a trigger. When a trigger sequence has completed, the corresponding bit in TCOMP_FLAG is set. For example, if trigger 3 has been completed, then bit 19 is set in the TSTAT register. This register is active only if the corresponding bit in IE[TCOMP_IE] = 1. NOTE A synchronization delay may be added to the end of the final conversion in a sequence, prior to this flag being set. This delay can range from two to four ADC CLK cycles. 0000_0000_0000_0000b - No triggers have been completed. Trigger completion interrupts are disabled. 0000_0000_0000_0001b - Trigger 0 has been completed and trigger 0 has enabled completion interrupts. 0000_0000_0000_0010b - Trigger 1 has been completed and trigger 1 has enabled completion interrupts. 0000_0000_0000_0011b-1111_1111_1111_1110b - Associated trigger sequence has completed and has enabled completion interrupts.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	1111_1111_1111_1111b - Every trigger sequence has been completed and every trigger has enabled completion interrupts.
15-0 TEXC_NUM	Trigger Exception Number Indicates which triggers have been interrupted by exceptions. Each bit in this field corresponds to a trigger. When the corresponding trigger sequence is interrupted by a high-priority trigger exception, the corresponding bit is set. For example, if trigger 3 has been interrupted, then bit 3 is set in this register. This register is active regardless of the value in IE[TEXC_IE]. 0000_0000_0000_0000b - No triggers have been interrupted by a high-priority exception. Or CFG[TRES] = 1. 0000_0000_0000_0001b - Trigger 0 has been interrupted by a high-priority exception. 0000_0000_0000_0010b - Trigger 1 has been interrupted by a high-priority exception. 0000_0000_0000_0011b-1111_1111_1111_1110b - Associated trigger sequence has interrupted by a high-priority exception. 1111_1111_1111_1111b - Every trigger sequence has been interrupted by a high-priority exception.

58.6.12 Trigger Control Register (TCTRL0 - TCTRL15)

Offset

For a = 0 to 15:

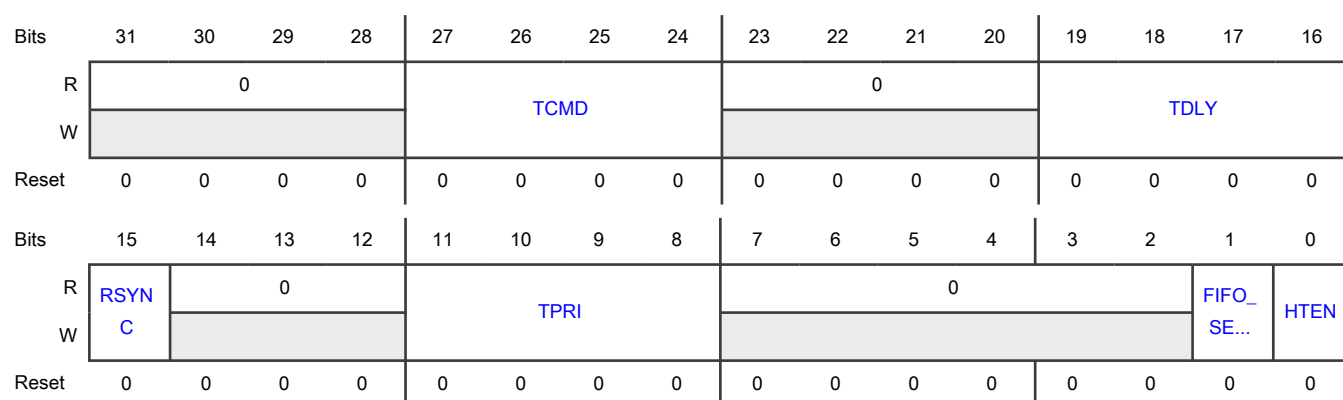
Register	Offset
TCTRLa	A0h + (a × 4h)

Function

Implements control fields associated with each trigger source. When ADC is executing commands, only one TCTRLn register controls ADC conversions.

Do not update the controlling TCTRLn register while ADC is active. Writing to a TCTRLn register while that trigger control register controls the ADC operation may cause unpredictable behavior.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 TCMD	Trigger Command Select Selects the command from the command buffer to execute upon detection of the associated trigger event. When a higher-priority trigger is received while ADC is converting, the command in this register associated with the new trigger is executed. This execution is done under the control of CFG[TPRCTRL]. See Trigger detect and command execution for details about the relationship between CFG[TPRCTRL] and TCMD. 0000b - Not a valid selection from the command buffer. Trigger event is ignored. 0001b - CMD1 0010b-1110b - Corresponding CMD is executed 1111b - CMD15
23-20 —	Reserved
19-16 TDLY	Trigger Delay Select Selects the trigger delay duration for the start of servicing a trigger event. Each trigger source has an associated programmable delay prior to beginning an initial conversion. When this field is 0, no delay is incurred. When this field is set to a non-zero value, the duration of the delay is 2^{TDLY} ADCK cycles.
15 RSYNC	Trigger Resync Enables trigger resync. If 1, this trigger source is used as a resync trigger. A resync trigger can be configured to restart or abort a running trigger sequence (the target). Each resync trigger is configured to have a resync target. When a resync trigger is asserted: The target sequence is aborted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	- The target FIFO destinations are cleared. - The target sequence can be restarted depending on CFG[TRES]. Multiple conditions must be met for a resync trigger to execute properly: - The resync trigger must have higher priority than the resync target. - The resync trigger TCTRLn[RSYNC] must be set to 1b. - The resync target, specified by TCTRLn[TCMD], must be executing when the resync trigger is asserted. See Resync functionality for more information. 0b - Disable 1b - Enable
14-12 —	Reserved
11-8 TPRI	Trigger Priority Setting Sets the priority of the associated trigger source. If two or more triggers have the same priority level, the lower-order trigger event has the higher priority. If Trigger 0 and Trigger 1 are pending triggers and TCTRL0[TPRI] is configured the same as TCTRL1[TPRI], the Trigger 0 command is serviced first. 0000b - Highest priority, Level 1 0001b-1110b - Set to corresponding priority level. 1111b - Lowest priority, Level 16
7-2 —	Reserved
1 FIFO_SEL_A	SAR Result Destination For Channel A Indicates the FIFO to which SAR results are written for Channel A, for all single-ended and differential conversions. Conversion results are stored in the FIFO number specified, under the following conditions: - Single-ended mode: CMDLn[DIFF] = 0b. - Differential mode: CMDLn[DIFF] = 1b. 0b - FIFO 0 1b - FIFO 1
0 HTEN	Trigger Enable Enables hardware trigger source to initiate conversion on the rising edge of the input trigger source. NOTE Enabling the hardware trigger does not disable software triggers.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - Disabled 1b - Enabled

58.6.13 FIFO Control Register (FCTRL0 - FCTRL1)

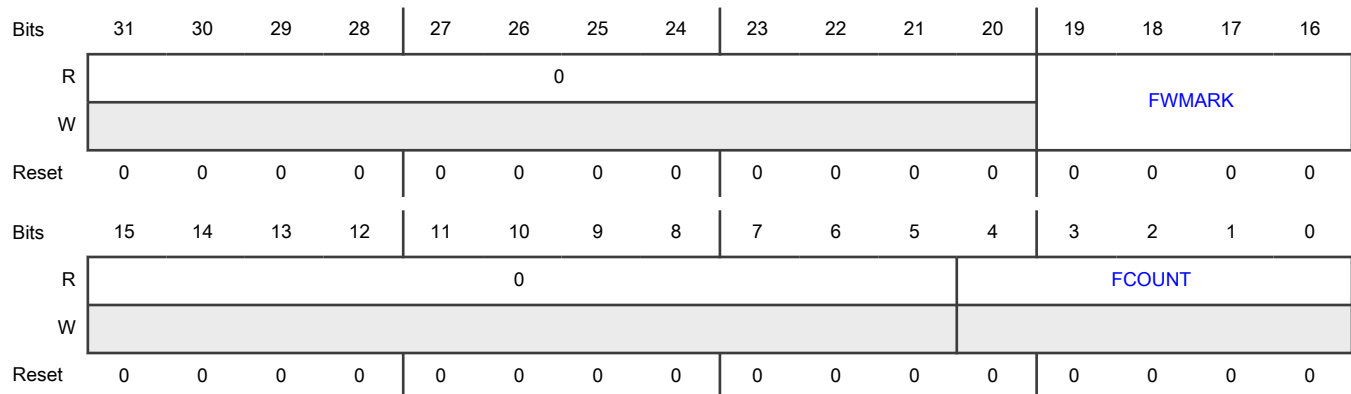
Offset

Register	Offset
FCTRL0	E0h
FCTRL1	E4h

Function

Contains control and status fields for each FIFO in the design. A programmable watermark can be set for each FIFO, which can be used to trigger an interrupt. In addition, the number of entries stored in each FIFO can be monitored by reading FCTRLn[FCOUNT].

Diagram



Fields

Field	Function
31-20 —	Reserved
19-16 FWMARK	Watermark Level Selection Selects the storage threshold for the ADC Result FIFO. When the number of data words stored in the FIFO is greater than this value, the STAT[RDY0] flag is asserted. When IE[FWMIEn] = 1, an interrupt request is generated. When DE[FWMDEn] = 1, a DMA request is generated.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-5 —	Reserved
4-0 FCOUNT	Result FIFO Counter Indicates the number of data words stored in the result FIFO. This value may be used with PARAM[FIFOSIZE] to calculate how much room is left in the result FIFO. This field is incremented with each storage of new data into the result FIFO and decrements with each read of the result FIFO. The FIFO is reset by writing to CTRL[RSTFIFOn] , which initializes FCTRLn[FCOUNT] to 0h.

58.6.14 Command Low Buffer Register (CMDL1 - CMDL15)

Offset

For a = 1 to 15:

Register	Offset
CMDLa	F8h + (a × 8h)

Function

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and varying conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of [TCTRLn\[TCMD\]](#). When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
```

```
unsigned int CMDL[15];
```

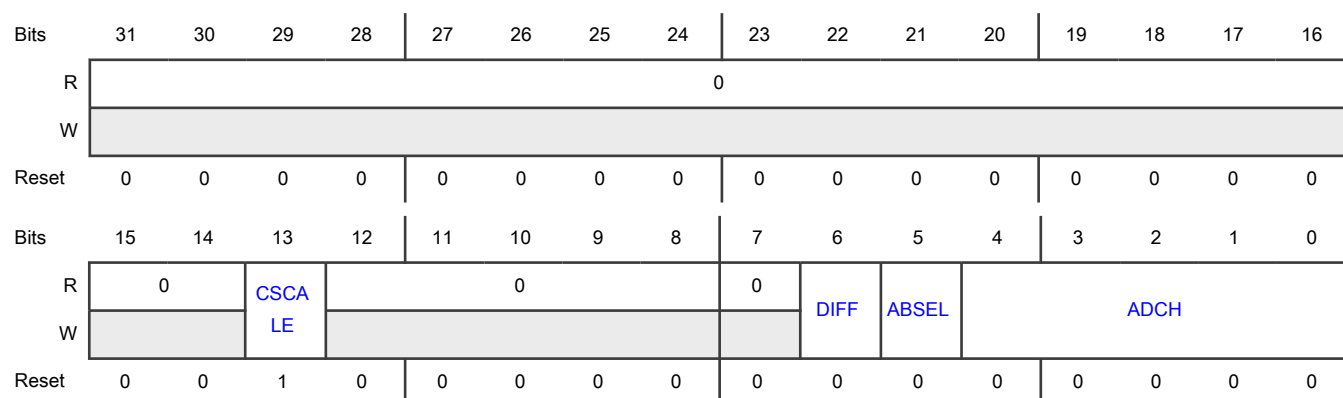
and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

The CMDLn[ADCH], CMDLn[DIFF], and CMDLn[ABSEL] fields control selection of paired or individual input channels. Each ADC command independently selects a channel. Each ADCH channel selection has an associated A-side and B-side input. Each ADCH pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

Some input channels are from on-chip sources such as temperature sensors and reference voltage sources and may only be connected to individual instances of ADC. Some input channel options in the field descriptions may not be available for your device. See the chip-specific information for the channels supported on this device.

Diagram



Fields

Field	Function
31-16 —	Reserved
15-14 —	Reserved
13 CSCALE	Channel Scale Reduces the selected ADC analog channel input voltage level by a factor. When selecting a CSCALE value, consider the maximum voltage on the ADC channel input. Your selection can ensure that the reducing factor always results in a voltage level at or below the VREFH reference. This reducing capability allows conversion of analog inputs higher than VREFH. A-side and B-side channel inputs are both scaled using the CSCALE field. 0b - Scale selected analog channel (factor of 30/64) 1b - Full-scale (factor of 1)
12-8 —	Reserved
7 —	Reserved
6 DIFF	Differential Mode Enable Configures ADC to operate in differential mode or dual-single-ended mode. 0b - Dual-single-ended mode 1b - Differential mode
5 ABSEL	A-side or B-side Select Selects which channel to convert when <code>CMDLn[DIFF] = 0b</code>. Configures the ADC result to be (CHnA - CHnB) or (CHnB - CHnA) when <code>CMDLn[DIFF] = 1b</code>.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - When CMDLn[DIFF] = 0b, the associated A-side channel is converted as single-ended. When CMDLn[DIFF] = 1b, the ADC result is (CHnA - CHnB). 1b - When CMDLn[DIFF] = 0b, the associated B-side channel is converted as single-ended. When CMDLn[DIFF] = 1b, the ADC result is (CHnB - CHnA).
4-0 ADCH	Input Channel Select Selects the input from one of the channel inputs or one of the input pairs, with CMDLn[DIFF] and CMDLn[ABSEL]. Each ADCH channel selection has an associated A-side and B-side input. NOTE Not all pairs should be converted as differential channels. See the chip-specific information for the pin pairings available for differential conversions for your device. NOTE Some input channels are from internal resources such as temperature sensors and band gap voltage sources and may only be connected to individual instances of ADC. Some input channel options in field descriptions might not be available for your device. See the chip-specific information for the ADC channel assignments for your device. 0_0000b - CH0A or CH0B or CH0A/CH0B pair. 0_0001b - CH1A or CH1B or CH1A/CH1B pair. 0_0010b - CH2A or CH2B or CH2A/CH2B pair. 0_0011b - CH3A or CH3B or CH3A/CH3B pair. 0_0100b-1_1101b - Select corresponding channel CHnA or CHnB or CHnA/CHnB pair. 1_1110b - CH30A or CH30B or CH30A/CH30B pair. 1_1111b - CH31A or CH31B or CH31A/CH31B pair.

58.6.15 Command High Buffer Register (CMDH1 - CMDH4)

Offset

Register	Offset
CMDH1	104h
CMDH2	10Ch
CMDH3	114h
CMDH4	11Ch

Function

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of [TCTRLn\[TCMD\]](#). When

ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

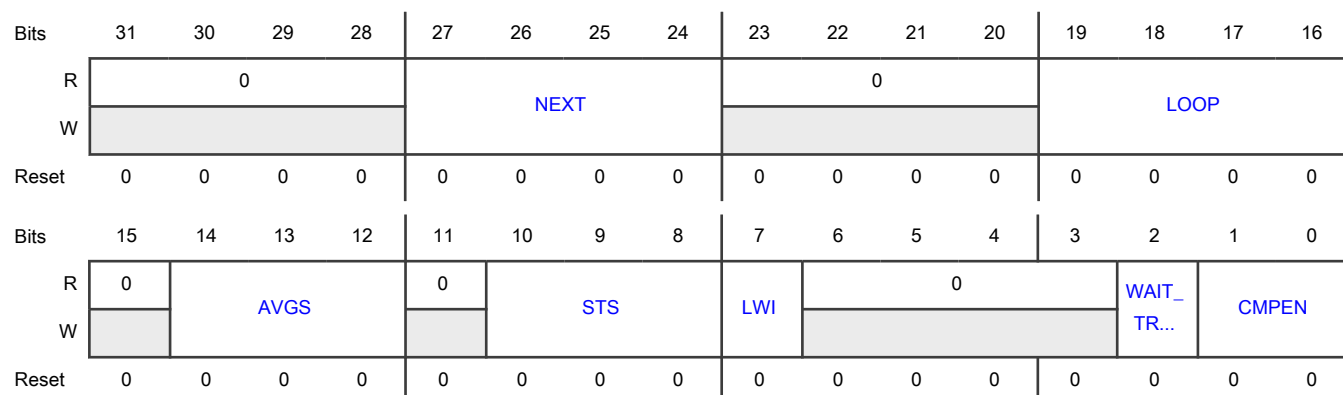
NOTE

The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
```

```
unsigned int CMDL[15];
```

and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

Diagram**Fields**

Field	Function
31-28 —	Reserved
27-24 NEXT	Next Command Select Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order. The order of execution is strictly controlled by this field. For example, a sequence of commands could be CMD2 to CMD1 to CMD3. Unending circular command execution can be configured by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0h causes conversions to terminate at the completion of the command. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or sequence of commands) completes. 0000b - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger. 0001b - CMD1 0010b-1110b - Select corresponding CMD command buffer register as next command 1111b - CMD15

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-20 —	Reserved
19-16 LOOP	Loop Count Select Selects the number of times that this command executes (and stores conversion result to RESFIFO) before finishing and transitioning to the next command or idle state. CMDHn[LWI] controls whether a single channel is converted on each iteration or an automatic channel increment results in channel scanning functionality. 0000b - Looping not enabled. Command executes one time. 0001b - Loop one time. Command executes two times. 0010b - Loop two times. Command executes three times. 0011b-1110b - Loop corresponding number of times. Command executes LOOP + 1 times. 1111b - Loop 15 times. Command executes 16 times.
15 —	Reserved
14-12 AVGS	Hardware Average Select Selects how many ADC conversions are averaged to create the ADC result (2^{AVGS}). An internal storage buffer captures temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for usage of AVGS, LOOP, and NEXT fields in command execution sequencing. 000b - Single conversion 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 128
11 —	Reserved
10-8 STS	Sample Time Select Selects the total sampling time. When this field contains a non-zero value, the sample time is $(3.5 + 2^{STS})$ ADCK cycles. The shortest sample time maximizes conversion speed for lower-impedance inputs. Extending sample time allows higher-impedance inputs to be sampled accurately. Longer sample times can lower overall power consumption when command looping and sequencing are configured and high conversion rates are not required.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000b - Minimum sample time of 3.5 ADCK cycles. 001b - 5.5 ADCK cycles. ($3.5 + 2^1$ ADCK cycles) 010b - 7.5 ADCK cycles. ($3.5 + 2^2$ ADCK cycles) 011b - 11.5 ADCK cycles. ($3.5 + 2^3$ ADCK cycles) 100b - 19.5 ADCK cycles. ($3.5 + 2^4$ ADCK cycles) 101b - 35.5 ADCK cycles. ($3.5 + 2^5$ ADCK cycles) 110b - 67.5 ADCK cycles. ($3.5 + 2^6$ ADCK cycles) 111b - 131.5 ADCK cycles. ($3.5 + 2^7$ ADCK cycles)
7 LWI	Loop with Increment Enables automatic channel incrementing. When 0, the LOOP field selects the number of times the selected channel is converted consecutively. When 1, automatic channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution. Example 1: LOOP = 8h, LWI = 0b, ABSEL = 0b, CMDLn[ADCH] = Dh. Convert on channel 13A 9 times. Example 2: LOOP = 8h, LWI = 1b, ABSEL = 0b, CMDLn[ADCH] = Dh. Run channels 13A-21A each one time. Maximum channel scanning using a single command buffer is defined by the maximum value of the LOOP field (16). 0b - Disabled 1b - Enabled
6-3 —	Reserved
2 WAIT_TRIG	Wait for Trigger Assertion Before Execution Selects whether commands execute automatically or a trigger must be received before execution. When 0, wait states are added before the command until the active trigger is asserted again. When WAIT_TRIG = 0, each command is automatically executed when called. 0b - Command executes automatically. 1b - Active trigger must be asserted again before executing this command.
1-0 CMPEN	Compare Function Enable Enables the automatic compare function. Selects whether to store only when the comparison operation is true, following ADC channel input sampling, conversion, and averaging. When the compare function is enabled, the conversion result is compared to the compare value registers (CVn[CVH] and CVn[CVL]). See Compare Function for details about the options for command sequencing related to the compare function. NOTE Not all Command Buffers have implemented the CMPEN field. The CMPEN field is only available in Command Buffers that have a corresponding Compare Value register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	00b - Disabled
	01b - Reserved
	10b - Enabled. Store on true.
	11b - Enabled. Repeat channel acquisition (sample, convert, and compare) until true.

58.6.16 Command High Buffer Register (CMDH5 - CMDH15)

Offset

For a = 5 to 15:

Register	Offset
CMDHa	FCh + (a × 8h)

Function

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of [TCTRLn\[TCMD\]](#). When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

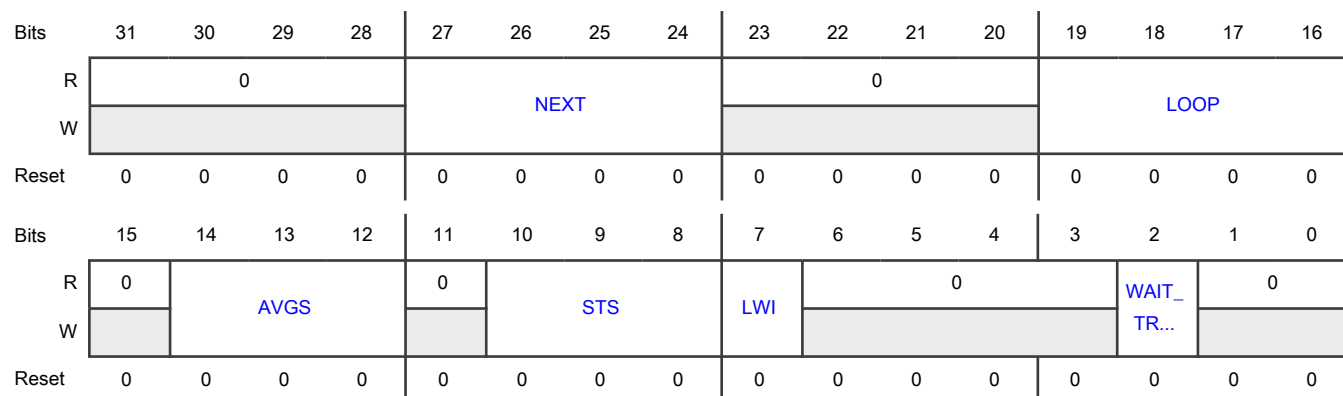
The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
```

```
unsigned int CMDL[15];
```

and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

Diagram



Fields

Field	Function
31-28 —	Reserved
27-24 NEXT	Next Command Select Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order. The order of execution is strictly controlled by this field. For example, a sequence of commands could be CMD2 to CMD1 to CMD3. Unending circular command execution can be configured by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0h causes conversions to terminate at the completion of the command. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or sequence of commands) completes. 0000b - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger. 0001b - CMD1 0010b-1110b - Select corresponding CMD command buffer register as next command 1111b - CMD15
23-20 —	Reserved
19-16 LOOP	Loop Count Select Selects the number of times that this command executes (and stores conversion result to RESFIFO) before finishing and transitioning to the next command or idle state. CMDHn[LWI] controls whether a single channel is converted on each iteration or an automatic channel increment results in channel scanning functionality. 0000b - Looping not enabled. Command executes one time. 0001b - Loop one time. Command executes two times. 0010b - Loop two times. Command executes three times. 0011b-1110b - Loop corresponding number of times. Command executes LOOP + 1 times. 1111b - Loop 15 times. Command executes 16 times.
15 —	Reserved
14-12 AVGS	Hardware Average Select Selects how many ADC conversions are averaged to create the ADC result (2^{AVGS}). An internal storage buffer captures temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for usage of AVGS, LOOP, and NEXT fields in command execution sequencing.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	000b - Single conversion 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 128
11 —	Reserved
10-8 STS	Sample Time Select Selects the total sampling time. When this field contains a non-zero value, the sample time is $(3.5 + 2^{\text{STS}})$ ADCK cycles. The shortest sample time maximizes conversion speed for lower-impedance inputs. Extending sample time allows higher-impedance inputs to be sampled accurately. Longer sample times can lower overall power consumption when command looping and sequencing are configured and high conversion rates are not required. 000b - Minimum sample time of 3.5 ADCK cycles. 001b - 5.5 ADCK cycles. $(3.5 + 2^1 \text{ ADCK cycles})$ 010b - 7.5 ADCK cycles. $(3.5 + 2^2 \text{ ADCK cycles})$ 011b - 11.5 ADCK cycles. $(3.5 + 2^3 \text{ ADCK cycles})$ 100b - 19.5 ADCK cycles. $(3.5 + 2^4 \text{ ADCK cycles})$ 101b - 35.5 ADCK cycles. $(3.5 + 2^5 \text{ ADCK cycles})$ 110b - 67.5 ADCK cycles. $(3.5 + 2^6 \text{ ADCK cycles})$ 111b - 131.5 ADCK cycles. $(3.5 + 2^7 \text{ ADCK cycles})$
7 LWI	Loop with Increment Enables automatic channel incrementing. When 0, the LOOP field selects the number of times the selected channel is converted consecutively. When 1, automatic channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution. Example 1: LOOP = 8h, LWI = 0b, ABSEL = 0b, CMDLn[ADCH] = Dh. Convert on channel 13A 9 times. Example 2: LOOP = 8h, LWI = 1b, ABSEL = 0b, CMDLn[ADCH] = Dh. Run channels 13A-21A each one time. Maximum channel scanning using a single command buffer is defined by the maximum value of the LOOP field (16). 0b - Disabled 1b - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
6-3 —	Reserved
2 WAIT_TRIG	Wait for Trigger Assertion Before Execution Selects whether commands execute automatically or a trigger must be received before execution. When 0, wait states are added before the command until the active trigger is asserted again. When WAIT_TRIG = 0, each command is automatically executed when called. 0b - Command executes automatically. 1b - Active trigger must be asserted again before executing this command.
1-0 —	Reserved

58.6.17 Compare Value Register (CV1 - CV4)

Offset

Register	Offset
CV1	200h
CV2	204h
CV3	208h
CV4	20Ch

Function

Contains values used to compare the conversion result when the compare function is enabled. This register is formatted like the D field in [Data Result FIFO Register \(RESFIFO0 - RESFIFO1\)](#), which has in different definitions for bit position and value format in different modes. There is a direct association of each compare value register to a specific command buffer register. For example, CV1 is only used during execution of CMD1 command.

NOTE

Not all command buffers have an associated compare value register. The compare function is only available on command buffers that have a corresponding compare value register.

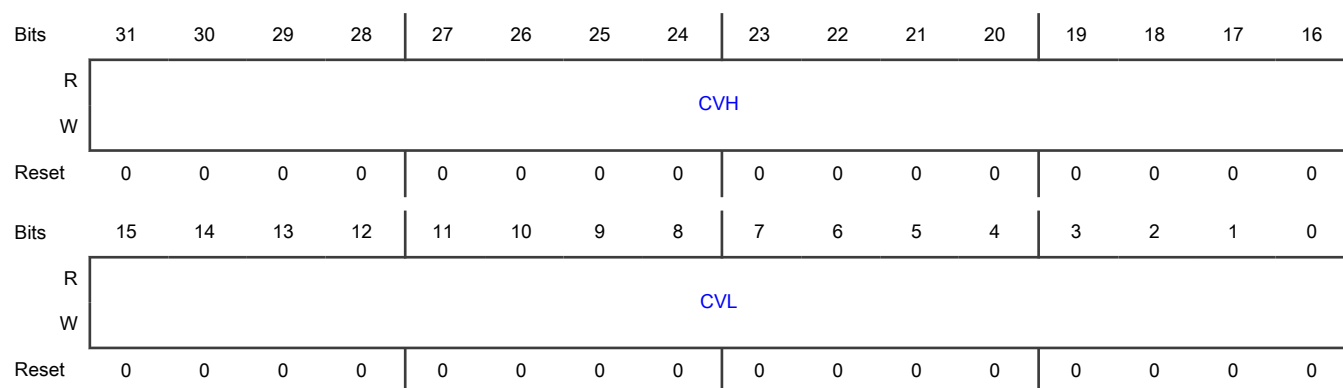
When ADC is executing commands, do not update the CV_n register associated with the active command (CMD_n). Writes to associated CV_n register during this time may result in unpredictable behavior.

NOTE

The 4 compare value registers are numbered [CV1] through [CV4]. In NXP-supplied header files, these registers are likely to be defined as a 4-element array indexed from 0. For example, the type declaration would be:

```
unsigned int CV[4];
```

and software would access ADC0 CV1 as `ADC0->CV[0]` and ADC0 CV4 as `ADC0->CV[3]`.

Diagram**Fields**

Field	Function
31-16 CVH	Compare Value High Determines the high compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values • Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVH usage.
15-0 CVL	Compare Value Low Determines the low compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values • Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVL usage.

58.6.18 Data Result FIFO Register (RESFIFO0 - RESFIFO1)

Offset

Register	Offset
RESFIFO0	300h
RESFIFO1	304h

Function

Stores the data result of ADC conversions in a 16-entry FIFO. Several tag fields of source command and trigger information are stored with the data. [FCTRLn\[FCOUNT\]](#) indicates how many valid data words are stored in the RESFIFO. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements [FCTRLn\[FCOUNT\]](#). The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to [CTRL\[RSTFIFOn\]](#).

The following table describes the format of data in the result FIFO in different modes of operation. The sign bit is the MSB in signed 2's complement modes. For example, when configured for 12-bit single-ended mode, D[15] and D[2:0] become 0. When configured for 13-bit differential mode, D[15] is the sign bit, and D[2:0] becomes 0.

Table 600. Data result register format description

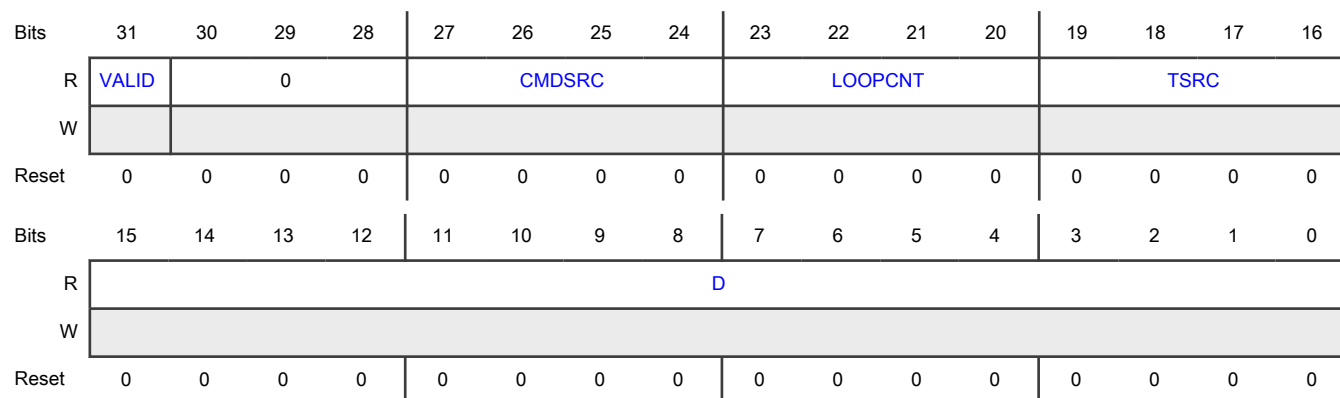
Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
13-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Signed 2's complement, left justified, zero extended
12-bit single-ended	0	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Unsigned, zero in D[15] and D[2:0]

NOTE

S: Sign bit;

D: Data, 2's complement data when indicated

Diagram



Fields

Field	Function
31 VALID	FIFO Entry is Valid Indicates whether the FIFO entry is valid, which determines what happens to reads from RESFIFO. 0b - FIFO is empty. Discard any read from RESFIFO. 1b - FIFO contains data. FIFO record read from RESFIFO is valid.
30-28 —	Reserved
27-24 CMDSRC	Command Buffer Source Indicates the executed command buffer that generated this result. 0000b - Not a valid value CMDSRC value for a data word in RESFIFO. 0h is only found in the initial FIFO state, prior to the storage of an ADC conversion result into a RESFIFO buffer. 0001b - CMD1 0010b-1110b - Corresponding command buffer used as control settings for this conversion. 1111b - CMD15
23-20 LOOPCNT	Loop Count Value Indicates the loop count value during the command that generated this result. When CMDHn[LOOP] is non-zero, results are stored multiple times during command execution at the loop boundary. 0000b - Result is from initial conversion in command. 0001b - Result is from second conversion in command. 0010b-1110b - Result is from (LOOPCNT + 1) conversion in command. 1111b - Result is from 16th conversion in command.
19-16 TSRC	Trigger Source Indicates the trigger source that initiated a conversion and generated this result. When multiple commands are chained together using CMDHn[NEXT], this field indicates the trigger source that started the command sequence. 0000b - Trigger source 0 0001b - Trigger source 1 0010b-1110b - Corresponding trigger source initiated this conversion. 1111b - Trigger source 15
15-0 D	Data Result Contains the result of an ADC conversion. The formatting for the data in D is summarized in Table 600.

Chapter 59

Analog Comparator (ACMP)

59.1 Chip-specific ACMP information

Table 601. Reference links to related information

Topic	Related module	Reference
Full description	ACMP	ACMP
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

There is one CMP module and control logic that is accessible only by the M33 CPU. Inputs to comparator mux include a selection of pins and the ACMP's internal DAC reference.

Initial configuration of a comparator peripheral is accomplished as follows:

- In the PSCCTL1 register in CLKCTL0 register section, set the ACMP0 bit to enable the clock to the register interface.
- In the ACMP0FCLKSEL and ACMP0FCLKDIV registers in CLKCTL1 register section, configure the comparator function clock.
- Clear the comparator reset using the PRSTCTL1 register in RSTCTL0 register section.
- Enable the ACMP analog function by clearing the ACMP_PD bit in the PDRUNCFG0 register in SYSCTL0 register section.
- The comparator interrupt is connected to an interrupt slot in the NVIC.

NOTE

The WINDOW/SAMPLE signal described in some comparator modes is not used on this chip, the signal is tied low.

59.1.1 Module instances

This device has one instance of the ACMP module, ACMP0.

59.2 Introduction

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 8-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The DAC is a 256-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. The 256-tap resistor ladder network divides the supply reference V_{in} into 256 voltage levels. A 8-bit digital signal input selects the output voltage level, which varies from 0 to $(255/256)V_{in}$. V_{in} can be selected from two voltage sources, V_{in1} and V_{in2} .

59.3 Features

The following subsections list the features of the CMP, the DAC, and the ANMUX.

59.3.1 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control
- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
 - Sampled
 - Windowed, which is ideal for certain PWM zero-crossing-detection applications
 - Digitally filtered:
 - Filter can be bypassed
 - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
 - Shorter propagation delay at the expense of higher power
 - Low power, with longer propagation delay
- DMA transfer support
 - A comparison event can be selected to trigger a DMA transfer
- Functional in all MCU power modes
- The window and filter functions are not available in STOP modes
- The comparator can be triggered by other peripherals to work for only a small fraction of the time
- Programmable round-robin Timer that may be used to save power by keeping the comparator turned off between compares.

59.3.2 8-bit DAC key features

The 8-bit DAC has the following features:

- 8-bit resolution
- Selectable supply reference source
- Configurable low power mode or high speed mode
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

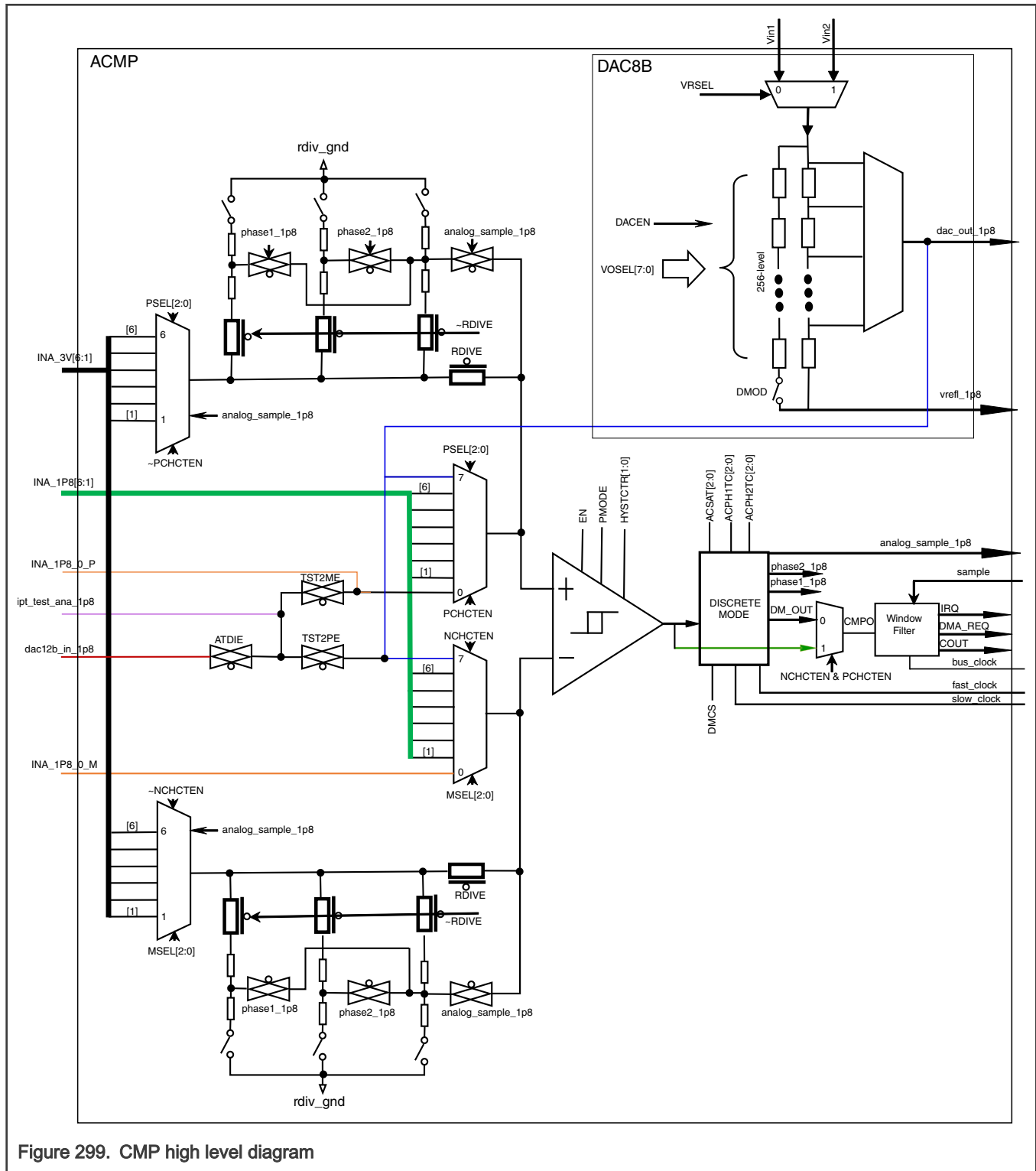
59.3.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel MUXes
- Operational over the entire supply range

59.4 CMP, DAC, and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.



59.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

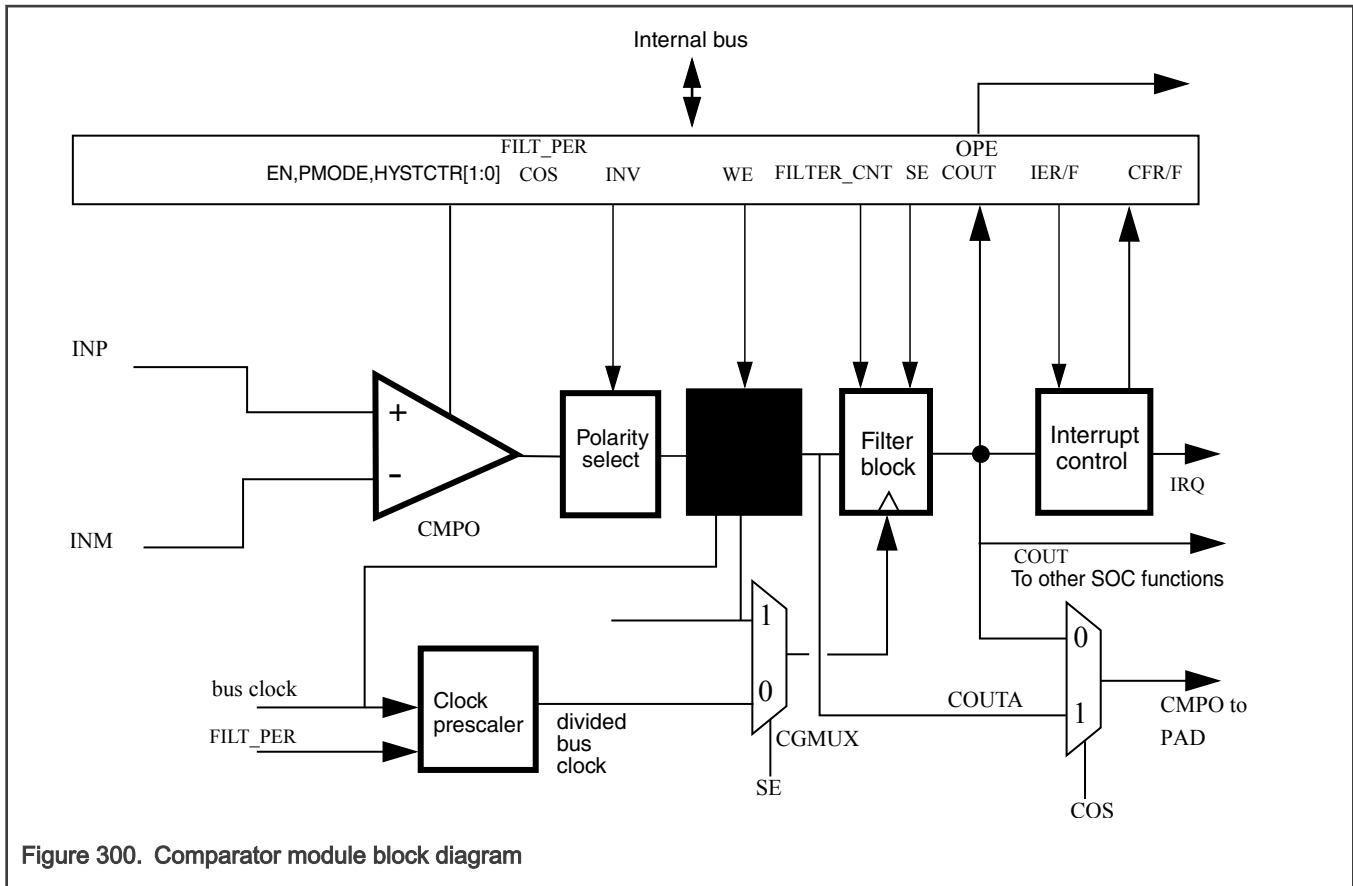


Figure 300. Comparator module block diagram

In the CMP block diagram:

- The Window Control block is bypassed when $C0[WE] = 0$.
- If $C0[WE] = 1$, the comparator output is sampled on every bus clock when $WINDOW=1$ to generate $COUTA$. Sampling does NOT occur when $WINDOW = 0$.
- The Filter block is bypassed when not in use.

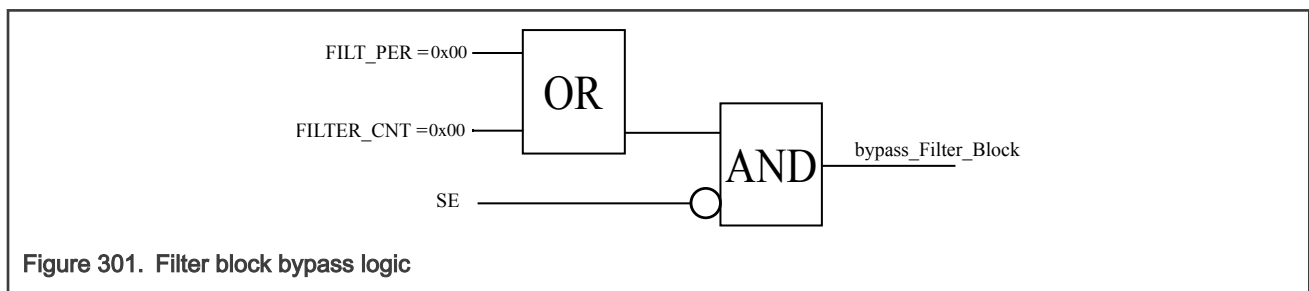


Figure 301. Filter block bypass logic

- The Filter block acts as a simple sampler if the filter is bypassed and $C0[FILTER\_CNT]$ is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and $C0[FILTER\_CNT]$ is set greater than 0x01.
 - If $C0[SE] = 1$, the external SAMPLE input is used as the sampling clock.
 - IF $C0[SE] = 0$, the divided bus clock is used as the sampling clock.
- If enabled, the Filter block will incur up to one bus clock additional latency penalty on $COUT$ due to the fact that $COUT$, which crosses clock domain boundaries, must be resynchronized to the bus clock.
- $C0[WE]$ and $C0[SE]$ are mutually exclusive.
- If enabled, the filter clock and the sample period must be at least 4 times slower than the system clock to the comparator.

59.6 CMP pin descriptions

This section provides the comparator pin descriptions. The external 3V domain inputs INA_3V[6:1] and 1.8V domain INA_1P8[6:1], INA_1P8_0_P, INA_1P8_0_M are muxed by CMP_C1[PSEL] and CMP_C1[MSEL] before going to the positive and negative ports of the comparator respectively.

Table 602. CMP signal descriptions

Signal	Description	I/O
INA_3V[6:1]	The 3V domain analog inputs[6:1]	I
INA_1P8[6:1]	Analog 1.8V input channels [6:1].	I
INA_1P8_0_P	Analog 1.8V input channel 0 positive.	I
INA_1P8_0_m	Analog 1.8V input channel 0 minus.	I

NOTE

It is not recommended to compare 3V domain channel with 1.8V domain channel. If the 3V domain channel's voltage is smaller than 1.8V, in this case, it can be compared with 1.8V domain channel by setting C3[RDIVE] to 0.

59.6.1 External pins

The CMP has two analog inputs: INP and INM. Each of these pins can accept an input voltage that varies across the full operating range of the MCU. If the module is not enabled, each pin can be used as a digital input or output. Consult the specific MCU documentation to determine what functions are shared with these analog inputs.

The user can select either filtered or unfiltered comparator outputs for use on an external I/O pad.

59.7 CMP functional modes

There are three main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CMP_C0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CMP_C0[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CMP_C0[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 603. Comparator sample/filter controls

Mode #	CMP_C0[EN]	CMP_C0[WE]	CMP_C0[SE]	CMP_C0[FILTER_CNT]	CMP_C0[FPR]	Operation
1	0	X	X	X	X	Disabled See the Disabled mode (# 1) .
2A	1	0	0	0x00	X	Continuous Mode See the Continuous mode (#s 2A & 2B) .
2B	1	0	0	X	0x00	
3B	1	0	0	0x01	> 0x00	Sampled, Non-Filtered mode See the Sampled, Non-Filtered mode (#s 3B) .
4B	1	0	0	> 0x01	> 0x00	Sampled, Filtered mode See the Sampled, Filtered mode (#s 4B) .
5B	1	1	0	X	0x00	Windowed mode Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5B) .
8	1	1	1	X	X	Trigger mode In RUN mode Comparator is configured to the modes same with these Modes #5A, #5B, #6 and #7 as elaborated in above rows depended on the CMP_C0[FILT_CNT] and CMP_C0[FPR] settings. In STOP mode, the comparator and possible the 8bit DAC is triggered work periodically and the active channels follow the round robin cycling scheme. See the Trigger mode .
All other combinations of CMP_C0[EN], CMP_C0[WE], CMP_C0[SE], CMP_C0[FILTER_CNT], and CMP_C0[FPR] are illegal.						

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

NOTE

Filtering and sampling settings must be changed only after setting CMP_C0[SE]=0, CMP_C0[FPR] =0 and CMP_C0[FILTER_CNT]=0x00. This resets the filter to a known state.

59.7.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

59.7.2 Continuous mode (#s 2A & 2B)

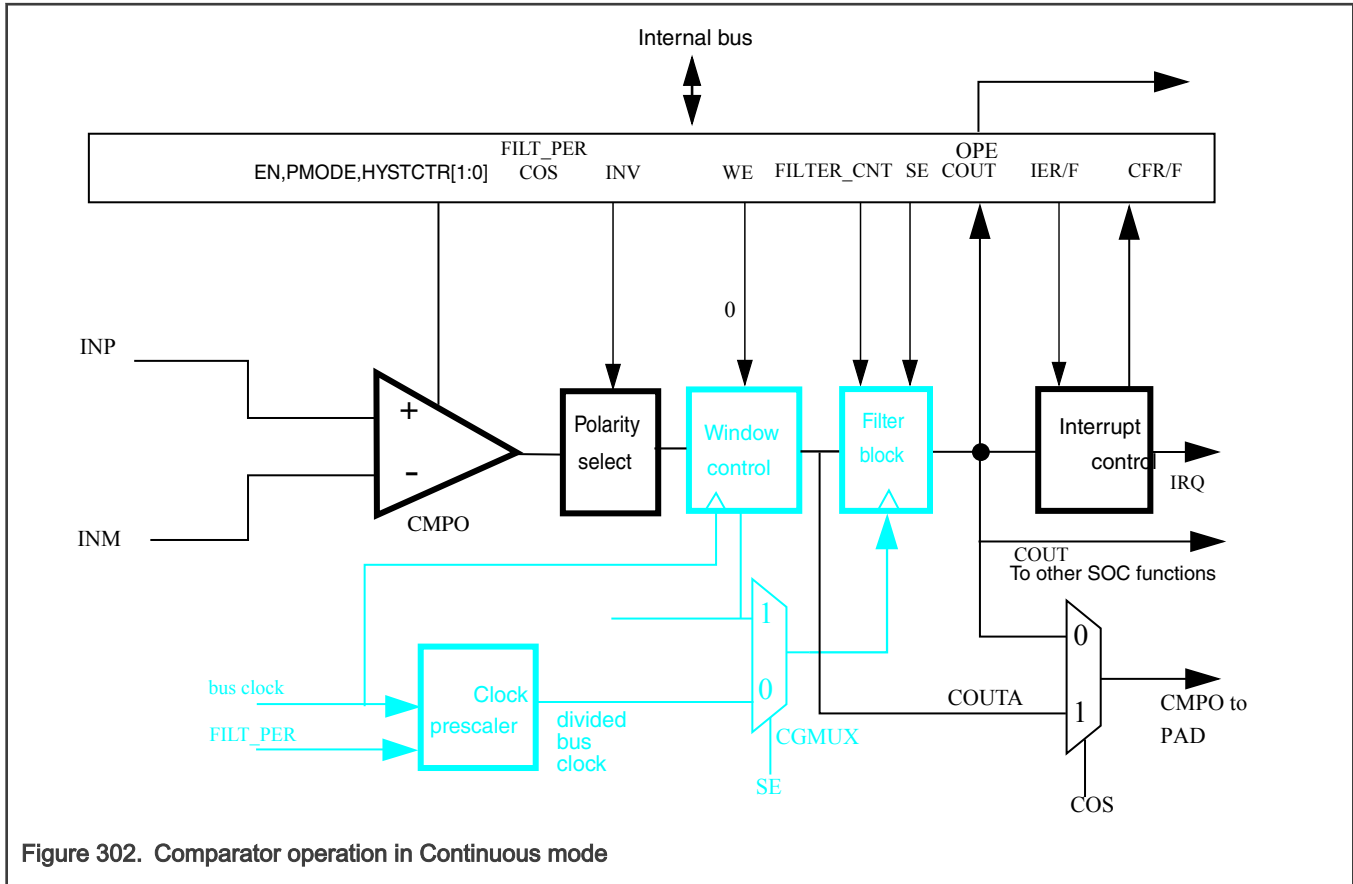


Figure 302. Comparator operation in Continuous mode

NOTE

See the chip configuration section for the source of sample/window input.

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. CMP_C0[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational unlocked mode. COUT and COUTA are identical.

For control configurations that result in disabling the filter block, see [Figure 301](#).

59.7.3 Sampled, Non-Filtered mode (#s 3B)

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unlocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Non-Filtered (# 3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

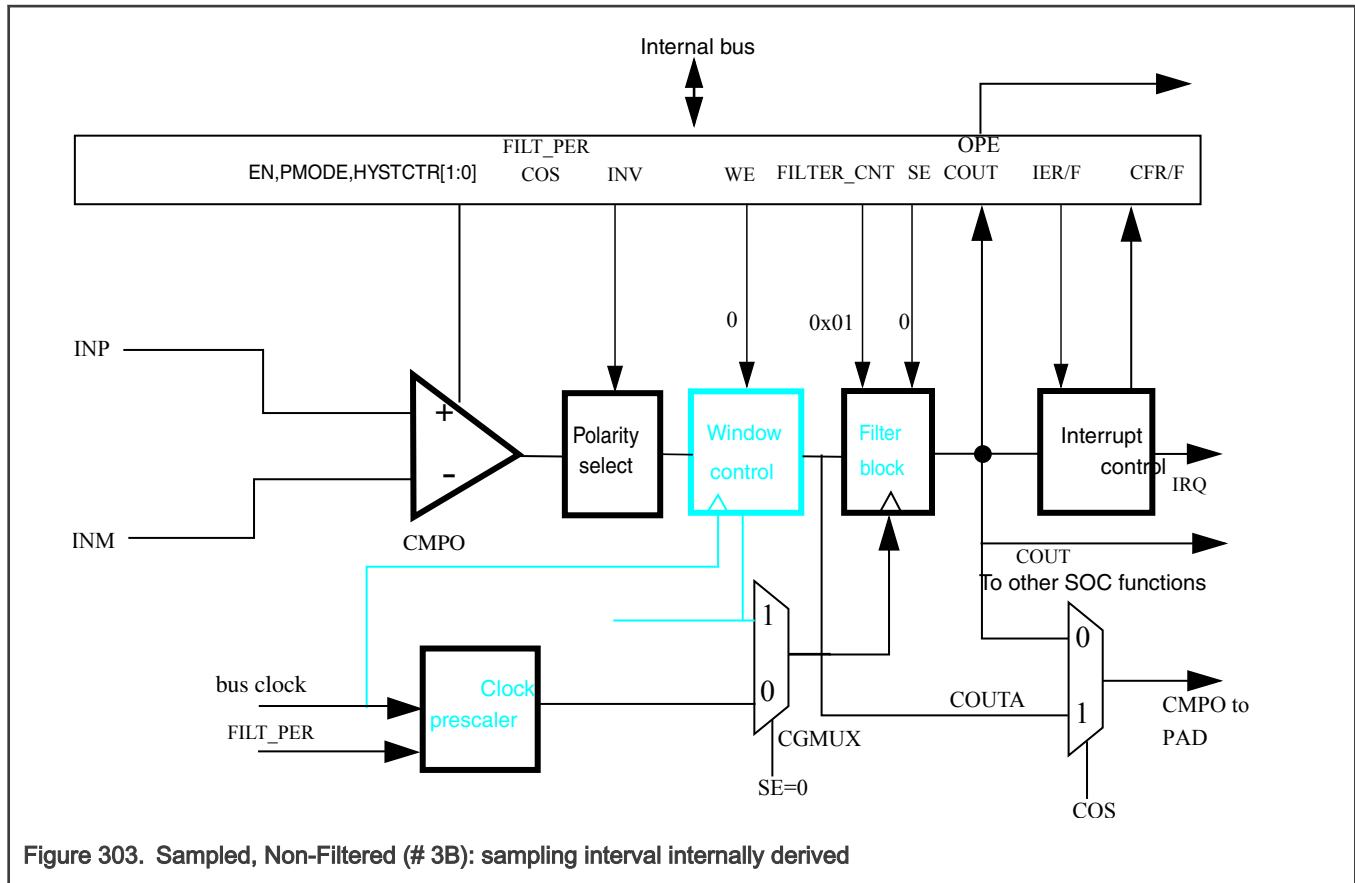
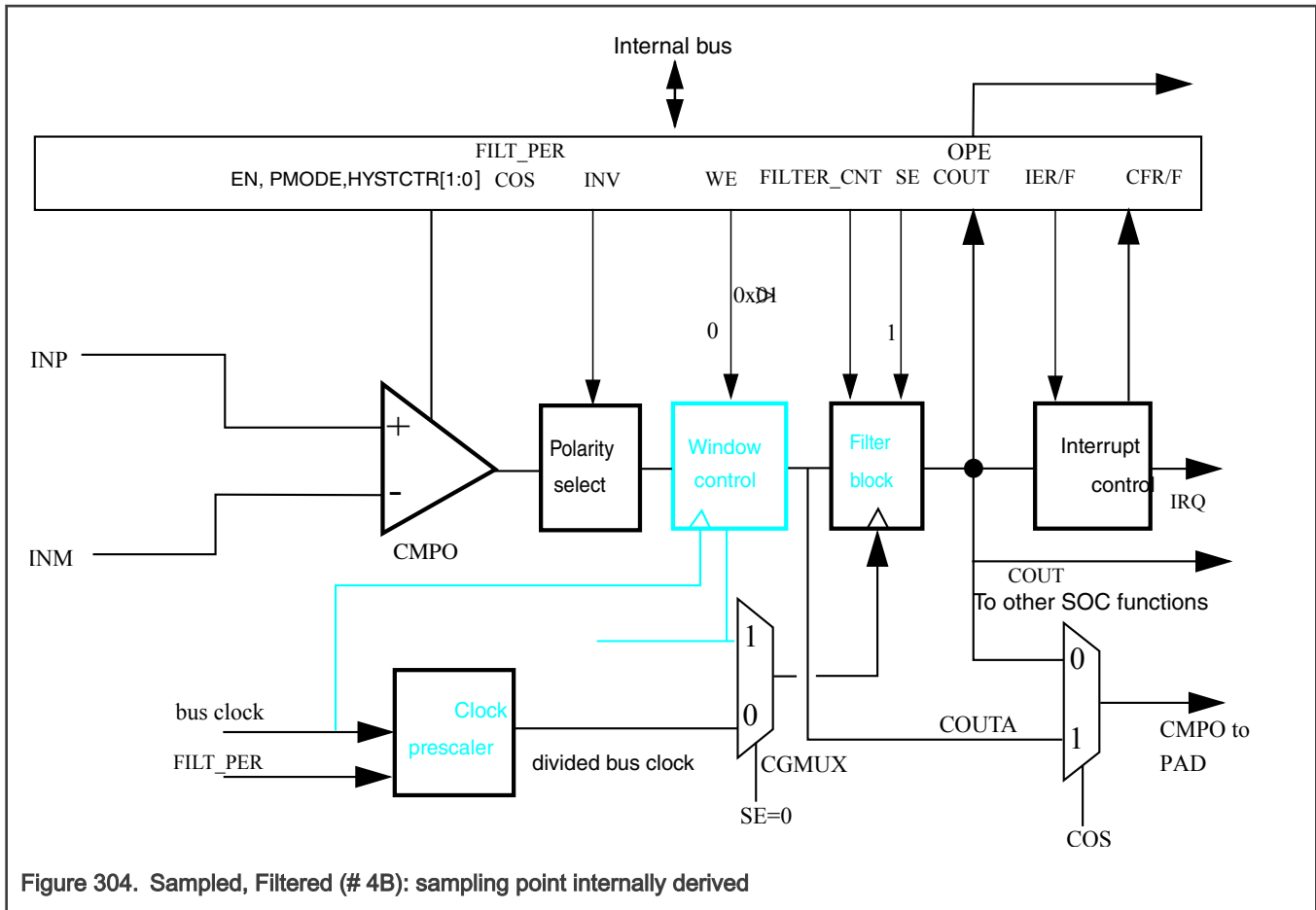


Figure 303. Sampled, Non-Filtered (# 3B): sampling interval internally derived

59.7.4 Sampled, Filtered mode (#s 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unlocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, $CMP\_C0[FILTER\_CNT] > 1$, which activates filter operation.



The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, $CMP\_C0[FILTER\_CNT] > 1$, which activates filter operation.

59.7.5 Windowed mode (#s 5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

NOTE

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

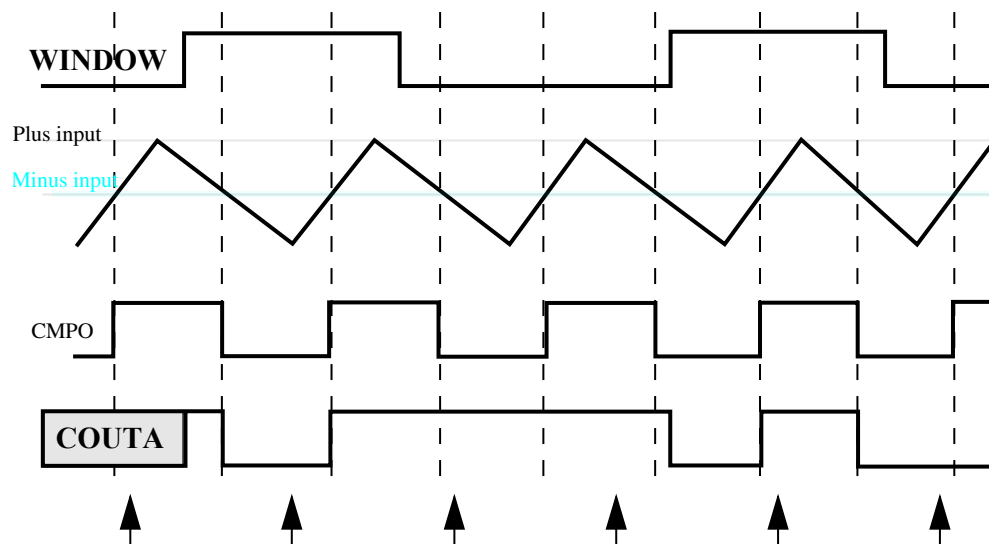


Figure 305. Windowed mode timing diagram

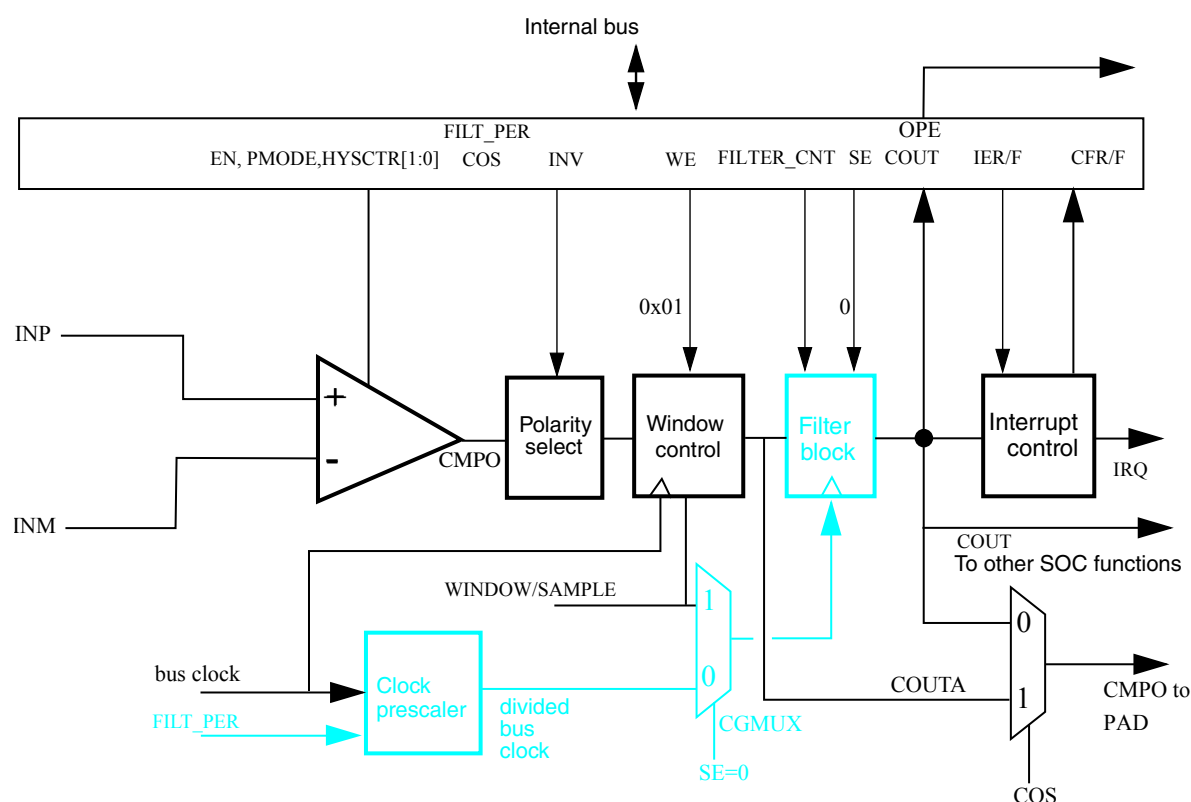


Figure 306. Windowed mode

For control configurations which result in disabling the filter block, see [Figure 301](#).

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

59.8 CMP functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM. CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CMP_C0[INVT] = 1.

CMP_C0[IER] and CMP_C0[IEF] are used to select the condition that causes the CMP module to assert an interrupt to the processor. CMP_C0[CFF] is set on a falling edge, and CMP_C0[CFR] is set on a rising edge of the comparator output. The optionally filtered CMPO can be read directly through CMP_C0[COUT].

59.8.1 Initialization

A typical startup sequence is as follows.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter. See the datasheet for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the [Low-pass filter](#) section.

During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and CMP_C0[CFR]/CMP_C0[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT initially equals 0 until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

59.8.2 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by CMP_C0[FPR], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

59.8.2.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CMP_C0[FILTER_CNT] > 0x01 and
- Setting CMP_C0[FPR] to a nonzero value or setting C0[SE]=1

If using the divided bus clock to drive the filter, it samples COUTA every CMP_C0[FPR] bus clock cycles.

The filter output is at logic 0 when first initialized, and subsequently changes when all the consecutive CMP_C0[FILTER_CNT] samples agree that the output value has changed. In other words, CMP_C0[COUT] is 0 for some initial period, even when COUTA is at logic 1.

Setting all of CMP_C0[SE], CMP_C0[FPR] and CMP_C0[FILTER_CNT] to 0 disables the filter and eliminates switching current associated with the filtering process.

NOTE

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CMP_C0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CMP_C0[SE]=1, the filter samples COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CMP_C0[FILTER_CNT] samples agree that the output value has changed.

59.8.2.2 Latency issues

The value of CMP_C0[FPR] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CMP_C0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CMP_C0[FILTER_CNT].

The values of CMP_C0[FPR] or SAMPLE period and CMP_C0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CMP_C0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation *in the absence of noise*. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 604. Comparator sample/filter maximum latencies

Mode #	CMP_C0[EN]	CMP_C0[WE]	CMP_C0[SE]	CMP_C0[FILTER_CNT]	Co[FPR]	Operation	Maximum latency ¹
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x00	X	Continuous Mode	T_{PD}
2B	1	0	0	X	0x00		
3B	1	0	0	0x01	> 0x00	Sampled, Non-Filtered mode	$T_{PD} + (CMP\_C0[FPR] * T_{per}) + T_{per}$
4B	1	0	0	> 0x01	> 0x00	Sampled, Filtered mode	$T_{PD} + (CMP\_C0[FILTER\_CNT] * CMP\_C0[FPR] * T_{per}) + T_{per}$
5B	1	1	0	X	0x00	Windowed mode	$T_{PD} + T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the bus clock.

NOTE

The delay in this table does not include the delay caused by the Discrete Mode.

59.8.3 CMP Discrete mode timing sequence

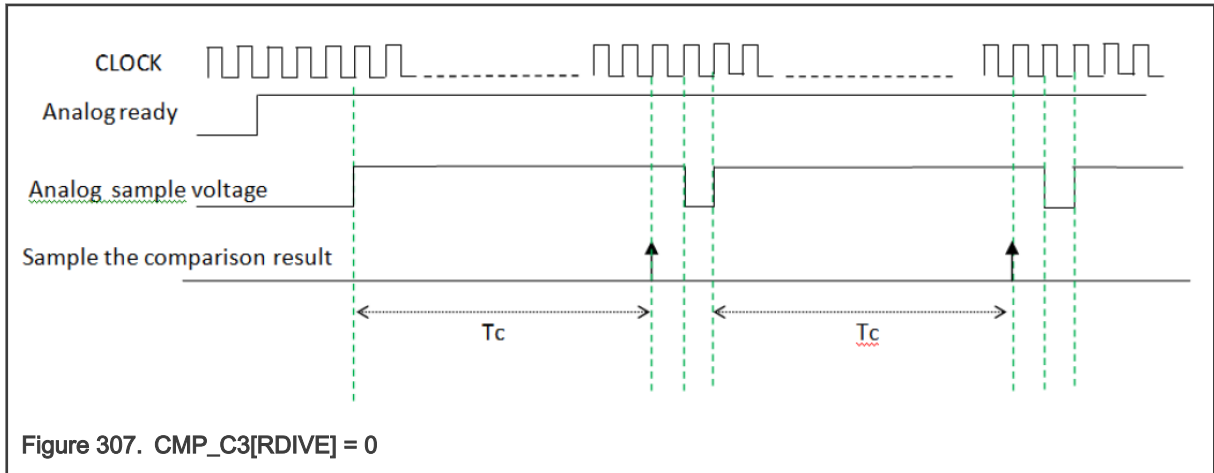
In order to support the 3 V domain input the analog comparator should be configured to work in Discrete Mode since the internal transistors are not 3 V tolerance. This section summarizes the possible configurations depending on the input range.

Note that the sample signal below means that the analog comparator samples the input analog input. It is different from the sample mode that occurs on the digital process to the final comparison result. Also, note that the cross domain round robin cycling (see [Trigger mode](#)) is not allowed.

- When CMP_C3[PCHCTEN] and CMP_C3[NCHCTEN] are set to 1, the CMP is in continuous time operation. No special timing is required.
- When either CMP_C3[PCHCTEN] or CMP_C3[NCHCTEN] is 0, it means at least one input comes from the 3V PAD.
 - If CMP_C3[RDIVE] is zero, it means the input signal comes from 3V PAD, but its range still is in the range of 0 to 1.8V, no timing requirement for the generation of Phase1 and Phase2.

In this condition, there are two modes for this comparator, high speed mode and low speed mode. Clock source may come from fast clock(16-20M)

In [Figure 307](#), the Analog ready indicates the analog settling time is reached and is ready for comparison. T_c is set by $CMP\_C3[ACSAT]$, in high speed mode, it is normally used as $1*T$, $2*T$, $4*T$ and $8*T$ where T is CLOCK period, and in low speed mode ($CMP\_C0[PMODE] = 0$), $CMP\_C3[ACSAT]$ is normally used as $16*T$, $32*T$, $64*T$ and $256*T$.



- b. If $CMP\_C3[RDIVE] == 1$, it means the signals come from the 3V PAD and it could over 1.8 V, Phase1, Phase2 will be generated.

If CMP is in high speed mode ($CMP\_C0[PMODE] = 1$), T_c , Phase1, and Phase2 can be set based on the possible combinations in [Table 605](#).

If CMP works in low speed mode, Phase1 is suggested to set to $1*T$, Phase2 is suggested to set to $8*T$, and T_c is normally set to $16*T$, $32*T$, $64*T$ and $256*T$.

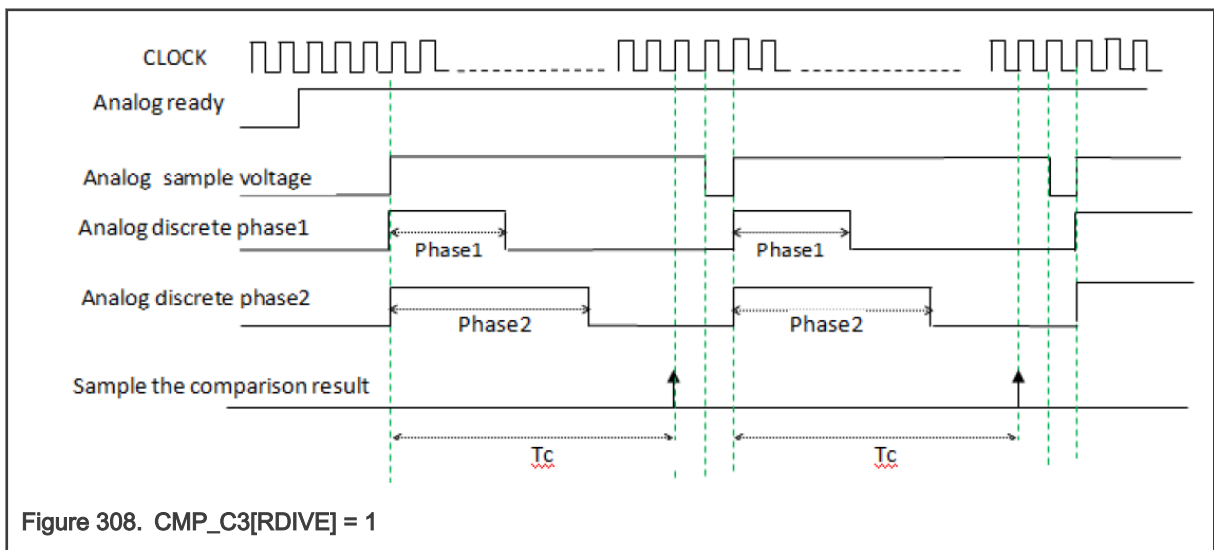


Table 605. Possible Combinations of $CMP\_C3[ACSAT]$, $CMP\_C3[ACPH1TC]$ and $CMP\_C3[ACPH2TC]$

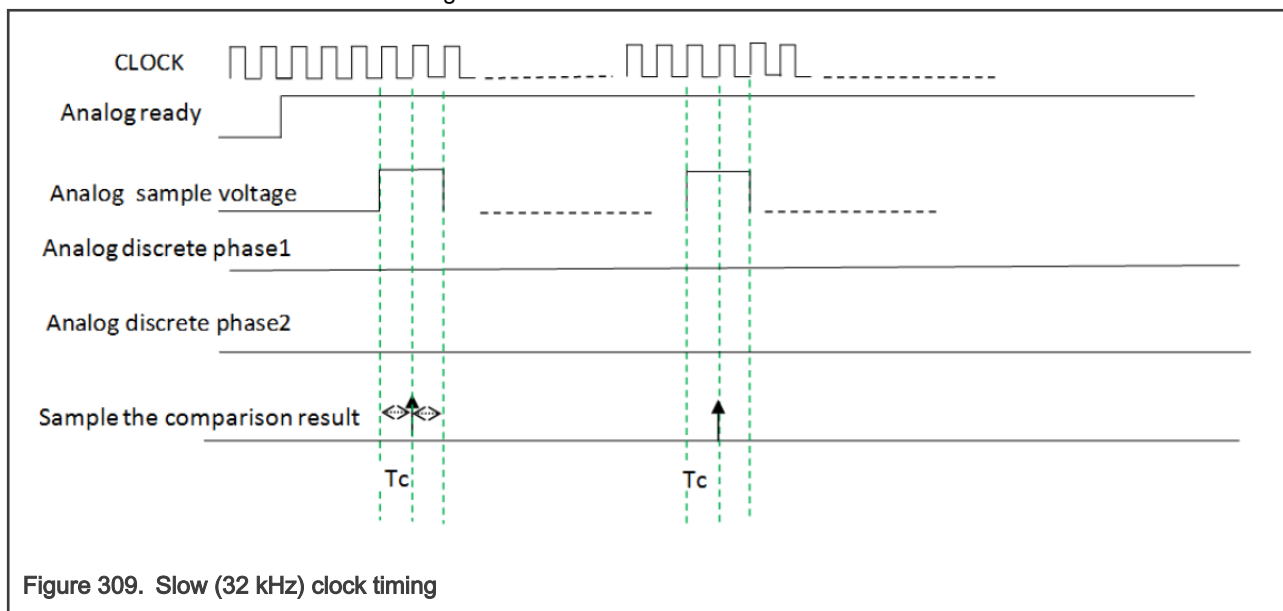
T_c	Phase1	Phase2
$1*T$	$1*T$	$1*T$
$2*T$	$2*T$	$2*T$

Table continues on the next page...

Table 605. Possible Combinations of CMP_C3[ACSAT], CMP_C3[ACPH1TC] and CMP_C3[ACPH2TC] (continued)

Tc	Phase1	Phase2
4*T	4*T	4*T
8*T	8*T	8*T
16*T	1*T	16*T
32*T	1*T	32*T
64*T	1*T	64*T
256*T	0	16*T

If the clock comes from 32KHz slow clock(CMP_C3[DMCLKSEL] = 0), both analog phase 1 and phase 2 will be driven to 0 and Tc is limited to T as shown in the figure below.



59.9 Interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both. Assuming the CMP DMA enable bit is not set, the following table gives the conditions in which the interrupt request is asserted and deasserted.

Table 606. CMP interrupt generations

When	Then
CMP_C0[IER] and CMP_C0[CFR] are set	The interrupt request is asserted
CMP_C0[IEF] and CMP_C0[CFF] are set	The interrupt request is asserted
CMP_C0[IER] and CMP_C0[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
CMP_C0[IEF] and CMP_C0[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

59.10 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting CMP_C0[DMAEN] and the interrupt is enabled by setting CMP_C0[IER], CMP_C0[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flags (CMP_C0[CFR] and CMP_C0[CFF]) to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes. When DMA support is enabled by setting CMP_C0[DMAEN] and the interrupt is enabled by setting CMP_C0[IER], CMP_C0[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, the system will go back to STOP modes. Refer to the DMA chapters in the device reference manual for the asynchronous DMA function for details.

59.11 DAC functional description

This section provides DAC functional description.

59.11.1 Digital-to-analog converter block diagram

The following figure shows the block diagram of the DAC module. It contains a 256-tap resistor ladder network and a 255-to-0 multiplexer, which selects an output voltage from one of 256 distinct levels that outputs from DACO. It is controlled through the control register2(CMP_C1). Its supply reference source can be selected from two sources V_{in1} and V_{in2} . The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

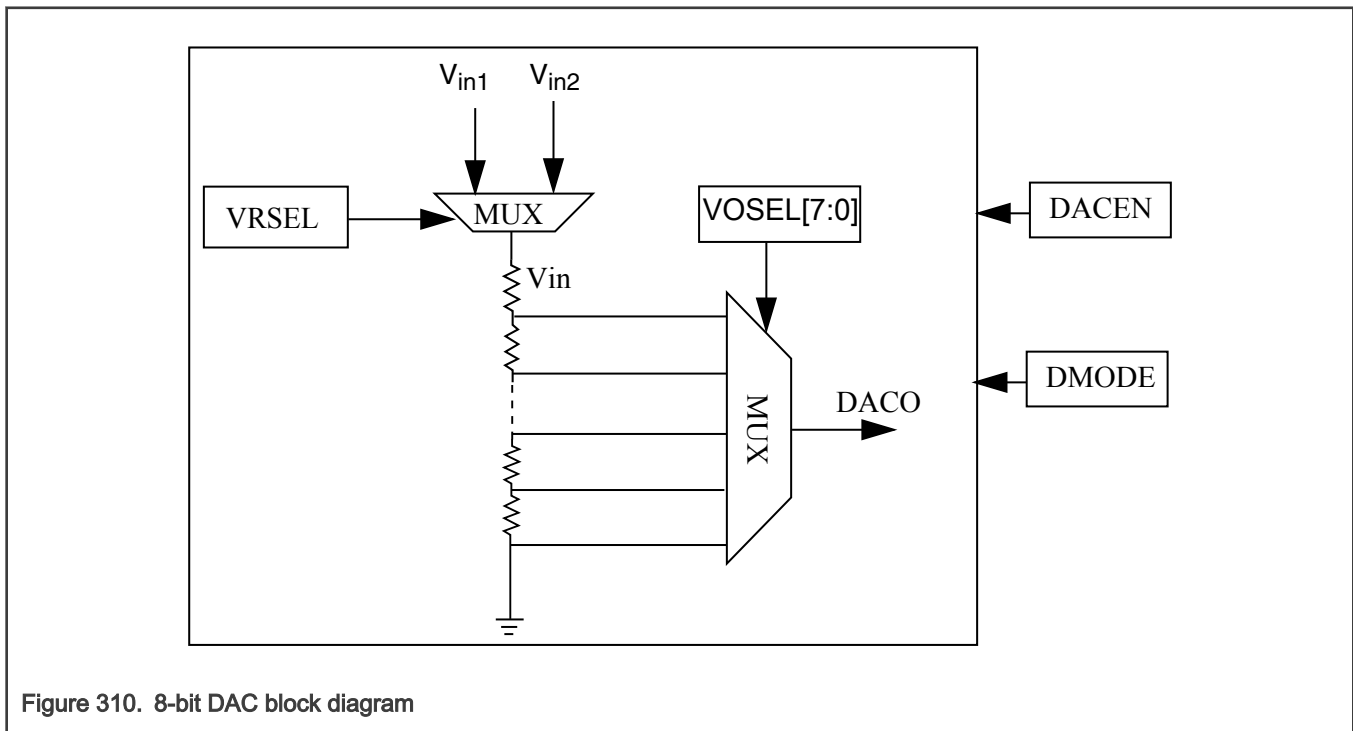


Figure 310. 8-bit DAC block diagram

59.11.2 Voltage reference source select

- V_{in1} must be used to connect to the primary voltage source as supply reference of 64 tap resistor ladder
- V_{in2} must be used to connect to an alternate voltage source, or primary source, if an alternate voltage source is not available

59.12 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

59.13 DAC clocks

This module has a single clock input, the bus clock.

59.14 DAC interrupts

This module has no interrupts.

59.15 Trigger mode

The CMP and the 8-bit DAC are designed to support the trigger mode operation, which is enabled when the MCU enters STOP modes with CMP_C0[WE], CMP_C0[SE] and CMP_C0[EN] are set.

With this mode enabled, the trigger events that include the operation clock and a trigger start signal will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. A fixed channel for either the plus side mux or the minus side mux is selected by software with CMP_C2[FXMP] and CMP_C2[FXMXCH]. It is a mandatory request that the round robin cycling period must be set longer than the time that all the active channels complete the specified comparison cycles specified by CMP_C2[NSAM].

The active channels selected by CMP_C1[CHN*n*] are then routed to the non-fixed channel mux and compared with the reference input in a round-robin manner. In order to meet the comparator stabilization time, after the configurable number of operation clocks defined by CMP_C2[NSAM], the comparison result is sampled for the selected channel. A software pre-programmed state for each channel is configured by writing to CMP_C2[ACO*n*] field. After all the active channels are sampled, if the comparison result changes from its pre-programmed state, the corresponding flag in CMP_C2[CH*n*F] is set. If CMP_C2[RRIE] is set, an asynchronous reset is asserted to bring the MCU out of STOP mode.

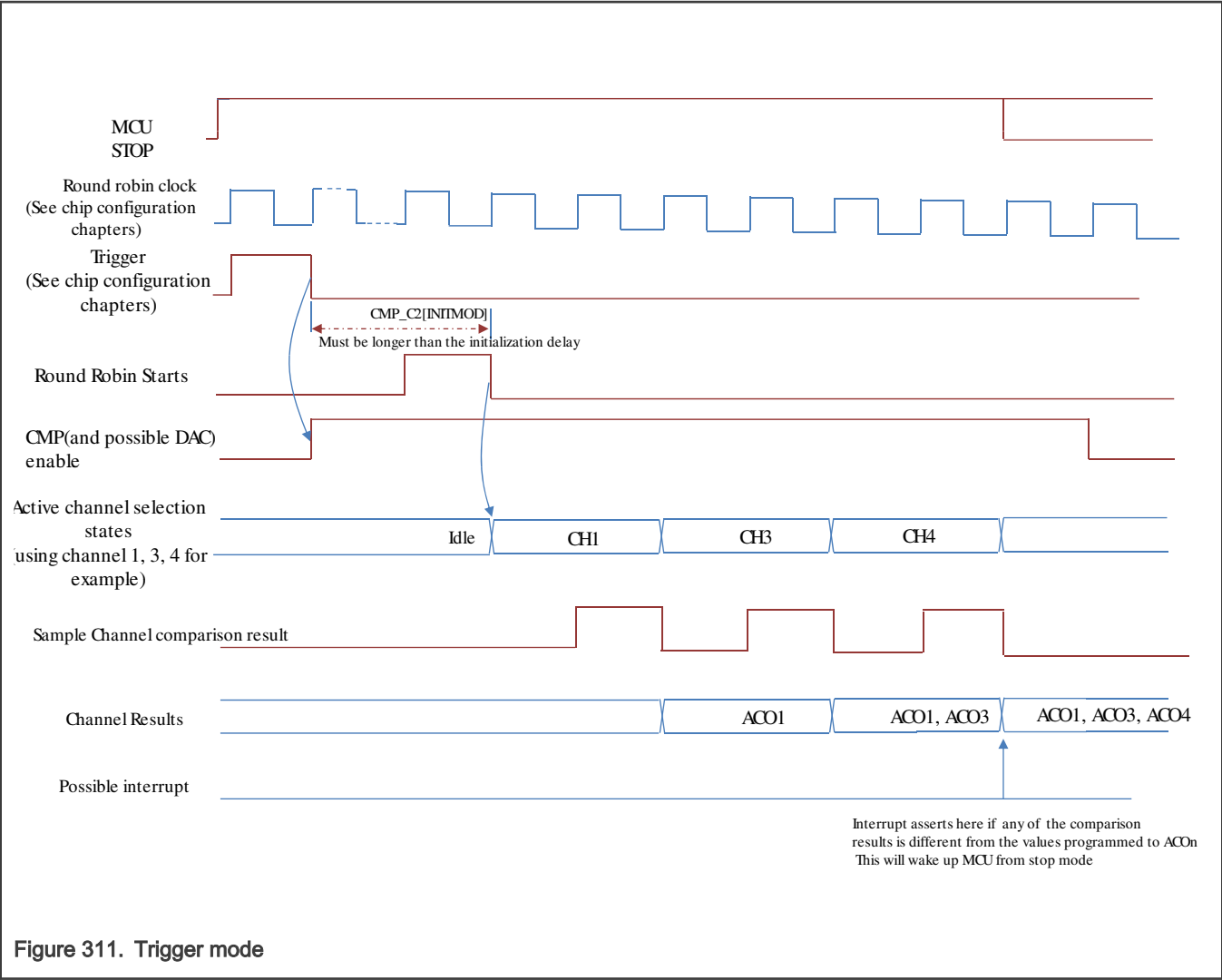
Note that these flags do not support generating a DMA transfer event.

This mode is active when the MCU is in STOP mode, so none of the window/filter functions are available. A basic assumption of this mode is that the selected inputs are changing at a much slower rate than the operation clock. It is suggested to configure the comparator in low power comparison mode as well. In programming the CMP_C2[INITMOD] registers it is need to make sure the INITMOD*round robin clock period must be longer than the initialization delay which can be referred from the chip datasheet.

The following diagram shows the basic flow of this mode. In the diagram, CMP_C1[CHN1], CMP_C1[CHN3], and CMP_C1[CHN4] are set, so channels #1, #3, and #4 are selected for round robin depended on their polarity setting. CMP_C2[NSAM] is set to 2'b01, so one clock later the comparison result of the selected channel is sampled. When channel #3 is compared, the result is sampled, and round robin ends. If any of the comparison results from channel #1, #3, or #4 changed from their programmed value (written to CMP_C2[ACO1], CMP_C2[ACO3], and CMP_C2[ACO4]), an interrupt is generated to wake up the MCU from the STOP mode. Software can then poll the CMP_C2[CH*n*F] to see which channel input(s) changed value during the STOP mode.

NOTE

In round robin mode, it should be ensured that the RTC_CLK period is greater than the comparison time corresponding to the value of CMPx_C0[PMODE].



59.16 Memory map/register definitions

59.16.1 CMP register descriptions

59.16.1.1 CMP memory map

ACMP0 base address: 4013_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	R	0100_0000h
4h	Parameter Register (PARAM)	32	R	0000_0000h
8h	CMP Control Register 0 (C0)	32	RW	0000_0000h
Ch	CMP Control Register 1 (C1)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
10h	CMP Control Register 2 (C2)	32	RW	0000_0000h
14h	CMP Control Register 3 (C3)	32	RW	1100_0000h
18h	Round-Robin Timer Control Register (RR_TIMER_CR)	32	RW	See section

59.16.1.2 Version ID Register (VERID)

Offset

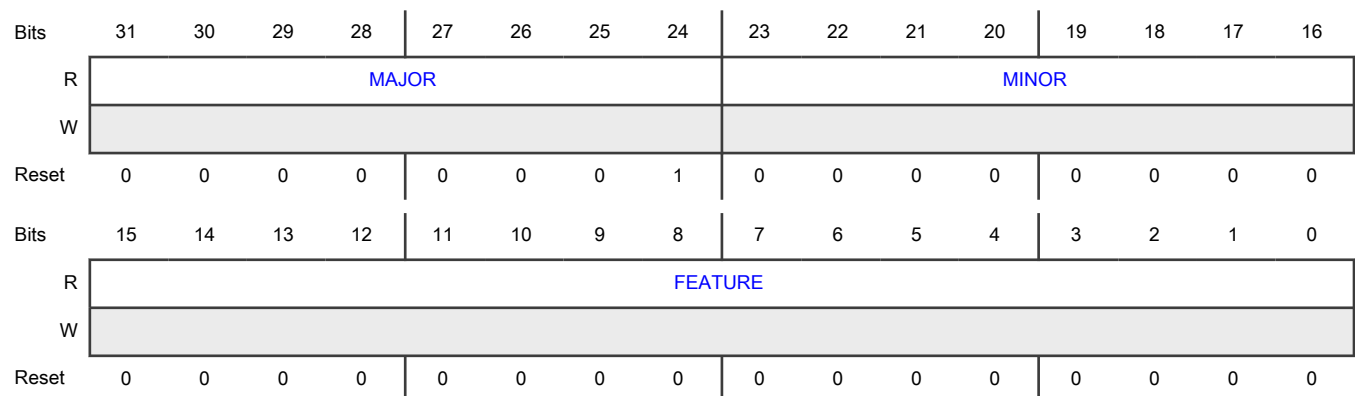
Register	Offset
VERID	0h

Function

Access:

- Supervisor read only
- User read only

Diagram



Fields

Field	Function
31-24 MAJOR	Major Version Number. This read only field returns the major version number for the module specification.
23-16 MINOR	Minor Version Number. This read only field returns the minor version number for the module specification.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15-0 FEATURE	Feature Specification Number. This read only filed returns the feature set number.

59.16.1.3 Parameter Register (PARAM)

Offset

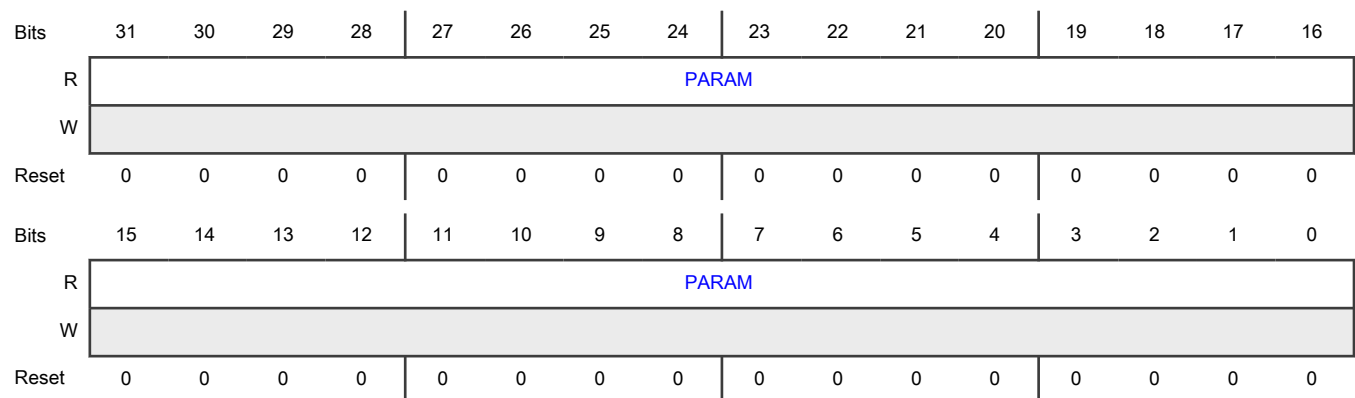
Register	Offset
PARAM	4h

Function

Access:

- Supervisor read only
- User read only

Diagram



Fields

Field	Function
31-0 PARAM	Parameter Registers. This read only filed returns the feature parameters implemented along with the Version ID register.

59.16.1.4 CMP Control Register 0 (C0)

Offset

Register	Offset
C0	8h

Function

Access:

- Supervisor read/write
- User read/write

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LINKEN	DMAEN	0	IER	IEF	CFR	CFF	COUT	FPR							
W						W1C	W1C									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SE	WE	0	PMODE	INVT	COS	OPE	EN	0	FILTER_CNT				0	HYSTCTR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 LINKEN	CMP to DAC link enable. This bit is used to enable the link from CMP to DAC enables. When this bit is set, the DAC enable/disable is controlled by CMP_C0[EN] bit instead of CMP_C1[DACEN]. 0b - CMP to DAC link is disabled 1b - CMP to DAC link is enabled.
30 DMAEN	DMA Enable Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set. 0b - DMA is disabled. 1b - DMA is enabled.
29 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 IER	Comparator Interrupt Enable Rising Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
27 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
26 CFR	Analog Comparator Flag Rising Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive 0b - A rising edge has not been detected on COUT. 1b - A rising edge on COUT has occurred.
25 CFF	Analog Comparator Flag Falling Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level sensitive . 0b - A falling edge has not been detected on COUT. 1b - A falling edge on COUT has occurred.
24 COUT	Analog Comparator Output Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as C0[INVT] when the Analog Comparator module is disabled, that is, when C0[EN] = 0. Writes to this field are ignored.
23-16 FPR	Filter Sample Period Specifies the sampling period, in bus clock cycles, of the comparator output filter, when C1[SE] = 0. Setting FPR to 0x0 disables the filter. Filter programming and latency details are provided in the CMP functional description. This field has no effect when C0[SE] = 1. In that case, the external SAMPLE signal is used to determine the sampling period.
15 SE	Sample Enable If a write to this register attempts to set both SE and WE, then only Sampling mode is taking effect in MCU Run mode and the Round Robin Cycling(Trigger Mode) is enabled in MCU STOP mode. 0b - Sampling mode is not selected. 1b - Sampling mode is selected.
14	Windowing Enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
WE	If a write to this register attempts to set both SE and WE, then only Sampling mode is taking effect in MCU Run mode and the Round Robin Cycling(Trigger Mode) is enabled in MCU STOP mode. 0b - Windowing mode is not selected. 1b - Windowing mode is selected.
13 —	Reserved
12 PMODE	Power Mode Select 0b - Low Speed (LS) comparison mode is selected. 1b - High Speed (HS) comparison mode is selected.
11 INVT	Comparator invert This bit allows selecting the polarity of the analog comparator function. It is also driven to the COUT output (on both the device pin and as C0[COUT]) when CR1[OPE]=0. 0b - Does not invert the comparator output. 1b - Inverts the comparator output.
10 COS	Comparator Output Select 0b - Set CMPO to equal COUT (filtered comparator output). 1b - Set CMPO to equal COUTA (unfiltered comparator output).
9 OPE	Comparator Output Pin Enable The OPE bit enables the path from the comparator output to a selected pin. 0b - When OPE is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin. 1b - When OPE is 1, and if the software has configured the comparator to own a packaged pin, the comparator is available in a packaged pin.
8 EN	Comparator Module Enable The EN bit enables the Analog Comparator Module. When the module is not enabled, the analog part remains in the off state, and consumes no power. When the same input is selected from analog mux to the positive and negative port, the comparator is disabled automatically. 0b - Analog Comparator is disabled. 1b - Analog Comparator is enabled.
7 —	Reserved
6-4 FILTER_CNT	Filter Sample Count

Table continues on the next page...

Table continued from the previous page...

Field	Function
	This field specifies the number of consecutive samples that must agree prior to the comparator output filter accepting a new output state. For information regarding filter programming and latency, please see the Functional Description. 000b - Filter is disabled. If SE = 1, then COUT is a logic zero (this is not a legal state, and is not recommended). If SE = 0, COUT = COUTA. 001b - 1 consecutive sample must agree (comparator output is simply sampled). 010b - 2 consecutive samples must agree. 011b - 3 consecutive samples must agree. 100b - 4 consecutive samples must agree. 101b - 5 consecutive samples must agree. 110b - 6 consecutive samples must agree. 111b - 7 consecutive samples must agree.
3-2 —	Reserved
1-0 HYSTCTR	Comparator hard block hysteresis control. See chip data sheet to get the actual hysteresis value with each level 00b - The hard block output has level 0 hysteresis internally. 01b - The hard block output has level 1 hysteresis internally. 10b - The hard block output has level 2 hysteresis internally. 11b - The hard block output has level 3 hysteresis internally.

59.16.1.5 CMP Control Register 1 (C1)

Offset

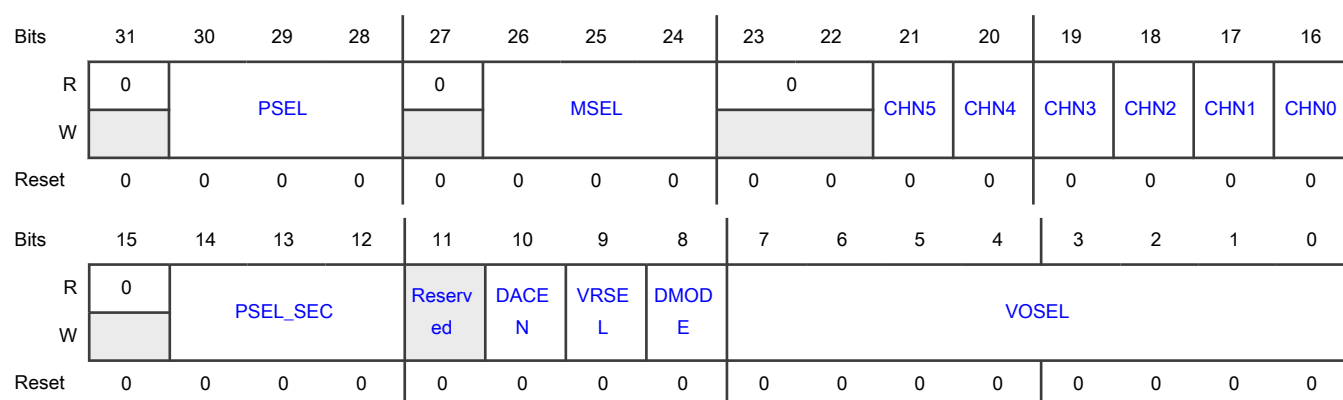
Register	Offset
C1	Ch

Function

Access:

- Supervisor read/write
- User read/write

Diagram



Fields

Field	Function
31 —	Reserved
30-28 PSEL	Plus Input MUX Control These bits determines which input is selected for the positive mux. Refer to the chip configuration chapters to get the detailed connections about Reference Input{0,1,2,3,4,5} and Internal Positive Input 0. The 6b DAC output has been connected internally. Note: For the round robin mode of operation, Internal Posivitive input 0 should be excluded from the activie channels and the fixed channel ports, meanwhile, the MSEL and PSEL bitfields in CMP_C1 register must have different values. 000b - Internal Posivite Input 0 for Plus Channel -- Internal Minus Input 001b - External Input 1 for Plus Channel -- Reference Input 0 010b - External Input 2 for Plus Channel -- Reference Input 1 011b - External Input 3 for Plus Channel -- Reference Input 2 100b - External Input 4 for Plus Channel -- Reference Input 3 101b - External Input 4 for Plus Channel -- Reference Input 4 110b - External Input 4 for Plus Channel -- Reference Input 5 111b - Internal 8b DAC output
27 —	Reserved
26-24 MSEL	Minus Input MUX Control These bits determine which input is selected for the negative mux. Refer to the chip configuration chapters to get the detailed connections about Reference Input{0,1,2,3,4,5} and Internal Negative Input 0. The 6b DAC output has been connected internally.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Note: For the round robin mode of operation, Internal negative input 0 should be excluded from the active channels and the fixed channel ports, meanwhile, the MSEL and PSEL bitfields in CMP_C1 register must have different values. 000b - Internal Negative Input 0 for Minus Channel -- Internal Minus Input 001b - External Input 1 for Minus Channel -- Reference Input 0 010b - External Input 2 for Minus Channel -- Reference Input 1 011b - External Input 3 for Minus Channel -- Reference Input 2 100b - External Input 4 for Minus Channel -- Reference Input 3 101b - External Input 5 for Minus Channel -- Reference Input 4 110b - External Input 6 for Minus Channel -- Reference Input 5 111b - Internal 8b DAC output
23-22 —	Reserved
21 CHN5	Channel 5 input enable Channel 5 of the input enable for the round-robin checker. If CHN5 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
20 CHN4	Channel 4 input enable Channel 4 of the input enable for the round-robin checker. If CHN4 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
19 CHN3	Channel 3 input enable Channel 3 of the input enable for the round-robin checker. If CHN3 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
18 CHN2	Channel 2 input enable Channel 2 of the input enable for the round-robin checker. If CHN2 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
17 CHN1	Channel 1 input enable Channel 1 of the input enable for the round-robin checker. If CHN1 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
16 CHN0	Channel 0 input enable

Table continues on the next page...

Table continued from the previous page...

Field	Function
	Channel 0 of the input enable for the round-robin checker. If CHN0 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
15 —	Reserved
14-12 PSEL_SEC	Secondary Plus channel select. 000b - Input 0 for Plus Channel 001b - Input 1 for Plus Channel 010b - Input 2 for Plus Channel 011b - Input 3 for Plus Channel 100b - Input 4 for Plus Channel 101b - Input 5 for Plus Channel 110b - Input 6 for Plus Channel 111b - Internal 8b DAC output for Plus Channel
11 —	Reserved
10 DACEN	DAC Enable This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power. 0b - DAC is disabled. 1b - DAC is enabled.
9 VRSEL	Supply Voltage Reference Source Select 0b - Vin1 is selected as resistor ladder network supply reference Vin. Vin1 is from internal PMC. 1b - Vin2 is selected as resistor ladder network supply reference Vin. Vin2 is from PAD.
8 DMODE	DAC Mode Selection 0b - DAC is selected to work in low speed and low power mode. 1b - DAC is selected to work in high speed high power mode.
7-0 VOSEL	DAC Output Voltage Select This bit selects an output voltage from one of 256 distinct levels. $DACO = (Vin/256) * (VOSEL[7:0])$, so the DACO range is from 0 to $(255/256)Vin$.

59.16.1.6 CMP Control Register 2 (C2)

Offset

Register	Offset
C2	10h

Function

Access:

- Supervisor read/write
- User read/write

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	RRIE	FXMP	0	FXMXCH				Reserved			CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W													W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	NSAM			INITMOD					0		ACOn						
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fields

Field	Function
31 —	Reserved
30 RRIE	Round-Robin interrupt enable This bit enables the interrupt/wake-up when the comparison result changes for a given channel. 0b - The round-robin interrupt is disabled. 1b - The round-robin interrupt is enabled when a comparison result changes from the last sample.
29 FXMP	Fixed MUX Port This bit is used to fix the analog mux port for the round-robin mode. 0b - The Plus port is fixed. Only the inputs to the Minus port are swept in each round. 1b - The Minus port is fixed. Only the inputs to the Plus port are swept in each round.
28	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
—	
27-25 FXMXCH	Fixed channel selection This field indicates which channel in the mux port is fixed in a given round-robin mode. If FXDACI is set, FXMXCH has no effect. 000b - External Reference Input 0 is selected as the fixed reference input for the fixed mux port. 001b - External Reference Input 1 is selected as the fixed reference input for the fixed mux port. 010b - External Reference Input 2 is selected as the fixed reference input for the fixed mux port. 011b - External Reference Input 3 is selected as the fixed reference input for the fixed mux port. 100b - External Reference Input 4 is selected as the fixed reference input for the fixed mux port. 101b - External Reference Input 5 is selected as the fixed reference input for the fixed mux port. 110b - Reserved. 111b - The 8bit DAC is selected as the fixed reference input for the fixed mux port.
24-22 —	Reserved
21 CH5F	CH5F External Channel 5 input changed flag. This bit is set If the channel 4 input changed from the last comparison with the fixed mux port.
20 CH4F	CH4F External Channel 4 input changed flag. This bit is set If the channel 4 input changed from the last comparison with the fixed mux port.
19 CH3F	CH3F External Channel 3 input changed flag. This bit is set If the channel 3 input changed from the last comparison with the fixed mux port.
18 CH2F	CH2F External Channel 2 input changed flag. This bit is set If the channel 2 input changed from the last comparison with the fixed mux port.
17 CH1F	CH1F External Channel 1 input changed flag. This bit is set If the channel 1 input changed from the last comparison with the fixed mux port.
16 CH0F	CH0F External Channel 0 input changed flag. This bit is set If the channel 0 input changed from the last comparison with the fixed mux port.
15-14	Number of sample clocks For a given channel, this field specifies how many round-robin clock cycles later the sample takes place.

Table continues on the next page...

Table continued from the previous page...

Field	Function
NSAM	00b - The comparison result is sampled as soon as the active channel is scanned in one round-robin clock. 01b - The sampling takes place 1 round-robin clock cycle after the next cycle of the round-robin clock. 10b - The sampling takes place 2 round-robin clock cycles after the next cycle of the round-robin clock. 11b - The sampling takes place 3 round-robin clock cycles after the next cycle of the round-robin clock.
13-8 INITMOD	Comparator and DAC initialization delay modulus. These values specify the round robin clock cycles used to determine the comparator and DAC initialization delays specified by the datasheet. For example the initialization delay is 80us and the round robin clock is 100kHz, then INITMOD should be set to $80\mu\text{s}/10\mu\text{s} = 8$. 000000 - The modulus is set to 64(same with 111111). other values - Initialization delay is set to $\text{INITMOD} * \text{round robin clock period}$.
7-6 —	Reserved
5-0 ACOn	ACOn The result of the input comparison for channel n. This field stores the latest comparison result of the input channel n with the fixed mux port. Reading this bit returns the latest comparison result. Writing this field defines the pre-set state of channel n.

59.16.1.7 CMP Control Register 3 (C3)

Offset

Register	Offset
C3	14h

Function

Access:

- Supervisor read/write
- User read/write

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			PCHC	0			NCHC	0			RDIVE	0			DMCS
W				TEN				TEN								
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				0				0							
W		ACSAT				ACPH1TC				ACPH2TC			Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-29 —	Reserved
28 PCHCTEN	Positive Channel Continuous Mode Enable. This bit is used to enable the positive channel working in continuous mode. When this bit is set to 1, it means all the inputs to positive channels are coming from 1.8v domain and no discrete timing is required for the positive channel mux. Otherwise, when it is cleared to 0, positive channel will work in discrete mode, means the positive inputs are coming from 3v domain. 0b - Positive channel is in Discrete Mode and special timing needs to be configured. 1b - Positive channel is in Continuous Mode and no special timing is required.
27-25 —	Reserved
24 NCHCTEN	Negative Channel Continuous Mode Enable. This bit is used to enable the negative channel working in continuous mode. When this bit is set to 1, it means all the inputs to negative channels are coming from 1.8v domain and no discrete timing is required for the positive channel mux. Otherwise, when it is cleared to 0, negative channel will work in discrete mode, means the negative inputs are coming from 3v domain. 0b - Negative channel is in Discrete Mode and special timing needs to be configured. 1b - Negative channel is in Continuous Mode and no special timing is required.
23-21 —	Reserved
20 RDIVE	Resistor Divider Enable This bit is used to enable the resistor divider for the inputs when they come from 3v domain and their values are above 1.8v.

Table continues on the next page...

Table continued from the previous page...

Field	Function
	0b - The resistor is not enabled even when either NCHEN or PCHEN is set to 1 but the actual input is in the range of 0 - 1.8v. 1b - The resistor is enabled because the inputs are above 1.8v.
19-17 —	Reserved
16 DMCS	Discrete Mode Clock Selection This bit is used to select the clock source in order to generate the required timing for comparator to work in discrete mode. There are generally two kinds of clocks coming from SoC, one is a fast clock (16 - 20MHz) to generate fast prop delay and another one is normally 32kHz to work in low power mode. 0b - Slow clock is selected for the timing generation. 1b - Fast clock is selected for the timing generation.
15 —	Reserved
14-12 ACSAT	Analog Comparator Sampling Time control. These values configure the analog comparator sampling timing (specified by the discrete mode clock period T which is selected by DMCS) in discrete mode. 000b - The sampling time equals to T 001b - The sampling time equals to 2*T 010b - The sampling time equals to 4*T 011b - The sampling time equals to 8*T 100b - The sampling time equals to 16*T 101b - The sampling time equals to 32*T 110b - The sampling time equals to 64*T 111b - The sampling time equals to 256*T
11 —	Reserved
10-8 ACPH1TC	Analog Comparator Phase1 Timing Control. These values configure the analog comparator phase1 timing when RDIV is set to 1 which means the input voltage level is above 1.8v. T means the discrete mode clock period in the table. 000b - Phase1 active time in one sampling period equals to T 001b - Phase1 active time in one sampling period equals to 2*T 010b - Phase1 active time in one sampling period equals to 4*T 011b - Phase1 active time in one sampling period equals to 8*T

Table continues on the next page...

Table continued from the previous page...

Field	Function
	100b - Phase1 active time in one sampling period equals to T 101b - Phase1 active time in one sampling period equals to T 110b - Phase1 active time in one sampling period equals to T 111b - Phase1 active time in one sampling period equals to 0
7 —	Reserved
6-4 ACPH2TC	Analog Comparator Phase2 Timing Control. These values configures the analog comparator phase2 timing when RDIV is set to 1 which means the input voltage level is above 1.8v. T means the discrete mode clock period in the table. 000b - Phase2 active time in one sampling period equals to T 001b - Phase2 active time in one sampling period equals to 2*T 010b - Phase2 active time in one sampling period equals to 4*T 011b - Phase2 active time in one sampling period equals to 8*T 100b - Phase2 active time in one sampling period equals to 16*T 101b - Phase2 active time in one sampling period equals to 32*T 110b - Phase2 active time in one sampling period equals to 64*T 111b - Phase2 active time in one sampling period equals to 16*T
3-0 —	Reserved

59.16.1.8 Round-Robin Timer Control Register (RR_TIMER_CR)

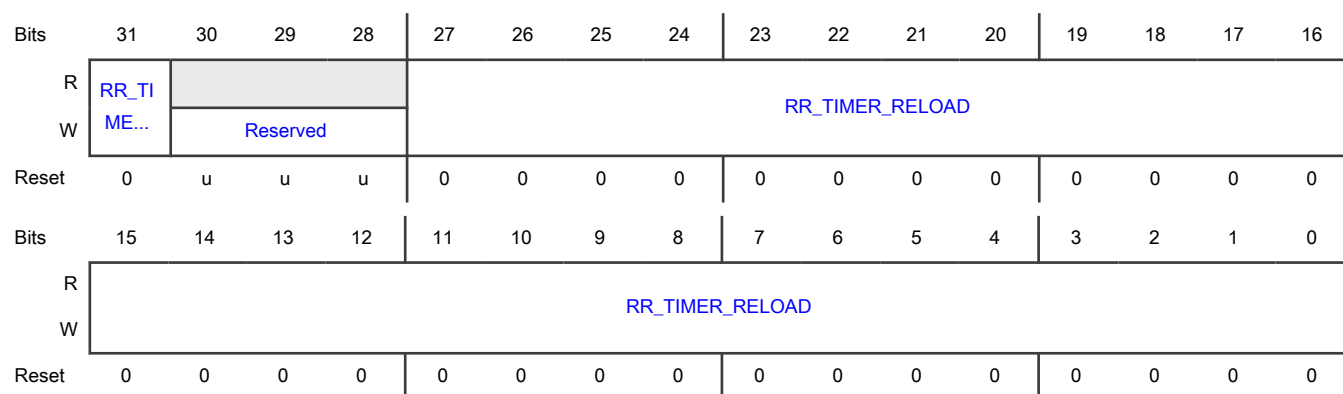
Offset

Register	Offset
RR_TIMER_CR	18h

Function

For very low-power operation, the analog comparator (ACMP) supports a “round-robin” operating mode. In this mode the ACMP is shut down most of the time but is woken up at periodic intervals to perform a series of comparisons on a designated set of channels, then put back into its sleep state. In this mode, comparisons are accomplished using the 32 kHz rr_clock.

Diagram



Fields

Field	Function
31 RR_TIMER_EN A	RR_TIMER enable When low, RR_TIMER count will be held at zero. When set, timer will commence continuous, repetitive counting beginning with the 1st or 2nd rising edge of the 32 KHz rr_clock. NOTE This bit may be set during the same write operation that establishes the RR_TIMER_RELOAD value. The RR_TIMER should be configured and enabled BEFORE enabling RR_MODE operation.
30-28 —	reserved
27-0 RR_TIMER_RE LOAD	This field establishes the repetitive count rate for the timer. Each time the timer counts down to zero it is reloaded with this value. The rr_trig signal will be generated at a rate of (rr_timer_reload + 1) times the rr_clock period (typically 30.6 uS)

Chapter 60

Debug Subsystem

60.1 Overview

This chapter outlines the debug capabilities implemented in this chip.

60.1.1 Features

- Supports Arm Serial Wire Debug mode for the Arm Cortex-M33 processor and the Fusion DSP.
- Trace port provides Cortex-M33 CPU instruction trace capability. Output via a Serial Wire Viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: the Cortex-M33 processor includes instruction breakpoints that can also be used to remap instruction addresses for code patches. Literal comparators can also be used to remap addresses for patches to literal values.
- Watchpoints: the Cortex-M33 processor includes data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan
- Instrumentation Trace Macrocell allows additional software controlled trace for the Cortex-M33 processor.
- The Fusion DSP also includes address and data breakpoints and trace capability.

60.1.2 Glossary

Table 607. Glossary

Abbreviation	Term	Description
RoT	Root of Trust	Vendor-owned key pair that authorizes data assets via cryptographic signatures. The public part of the key is typically pre-configured in products so that data from untrusted sources can be cryptographically verified. The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC).
RoTpub	RoT Public Key	The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTID	RoT Identifier	RoTID allows the debugger to infer which RoT public key(s) are acceptable to the device. If the debugger cannot or does not provide such a credential, the authentication process will fail.
RoTMETA	RoT meta-data	The <i>RoT meta-data</i> required by the device to corroborate; the ROTID sent in the DAC, the field in this DC, and any additional RoT state that is not stored within the device. This allows different RoT identification, management and revocation solutions to be handled.
SoCC	SoC Class	A unique identifier for a set of SoC's that require no SoC-specific differentiation in their debug authentication. The main usage is to allow a different set of debug domains and options to be negotiated between the device configuration and credentials. A class can

Table continues on the next page...

Table 607. Glossary (continued)

Abbreviation	Term	Description
		contain just a single revision of a single SoC model, if the granularity of debug control warrants it.
DCK	Debug Credential Key	A user-owned key pair. The public part of the key is associated with a DC, the private part is held by the user and used to produce signatures during authentication.
DC	Debug Credential	A user public key and associated attributes, bound together and signed by a RoT, serves as an <i>identity</i> .
CC	Credential Constraint	In product configuration, CCs are limitations on the DCs that the device will accept for authentication. In DCs, CCs are vendor/RoT-authorized usages of the DC, as well as inputs to the desired debug behavior.
VU	Vendor Usage	A CC (constraint) value that is opaque to the debug authentication protocol itself but which can be leveraged by vendors in product-specific ways.
SoCU	SoC Usage	A CC (constraint) value that is a bit mask, and whose bits are used in an SoCC-specific manner. These bits are typically used for controlling which debug domains are accessed via the authentication protocol, but device-specific debug options can be managed in this way also.
CB AB	Credential Beacon Authentication beacon	A value that is passed through the authentication protocol, which is not interpreted by the protocol but is instead made visible to the application being debugged. A credential beacon is associated with a DC and is therefore vendor/RoT-signed. An authentication beacon is provided and signed by the debugger during the authentication process.
DCFG_*	Debug Config	Refers to device configuration settings stored in OTP.
CPFA		Customer Programmable Factory area.
CPIA		Customer Programmable in-field area.

60.2 Functional description

60.2.1 Debug subsystem

This figure shows the external connections for Serial Wire Debug.

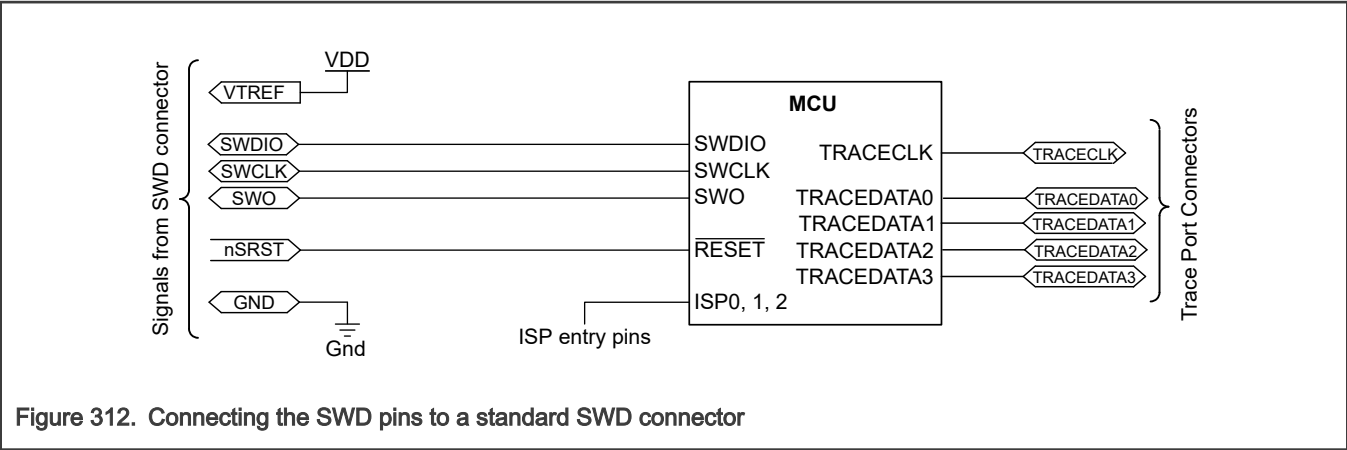


Figure 312. Connecting the SWD pins to a standard SWD connector

This figure shows top-level debug ports and connections in this device.

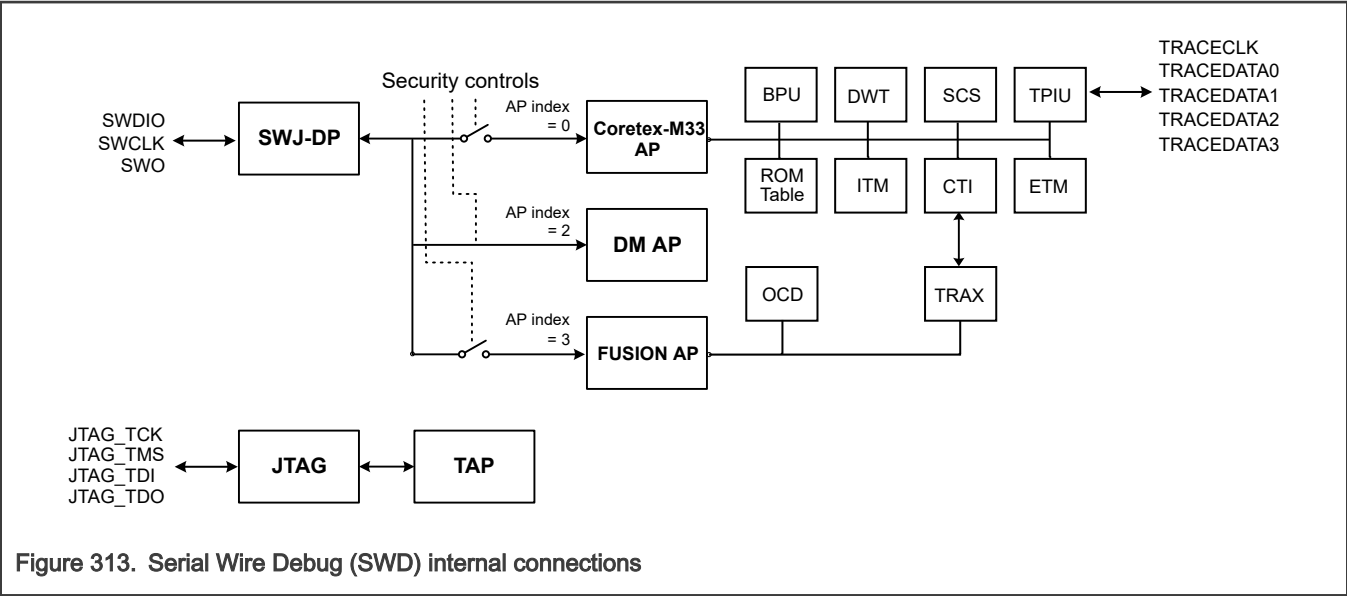


Figure 313. Serial Wire Debug (SWD) internal connections

Blocks SWJ-DP and DM-AP are always enabled and are accessible to the external world. The remaining blocks are enabled/disabled under the hardware state machine and software control.

Table 608. SWD blocks

Block	Description
JTAG-TAP	Test access port used by the NXP product and test engineering team
DAP	Debug access port that a has Serial Wire port (SWJ-DP) which interprets the data coming in and routes it to appropriate Access Port (AP)
Cortex-M33 AP	Debug access port for the Arm Cortex-M33 processor
Fusion DSP AP	Debug access port for the Fusion DSP
DM AP	Debug Access port for the Debug Mailbox

Table continues on the next page...

Table 608. SWD blocks (continued)

Block	Description
	- This port is always enabled and the external world can send and receive data to/from ROM. - This port is used to implement NXP debug authentication protocol version 1.0.
BPU	The Breakpoint Unit (BPU) implements hardware breakpoints. RT500 implements 8 instruction comparators.
DWT	The Data and Address Watchpoints (DWT) contains comparators that you can configure as a hardware watchpoint, a PC sampler event trigger, or a data address sampler event trigger. RT500 implements 4 comparators.
SCS	The processor provides debug through registers in the System Control Space (SCS).
TPIU	The Trace Port Interface Unit (TPIU) acts as a bridge between the on-chip trace data from the Instrumentation Trace Macrocell (ITM) to a data stream, encapsulating IDs where required, that is then captured by a Trace Port Analyzer (TPA). The TPIU is specially designed for low-cost debug. RT500 implements CoreSight SoC-400 TPIU. For more information about the CoreSight SoC-400 TPIU, see the Arm® CoreSight™ SoC-400 Technical Reference Manual.
ROM table	The ROM table identifies which debug IP is available.
ITM	The Instrumentation Trace Macrocell (ITM) is an application-driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information.
CTI	The Cross Trigger Interface (CTI) enables the debug logic, and ETM present in Arm Cortex-M33 processor to interact with each other and with debug components (OCD, TRAX) present in Fusion DSP. This is called cross triggering. For example, you can configure the CTI to generate an interrupt when the ETM trigger event occurs or to start tracing when a DWT comparator match is detected.
ETM	The Embedded Trace Macrocell (ETM)-M33 provides non-intrusive program-flow trace for the Arm Cortex-M33 processor. ETM-M33 generates information that trace software tools use to reconstruct the execution of all or part of a program.
OCD	On-Chip Debugging (OCD) support provides access to and control of the architectural (software-visible) state of the Fusion DSP processor through the Debug Access Port. Through the OCD an external debug agent can:

Table continues on the next page...

Table 608. SWD blocks (continued)

Block	Description
	• Gain control over the processor (stop the processor) either by explicitly generating a debug interrupt or upon any debug exception. • Read and write any software visible register and/or memory location. • Resume normal processor execution (let the processor run). • Communicate with a running processor via the Debug Data Register (DDR).
TRAX	TRAX is a collection of hardware and software that provides increased visibility into the activity of running Fusion DSP processors using compressed execution traces. TRAX is particularly useful for software and system development and debugging.

60.2.2 Debug Access Port (DAP)

The Debug Access Port which has Serial Wire Port (SWJ-DP) interprets the data coming in and routes the data to the appropriate Access Port (AP). External I/O pins that interface with DAP are described in [Table 623](#). This block is always enabled. To disable this function software should set the SWD pins in GPIO mode or select other functions for these pins in the IOPCTL (IO Pad Controller).

60.2.3 Arm Cortex-M33 Access Port

The Debug Access Port for the Arm Cortex-M33 processor is disabled on reset and enabled by the hardware state machine and through the CS_PROTCPU0 register. This port has additional controls to enable/disable different features controlled through the DBG_FEATURES register and the DBG_FEATURES_DP register. Debug Authentication makes use of these features. See [Debug authentication](#).

DBGEN: Invasive debugging of TrustZone for Armv8-M architecture defined non-secure domain.

- Breakpoints and watch points to halt the processor on specific activity.
- A debug connection to examine and modify registers and memory, and provide single-step execution.

NIDEN: Non-Invasive debugging of TrustZone for defined non-secure domain.

- A collection of information on instruction execution and data transfers.
- Deliver trace to off-chip in real-time to tools to merge data with source code on a development workstation for future analysis.

SPIDEN: Invasive debugging of TrustZone for Armv8-M architecture defined secure domain.

SPNIDEN: Non-Invasive debugging of TrustZone for Armv8-M architecture defined secure domain.

60.2.4 Debugger Mailbox Access Port (DM-AP)

The Debugger Mailbox Access Port (DM-AP) offers a register-based mailbox accessible by both CPUs and the device debug port (DP) of the MCU. This port is always enabled and the external world can send and receive data to and from ROM. This port is used to implement the NXP Debug Authentication Protocol.

60.2.4.1 Re-synchronization request

Communication with the DM is initiated by the debugger. The debugger first sets CSW[RESYNCH_REQ] = 1. The debugger must then reset the device by either writing CSW[CHIP_RESET_REQ] = 1, or by driving the actual reset pin of the device if it is able to do so.

60.2.4.2 Acknowledgement of re-synchronization request

After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. If the device has not completed the re-synchronization process, the debugger stalls. The debugger can repeat this process until it is able to read CSW[RESYNCH_REQ] = 0.

60.2.4.3 Return phase

Following the initial re-synchronization, communication by the debugger to the device is in the form of 32-bit writes to the REQUEST register. Allowed commands are listed in [DM-AP commands](#). After receiving the commands, ROM processes them and writes back the result in the RETURN Register. The debugger can read the result in the RETURN register. A WAIT response is returned to the debugger if it attempts to read the RETURN register before ROM writes the result.

60.2.5 JTAG boundary scan

The JTAG port is used for boundary scan testing. JTAG mode is disabled on reset and enabled by Boot Code if the device is configured for that to be allowed. JTAG boundary scan is available on the RT500. JTAG mode entry takes place during Reset with a specific sequence. See [Enabling the TAP controller](#) and [JTAG signal properties in Boundary Scan mode](#) section in the Reference Manual.

Table 609. Pin setting for JTAG boundary scan entry

Pin	Function	Default state	State for JTAG Entry
PIO0_7	TRST_B	Weak pull-down, input enabled	Strong pull-up
PIO1_15	ISP Select 0 ¹	Weak pull-up, input enabled	Strong pull-down
PIO3_28	ISP Select 1 ¹	Weak pull-up, input enabled	Strong pull-down
PIO3_29	ISP Select 2 ¹	Weak pull-up, input enabled	Strong pull-down
PIO2_25	SWCLK	Weak pull-down, input enabled	Strong pull-up
PIO2_26	SWDIO	Weak pull-up, input enabled	Strong pull-down

1. Set by Boot ROM on boot-up

JTAG pins are separate.

NOTE

Boundary scan operations should not be started until 250 ms after POR. The test TAP must be reset after the boundary scan and left in either TLR or RTO state.

NOTE

POR or a LOW on the TRST pin puts the test TAP controller in the test-logic reset state. The first TCK clock while RESET = HIGH places the test TAP in Run-Test Idle mode.

60.3 Debug Mailbox protocol

Boot ROM implements debug mailbox protocol to interact with tools over the SWD interface.

The protocol has following features:

- Request/response based
- Expandable with arbitrary commands and parameters
- Support for relatively large command and response data
- All commands and responses are 32-bit word aligned
- Supports data above 32-bits by using an ACK_TOKEN that moderates the transfer in 32-bit value chunks.
- Requests and responses use the same basic structure

The ROM provides three debug methods/mechanisms to attach the debugger in a predictable manner:

- Start debug session method
- Debug access trigger method
- Debug authentication mechanism

Debug authentication is disabled by default, and is set up (if required) during development or device provisioning during the production phase of a product using the device. See [Debug authentication](#) for full details of debug authentication.

When program control is in ROM memory context (i.e., instructions are fetched from the ROM memory address range) during the boot process, the debug access port (AP) of CPU0 is disabled irrespective of device life-cycle state or DCFG_SOCU settings. This mechanism is referred as 'Boot-ROM protection' in this document. Thus the method to initiate a debug session will vary depending on the device state and intended debug scenario. The scenarios described in the rest of these section are as follows:

- Boot media (external Octal/Quad-SPI flash or eMMC/SD card flash) is uninitialized or does not contain a valid, bootable image. If the boot media does not contain a valid image, the ROM will proceed to ISP mode and wait to be booted via one of its serial interfaces; in ISP mode the debug interface is disabled.
- ISP mode, initiated because the ISP pin was asserted on the device at reset.
- Connection to a device running a valid application, with the intent to update flash with a new application.
- Connection to a device running a valid application in flash, with the intent to debug without updating flash (also called a "debug attach").

60.3.1 Debug session with uninitialized/invalid flash image or ISP mode

When the device boots, there may be no valid image in the boot media, at which point the ROM-based program control enters the In-System Programming (ISP) loop, and debug access is disabled for security reasons. Another scenario is where the device may be placed into ISP mode because the ISP has been asserted as the device leaves reset. This section describes how to establish a debug session for these scenarios.

To ensure the state machine controlling debug mailbox commands is in a known state, the debugger may need to reset this logic; this is done by setting the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW. After requesting a re-synchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed the re-synchronization process, indicated by reading a value of 0 from the CSW register (checking only the lower 16-bits of this 32-bit value).

To start a debug session and control the exchange of debug information, use the DM-AP commands. Following a successful initial re-synchronization, communication by the debugger to the device is achieved using 32-bit DM-AP command writes to the REQUEST register in the Debug Mailbox (DM-AP Commands are shown in [DM-AP commands](#).) The debugger can read results of communications via the RETURN register. The debugger should poll the RETURN register in the same manner as it polled the CSW following a re-synchronization request in order to ensure transactions have completed successfully. Response codes are shown in [DM-AP response codes](#). To handle situations where debug is disabled due to Boot-ROM protection, the debug system must use a specific command (Start Debug Session) and follow the debug mailbox protocol to indicate to the ROM-based boot code its intention of connecting a debug session over SWD. Upon receiving the command, the boot code ensures any unwanted peripheral interrupts are disabled and secrets managed before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

Once the Start Debug Session and device reset have been successfully executed, the AP for CPU0 (Cortex-M33 AP) will be accessible and can be used to set breakpoints, etc. as with other Arm Cortex-M devices.

Following is a sample that shows how a debug session is initiated for the scenarios described above:

```
// Pseudo Code Syntax
// -----
// WriteDP "register" "value"
// value = ReadDP "register"
// AP transactions presume the DM AP is selected
// WriteAP "register" "value"
// value = ReadAP "register" "value"
// -----

// Read AP ID register to identify DM AP at index 2
WriteDP 2 0x020000F0
// The returned AP ID should be 0x002A0000
value = ReadAP 3
print "AP ID: ", value

// Select DM AP index 2
WriteDP 2 0x02000000
// Write DM RESYNC_REQ + CHIP_RESET_REQ
WriteAP 0 0x21
// Poll CSW register (0) for zero return, indicating success
value = -1
while value != 0 {
    value = ReadAP 0
}
print "RESYNC_REQ + CHIP_RESET_REQ: ", value
// Write DM START_DBG_SESSION to REQUEST register (1)
WriteAP 1 7
// Poll RETURN register (2) for zero return
value = (ReadAP 2)
print "DEBUG_SESSION_REQ: ", value
```

Following a successful debug connection, a flash loader would normally be loaded into RAM and used to program the application to be debugged, and the required breakpoints in the application code set. Once this is done, a SYSRESET_REQ command should be issued to ensure the ROM fully executes (as would happen in a deployed end application) before reaching the downloaded application.

60.3.2 Debug session with valid application in flash

In this scenario description it is assumed that debug has not been disabled by an application already in flash, so the CPU0 AP (Cortex-M33 AP) is visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware or issue a Debug Session Request. The same methods described in [Debug session with uninitialized/invalid flash image or ISP mode](#) may be used in order to simplify debug support implementations.

60.3.3 Debug session attaching to a running target

This scenario is when the device has booted and is running an application that has not disabled debug, and the host system is attempting to connect to that device without resetting it and with no intention to update flash. In this case, the CPU0 AP (Cortex-M33 AP) should be visible and breakpoints can be set without the need to re-synchronize the Mailbox hardware.

60.3.4 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector in order to break code execution at this point, however the chip's ROM prevents this. To allow the debug system to halt execution immediately after the ROM has completed preparations after a debug session request, a watchpoint may be set to trigger on a read access to address 0x50002034.

The debugger should use following reset sequence:

1. Set the data watchpoint to halt the core on a read of address location 0x50002034.

2. If all data watchpoints comparators are occupied, back-up one of the watchpoint settings and replace it with the above watchpoint location.
3. Reset the core and peripherals by setting the AIRCR[SYSRESETREQ].
4. Wait for 100 msec to allow ROM to re-enable debug access.
5. Check the DHCSR register to see the core halted due to the data watchpoint.
6. If read of DHCSR times out or returns error response, then it indicates that the ROM entered ISP command handling loop due to an invalid image in boot media.
 - a. Execute start debug session sequence described in [Debug session with uninitialized/invalid flash image or ISP mode](#)
 - b. Wait for 10 msec.
 - c. Enable the Cortex-M33 AP.
 - d. Read DHCSR and issue HALT to Cortex-M33 AP.
7. Clear data watchpoint added in step 1. If watchpoint was set as described in step 2 then restore the watchpoint configuration.

60.4 Reset handling

The debug domain (DP, Cortex-M33 AP, DM-AP) is reset upon POR (Power On Reset). On other resets, the debug domain retains its state and the defined breakpoints, watch points, etc., survive even when the debugging tool issues a reset to the device.

The DAP for CPU0 is disabled during a power on reset or assertion of the reset pin and enabled by the ROM if/when the correct debug initiation procedure(s) are followed. If the DAP is not being used, the debug enablement protocol can be used to initiate a debug session. If Debug Authentication is required, see [Debug authentication](#).

60.5 Mailbox commands

This section describes the Return and Request message formats and available mailbox commands and their associated parameters.

60.5.1 Request

The first word transmitted in a request is a header word containing the command ID and number of following data words. The command packet is sent to the device by writing 32-bits at a time to the REQUEST register. When sending command packets greater than 32-bits, the debugger should read an ACK_TOKEN in the RETURN register before writing the next 32-bits.

Following the header are the number of 32-bit words specified in the header.

Table 610. Request register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandID[7:0]	commandID[15:8]	dataWordCount[7:0]	dataWordCount[15:8]
1	<i>data...</i>			

The C structure definition for a request is as follows:

```
struct dm_request {
    uint16_t commandID;
    uint16_t dataWordCount;
    uint32_t data[ ];
};
```

60.5.1.1 DM-AP commands

Commands for the DM-AP are listed below. These would be written to the REQUEST register. With the exception of the "Enter ISP mode" and "Start Debug Session" command, the issuance of a command (or sequence of commands) is normally followed by the issuance of an "Exit DM-AP" command to resume normal device boot flow.

Table 611. DM-AP commands

Name	Command code	Parameter/ Response	Description
Start DM-AP	0x01	<u>Parameters:</u> None <u>Response:</u> 32-bit status	Cause the device to enter DM-AP command mode. This must be done prior to sending other commands.
Bulk Erase	0x02	<u>Parameters:</u> None <u>Response:</u> 32-bit status	Erase the entire on-chip flash memory.
Reserved	0x03	<u>Parameters:</u> None <u>Response:</u> 32-bit status	Reserved, Returns 3.
Exit DM-AP	0x04	<u>Parameters:</u> None <u>Response:</u> 32-bit status	Cause the device to exit DM-AP command mode. The Device returns to normal mode.
Enter ISP Mode	0x05	<u>Parameters:</u> dataWordCount: 0x1 data[0]: ISP mode enum. Same as IAP API. <u>Response:</u> commandStatus[15:0]: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	Enter specified ISP mode. By default, customers would disable ISP boot through OTP before field deployment. Loading large images through SWD in field might be cumbersome. So, technicians would prefer downloading image through faster ISP interfaces (USB).
Set FA Mode	0x06	<u>Parameters:</u> dataWordCount: 0x0 <u>Response:</u> commandStatus: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	Set the part permanently in "Fault Analysis" mode to return to NXP factory. ROM erases customer sensitive assets (key codes) stored in OTP. Sets Fault Analysis (FA or RMA) mode bit FA_MODE_EN in OTP field. So, that customers could ship these parts to NXP for FA/RMA testing.
Start DBG Session	0x07	<u>Parameters:</u> dataWordCount: 0x0 <u>Response:</u> commandStatus: status code dataWordCount[14:0]: 0x0	Start Debug Session This command instructs ROM to clean-up any peripherals (SysTick, UART, SPI, I2C, FlexSPI, SD/MMC, USB, MPU and SAU) initialized for ISP/ boot mechanisms. When program control is in ROM, debug access is disabled. On systems, when there is no valid image in boot media the

Table continues on the next page...

Table 611. DM-AP commands (continued)

Name	Command code	Parameter/ Response	Description
		newProtocol: 0x1	program control enters ISP loop and hence debug access is disabled. The external world has to use this command to indicate to ROM the intention of connecting to the debugger. ROM upon receiving the command cleans all peripherals and secrets before enabling debug access. After enabling debug access ROM enter while(1) loop.
Debug Auth. Start	0x10	<i>Parameters:</i> dataWordCount: 0x0 <i>Response:</i> dataWordCount[14:0]: 0x12C or 0x22C newProtocol: 0x1 data[]: DAC	Start Debug Authentication Protocol. ROM responds to debugger with <i>DAC (Debug Authentication Challenge)</i> message
Debug Auth. Response	0x11	<i>Parameters:</i> dataWordCount: 0x12C data[]: DAR <i>Response:</i> commandStatus[15:0]: status code dataWordCount[14:0]: 0x0 newProtocol: 0x1	Debug Authentication Response

60.5.2 Response

The first word transmitted in a response is a header word containing the command status and number of following data words. The command response can be read 32-bits at a time through the RETURN register. The initial 32-bits contains the response header as shown in the table that follows. When reading a response greater than 32-bits, the debugger should write the ACK_TOKEN to the REQUEST register after every read of the RETURN register until the full response packet is received.

NOTE

To support legacy LPC command and response value, Bit_31 in header will be used to indicate that the response follows new protocol structure.

Table 612. Response register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandStatus[7:0]	commandStatus[15:8]	dataWordCount[7:0]	dataWordCount[14:8] new_protocol[15]
1	<i>data...</i>			

The C structure definition for a response is as follows:

```

struct dm_response {
    uint16_t commandStatus;
    uint16_t dataWordCount;
    uint32_t data[];
};

```

60.5.2.1 DM-AP response codes

Response codes for DM-AP commands are listed below. These commands can be read from the RETURN register.

Table 613. Register overview: DM-AP legacy return codes

Return code	
0x0000 0000	Command succeeded.
0x0010 0001	Debug mode not entered. This is returned if other commands are sent prior to the "Enter DM-AP" command.
0x0010 0002	Command not recognized. A command was received other than the ones defined above.
0x0010 0003	Command failed.

This table lists the return codes for the DM-AP new protocol.

Table 614. Register overview: DM-AP new protocol return codes

Return code	
0x8000 0000	Command succeeded.
0x8010 0001	Debug mode not entered. This is returned if other commands are sent prior to the "Enter DM-AP" command.
0x8010 0002	Command not recognized. A command was received other than the ones defined above.
0x8010 0003	Command failed.

60.5.3 ACK_TOKEN

- When command has parameters the debugger waits for ACK_TOKEN (sent through RETURN register) before sending next 32-bit value.
- Similarly when response packet has data to send back to debugger. Boot ROM waits for debugger to send ACK_TOKEN (sent through REQUEST register) before sending next 32-bit value.
- Upper 16-bits are set by receiving end with number of remaining words expected.
- Lower 16-bits are always set to 0xA5A5.

Table 615. ACK_TOKEN register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0xA5	0xA5	remainCount[7:0]	remainCount[15:8]

The C structure definition for a ACK_TOKEN is as follows:

```

struct dm_ack_token {
    uint16_t token; /* always set to 0xA5A5 */
    uint16_t remainCount; /* count of remaining word */
};

```

60.5.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW register. The state machine hardware will prevent further communication in any direction once an overrun has occurred. Once such an error occurs, the debugger needs to start with a new re-synchronization request to clear the error flag.

60.6 Debug authentication

The fundamental principles of debugging, which require access to the system state and system information, conflict with the principles of security, which require the restriction of access to assets. Thus, many products disable debug access completely before deploying the product. This causes challenges for product design teams to do proper Return Material Analysis (RMA). To address these challenges, the chip offers a debug authentication protocol as a mechanism to authenticate the debugger (an external entity) has the credentials approved by the product manufacturer before granting debug access to the device.

The debug authentication scheme for this chip is a challenge-response scheme that assures that only a debugger in possession of the required debug credentials can successfully authenticate over the debug interface and access restricted parts of the device. This protocol is divided into steps as shown and described below:

1. The debugger initiates the Debug Mailbox message exchange by setting the CSW[RESYNCH_REQ] bit and CSW[CHIP_RESET_REQ] bit of DM-AP.
2. The debugger waits (minimum 30 ms) for the devices to restart and enter debug mailbox request handling loop.
3. The debugger sends Debug Authentication Start command (command code 0x10) to the device.
4. The device responds back with Debug Authentication Challenge (DAC) packet based on the debug access rights pre-configured in OTP fields, which are collectively referred as Device Credential Constraints Configuration (DCFG_CC). The response packet also contains a 32 bytes random challenge vector.
5. The debugger responds to the challenge with a Debug Authentication Response (DAR) message by using an appropriate debug certificate, matching the device identifier in the DAC. The DAR packet contains the debug access permission certificate, also referred as Debug Credential (DC), and a cryptographic signature binding the DC and the challenge vector provided in the DAC.
6. The device on receiving the DAR, validates the contents by verifying the cryptographic signature of the message using the debugger's public key present in the embedded the Debug Credential (DC). On successful validation of DAR, the device enables access to the debug domains permitted in the DC.

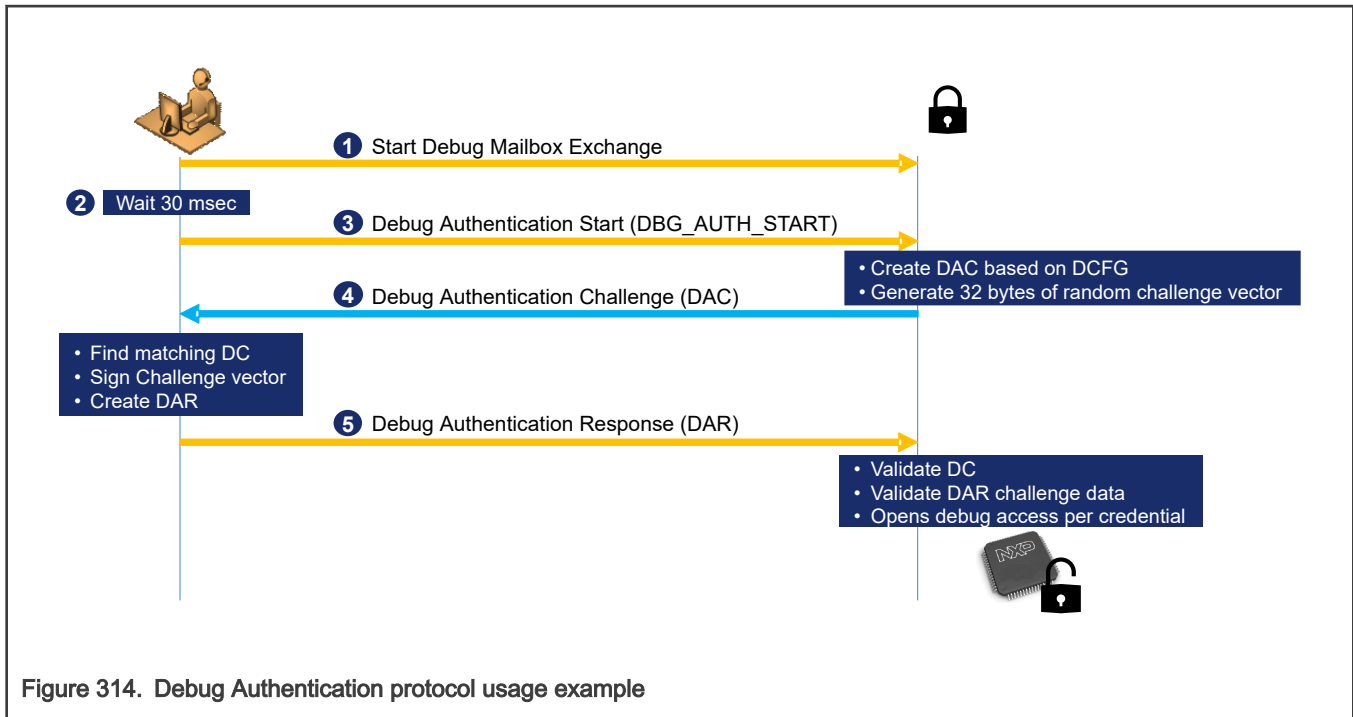


Figure 314. Debug Authentication protocol usage example

60.6.1 Debug Access Control Configuration

The boot code present in the ROM handles the device side of the Debug authentication process. However, the debug access control rights and security policies can be configured by programming the following configuration fields which are collectively referred as Device Configuration for Credential Constraints (DCFG_CC), present in One-Time-Programmable (OTP) fuses.

- **DCFG_VER:** This field controls the cryptographic primitives used during authentication.
- **DCFG_ROTID:** This field defines the Root of trust identifier (ROTID). The ROTID field is used to bind the devices to specific Certificate Authority (CA) keys issuing the debug credentials. These CA keys are also referred as Root of Trust (RoTK) keys.
- **DCFG_UUID:** Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set only DC with matching device UUID can unlock the debug access.
- **DCFG_CC_SOCU:** This configuration field specifies access rights to various debug domains.
- **DCFG_VENDOR_USAGE:** This field can be used to define a vendor specific debug policy use case such as DC revocations or department identifier. It is recommended to use this field for revocation of already issued debug certificates.

These fields should be programmed as part of OEM provisioning process.

60.6.1.1 Protocol Version (DCFG_VER)

The chip supports two instantiations of Debug authentication protocol versions, which are defined based on the different-sized RSA keys.

- **Version 1.0:** Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 2048-bit modulus and a 32-bit exponent.
- **Version 1.1:** Uses RSASSA-PKCS1-v1_5 signature verification using RSA keys with 4096-bit modulus and a 32-bit exponent.

To enforce usage of RSA4096 keys, set the RSA4K field in BOOT_CFG0, OTP word 96 to 0x1.

Note, both debug authentication certificates and image signing certificates use the same Root of Trust keys (RoTK). Hence when this field is set the secure boot image signing key certificate chain should also use RSA 4096-bit keys.

60.6.1.2 Root of Trust Identifier (DCFG_ROTID)

The Root of Trust Identifier used in debug authentication protocol is composed of two elements.

- A 256-bit cryptographic hash (SHA256) over the Root of Trust Keys Table. This is same as Root Keys Table Hash (RKTH) field referred in Secure boot ROM chapter. RKTH is a 32-byte SHA-256 hash of SHA-256 hashes of up to four root public keys.
 - OTP words 120 to 127 specifies the ROTID.
- Revocation bits for the four root keys in the table.
 - REVOKE_ROOTKEY field (bits 0 to 3) in OTP word SEC_BOOT_CFG[5] (OTP word 103).

60.6.1.3 Enforce UUID checking (DCFG_UUID)

Controls whether to enforce UUID check during Debug Credential (DC) validation. If this field is set then only a DC containing a UUID attribute that is an exact match to the device can unlock the debug access.

This field can be set by programming FORCE_UUID_MATCH (fuse bit 24) in OTP word 95.

This device-specific constraint, if enabled, is in addition to all the other constraints defined and enforced by the authentication protocol.

60.6.1.4 Credential Constraints (DCFG_CC_SOCU)

The DCFG_CC_SOCU is a configuration that specifies debug access restrictions per debug domain. These access restrictions are also referred as constraint attributes in this section. The debug subsystem is sub-divided in to multiple debug domains to allow finer access control. [Table 617](#) shows debug domains and their corresponding control bit position in DCFG_CC_SOCU. Which is logically composed of two components:

- **SOCU_PIN:** A bitmask that specifies which debug domains are predetermined by device configuration.
- **SOCU_DFLT:** Provides the final access level for those bits that the SOCU_PIN field indicated are predetermined by device configuration.

The following table shows the restriction levels:

Table 616. Access restriction levels

Restriction Level	SOCU_PIN [n]	SOCU_DFLT [n]	Description
0	1	1	Access to the sub-domain is always enabled. This setting is provided for module use case scenario where DCFG_CC_SOCU_NS would be used to define further access restrictions before final deployment of the product. See Module use case with OEM tier 1 and tier 2 lifecycle states for details.
1	0	0	Access to the sub-domain is disabled at startup. But the access can be enabled through debug authentication process by providing appropriate Debug Credential (DC) certificate.
2	0	1	Illegal setting. Part may lock-up if this setting is selected.
3	1	0	Access to the sub-domain is permanently disabled and cannot be reversed. This setting offers the highest level of restriction.

The supported Debug domains are shown below:

Table 617. CC List Table

Bit	Symbol	Description
0	NIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined non-secure domain of CPU0.
1	DBGEN	Controls invasive debugging of TrustZone for Arm8-M defined non-secure domain of CPU0.
2	SPNIDEN	Controls non-Invasive debugging of TrustZone for Arm8-M defined secure domain of CPU0.
3	SPIDEN	Controls invasive debugging of TrustZone for Arm8-M defined secure domain of CPU0.
4	TAPEN	Controls TAP (Test Access Point) controller used for structural integrity testing of silicon by NXP as part of Return Material Analysis (RMA).
5	DSP_DBGEN	Controls invasive debugging of Fusion DSP.
6	ISP_CMD_EN	Controls whether ISP boot flow DM-AP command (command code: 0x05) can be issued after authentication.
7	PMCMMD_EN	Controls whether permanent modification DM-AP commands such as Bulk Erase (command code: 0x02) and Set FA Mode (command code: 0x06), can be issued through after authentication.
31:8		Reserved

The following OTP fields are provided to define the restriction levels.

- DCFG_CC_SOCU (OTP word 95) & DCFG_CC_SOCU_AP (OTP word 104)
- DCFG_CC_SOCU_NS (OTP word 100)
 - This OTP word can be used to define further restrictions in scenarios where OEM Tier1 and OEM Tier 2 product life cycle states are used. See [Module use case with OEM tier 1 and tier 2 lifecycle states](#) for further details. These fields can be used to increase the restriction level specified in DCFG_CC_SOCU (OTP word 95) but cannot be used to reduce the restriction level.

Since these fields are security critical, they are constructed with three-fold protection mechanism to provide resistance from several side channel attacks.

- The OTP words itself are protected by ECC mechanism built-in OTP controller to catch any over programming hacks. These words can be programmed only once and all 32-bits together only.
- In addition to ECC check, the CRC8 checks offers second level of protection from side-channel glitch attacks during read transaction from OTP fuses.
- A third layer of protection is provided through redundant OTP word DCFG_CC_SOCU_AP (OTP word 104) which should be programmed with exact anti-pole (inverse value) value of DCFG_CC_SOCU (OTP word 95). Any mis-match between DCFG_CC_SOCU and DCFG_CC_SOCU_AP is deemed as an attack and debug is disabled permanently.

Table 618. Layout of DCFG_CC_SOCU (OTP word 95) and DCFG_CC_SOCU_NS (OTP word 100)

Bits	Symbol	Description
7:0	CRC8	CRC-8/ITU of upper 3 bytes (bits 8 to 31). Since these fields are security critical, they are constructed with built-in integrity protection to protect from side channel glitch attacks. The lower byte (0 to 7 bits) of these OTP words should be written with CRC-8/ITU of upper 3 bytes (bits 8 to 31). This construction makes the probability of a successful glitch attack to flip the exact control bits extremely difficult. The CRC8 calculation should be based on $x^8 + x^2 + x + 1$ polynomial.

Table continues on the next page...

Table 618. Layout of DCFG_CC_SOCU (OTP word 95) and DCFG_CC_SOCU_NS (OTP word 100) (continued)

Bits	Symbol	Description
		Polynomial=0x07, initial value= 0x00, XorOut=0x55.
15:8	CC_SOCU_DFLT	Defines the default access rights for the debug domains whose corresponding. If a bit is set, access is allowed. Otherwise access is denied.
23:16	CC_SOCU_PIN	Controls non-Invasive debugging of TrustZone for Arm8-M defined secure domain of CPU0.
24	FORCE_UUID_M ATCH	Force UUID matching.
30:25	—	Reserved
31	DEV_TEST_EN	Enable test mode. This bit should be clear in OTP words. But during development to test different DCFG_CC_SOCU settings without programming OTP words, developers can write the values to shadow registers corresponding to OTP words (95 & 104) with this bit set.

60.6.1.5 DCFG_VENDOR_USAGE

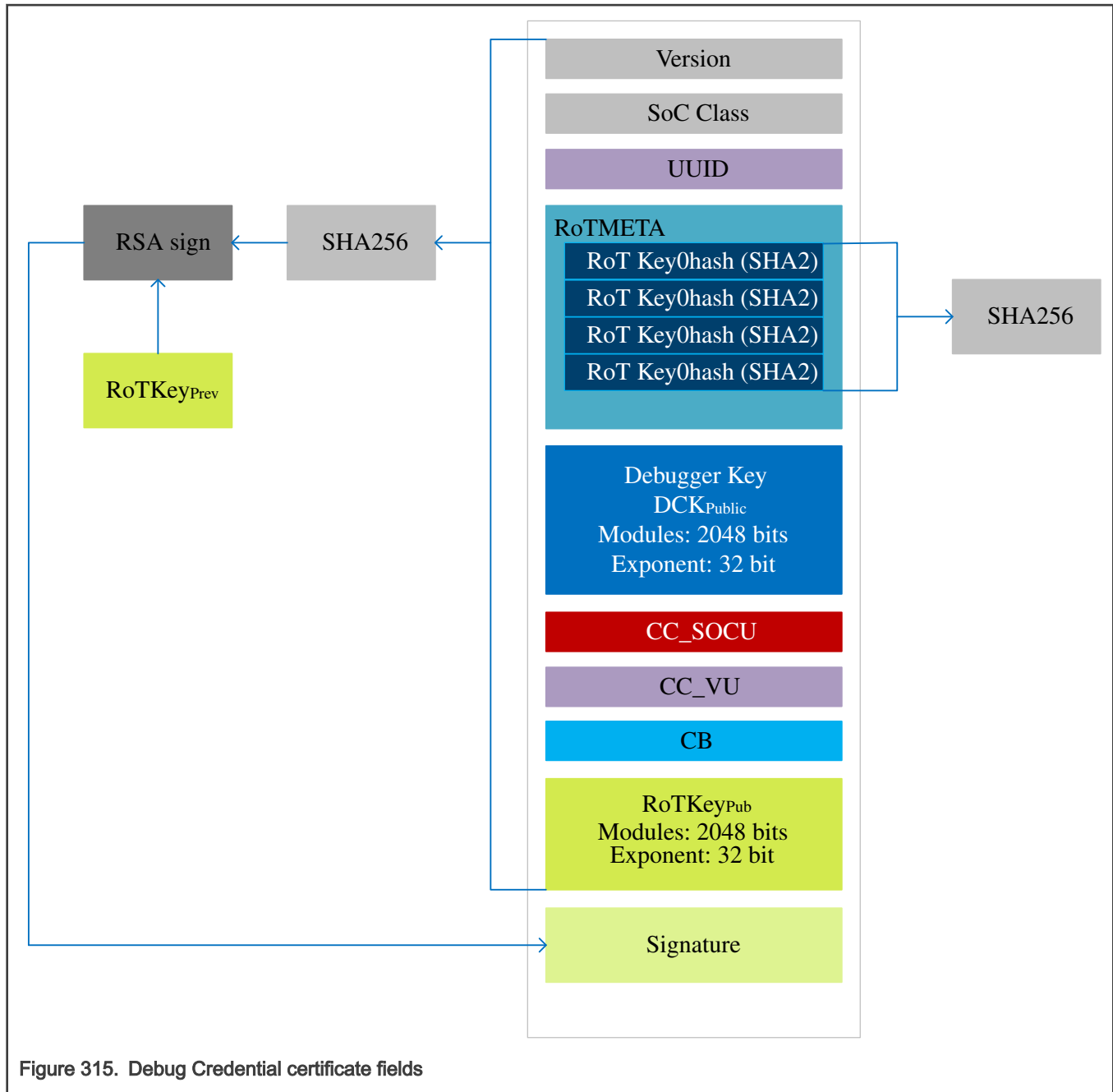
This field can be used to define vendor specific debug policy use case such as Debug Credential (DC) certificate revocations or department identifier or model identifier. It is recommended to use field for revocation of already issued debug certificates. During Debug Authentication Response (DAR) processing the device checks that the value specified in Vendor Usage field of DC matches exactly the value programmed in DCFG_VENDOR_USAGE fields of device configurations.

This chip provides a 16-bit redundant OTP word called DAP_VENDOR_USAGE (OTP Word 103). A redundant OTP word can be programmed 1 bit at a time. It is recommended to use this field to revoke DC certificates issued till that point.

60.6.2 Debug Credential Certificate (DC)

By prior construction, the debugger should already have a DCK (Debug Credential Key). The public key part of this key pair is used to represent the identity of the debugger through the creation of a DC, which binds that public key to some usage attributes, and is then authorized/signed by the vendor's RoT.

Total DC size is 940 bytes (v1.0) or 1708 bytes (v1.1).



The data structure is represented as a packed binary concatenation of its component fields as shown in the list below:

Table 619. Debug Credential certificate fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version. - Set 0x00010000 for v1.0, which uses RSA2048 keys - Set 0x00010001 for v1.1, which uses RSA4096 keys	4

Table continues on the next page...

Table 619. Debug Credential certificate fields (continued)

Name	Offset	Description	Size in bytes
SOCC	0x004	SoC class specifier. Always set this field to 0x0000 0000.	4
UUID	0x008	Unique device identifier, in case this certificate is used on a device configured for UUID-matching. If UUID matching is enabled certificate is restricted to a specific device, otherwise certificate is enabled for whole SoC Class.	16
ROTMET A	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing this certificate. SHA256 hashes of four RoT public keys should be set in this field. Note, on this device same RoT keys are used for certifying image signing keys and debug keys.	128
DCK_MO D	0x098	Modulus value of Debugger public key (DCKpub). - For version v1.0, the modulus is 2048-bit (256 bytes). - For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
DCK_EX P	0x198 or 0x298	The 32-bit exponent value of Debugger public key (DCKpub).	4
CC_SOC U	0x19C or 0x29C	SoC specific Credential Constraints. Specifies the debug access rights allowed for this certificate holder.	4
CC_VU	0x1A0 or 0x2A0	Vendor Usage. Should match DCFG_VENDOR_USAGE field in Device Configuration for Credential Constraints (DCFG_CC). See DCFG_VENDOR_USAGE . It can be used to revoke Debug Certificates.	4
CB	0x1A4 or 0x2A4	Credential beacon that vendor has associated with this DC. Only lower 16-bits of this field are effective. This field can be used to extend the Debug Authentication process. When a non-zero value is used in this field ROM differs opening debug access to user application. The result of the authentication process is written to DBG_FEATURES register while the user application after doing its extended processing, such as clean-up of critical keys & Secrets, should copy the value to DBG_FEATURES_DP register to enable the debug access. - To aid user application ROM stores beacon values in DBG_AUTH_SCRATCH register (0x40002FC0) - Lower 16 bits of the register contains Credential Beacon (CB). - Upper 16 bits of the register contains Authentication Beacon (AB) present in DAR packet.	4
RoTK_M OD	0x1A8 or 0x2A8	Modulus value of Root of Trust Vendor public key used for signing this certificate. - For version v1.0, the modulus is 2048-bit (256 bytes). - For version v1.1, the modulus is 4096-bit (512 bytes).	256 or 512
RoTK_E XP	0x2A8 or 0x4A8	The 32-bit exponent value of Root of Trust Vendor public key used for signing this certificate.	4
SIG	0x2AC or 0x4AC	A cryptographic signature by the RoT over the eight previous fields. This ensures the DC is “blessed” by the Vendor for use by the debugger.	256 or 512

Table continues on the next page...

Table 619. Debug Credential certificate fields (continued)

Name	Offset	Description	Size in bytes
		• 256 bytes (v1.0) or 512 bytes (v1.1), as calculated below. • SIG(RoTpriv,HASH(DC :: 1 DC :: 2 ... DC :: 8)) • SHA256 is used for Hash function. • RSASSA-PKCS1-v1_5 is used for SIG function. • RSA signature scheme from PKCS1 specification v1.5 (RFC 2313).	

60.6.3 Debug Authentication Challenge (DAC)

The debug authentication protocol begins with a DAC (Debug Authentication Challenge) message, issued by the device to the debugger. Total message size is 104 bytes.

From debug authentication protocol perspective, what matters is that the debugger selects a Debug Credential (DC) certificate that will successfully authenticate, based on the constraints provided in the DAC, and will provide the required debug behavior post-authentication (for example, whether to debug secure world, with the desired credential beacon). If such a credential cannot be found, the debugger should report a corresponding error to the user.

Note: The debugger must also be able to produce signatures using the private key corresponding to the selected DC, so that any credential store can manage this association between credentials and the corresponding private keys.

The named elements of this message are shown in this figure.

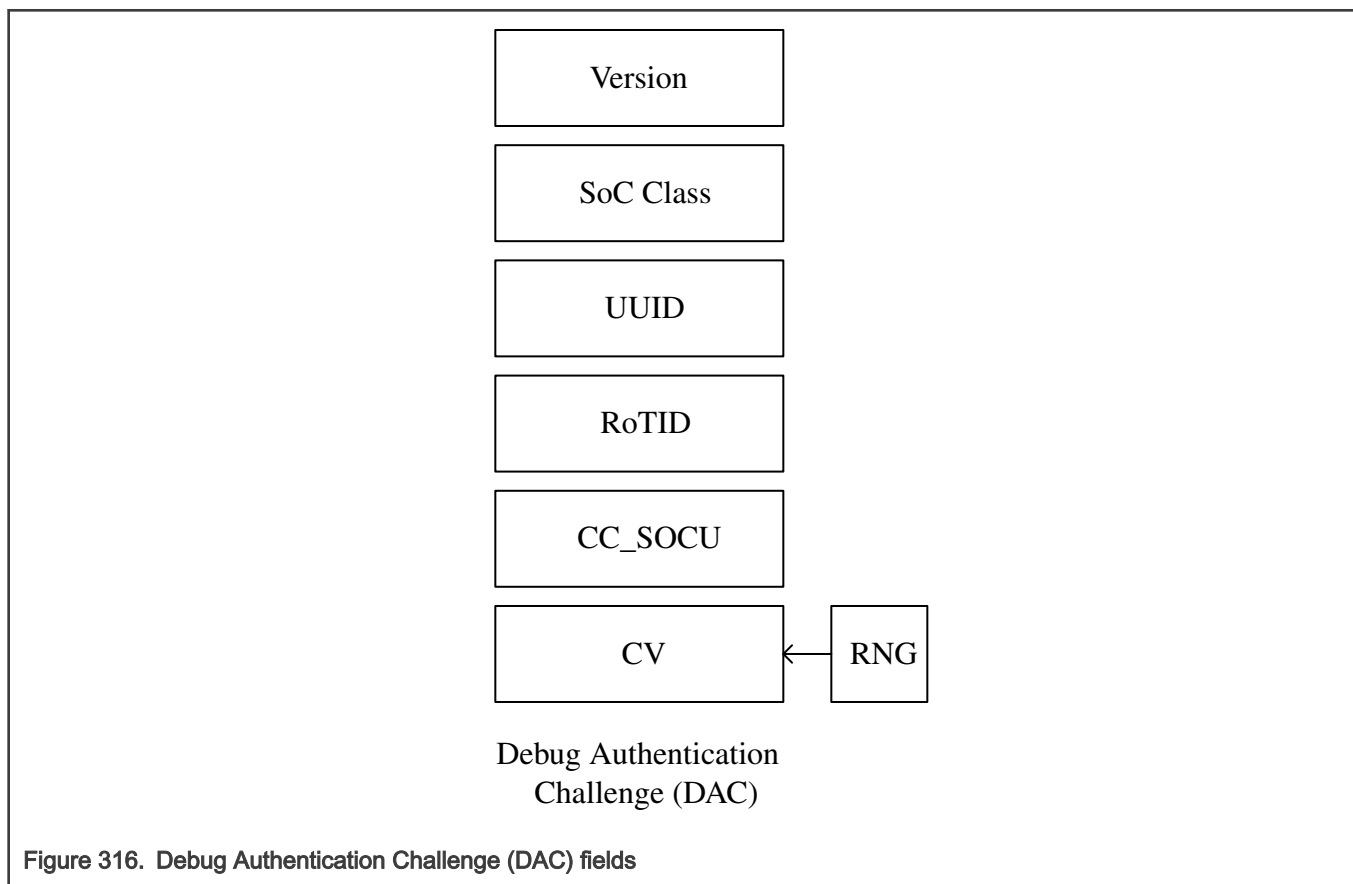


Table 620. Debug Authentication Challenge (DAC) fields

Name	Offset	Description	Size in bytes
VERSION	0x000	Identifies the Debug Authentication protocol version The value returned in this field is set based on the RSA4K field is set to 0x00010000 indicating protocol version v1.0, when DCFG_RSA4K is set to 0. 0x00010001 for v1.1, which uses RSA4096 keys.	4
SOCC	0x004	SoC class specifier Always sets this field to 0x0000 0000.	4
UUID	0x008	Unique device identifier If UUID matching is enabled in DCFG_CC_SoCU, then device will include its UUID in the challenge packet. Or else this field is set to zeros.	16
ROTID	0x018	Meta data used for authenticating Certificate Authority key (a.k.a. RoT key) used for signing DC certificate. SHA256 hashes of four RoT public keys would be set in this field. Check Root of Trust Identifier (DCFG_ROTID) for details. Note, on this device same RoT keys are used for certifying image signing keys and debug keys. 32 bytes RKTH value 4 bytes containing revocation flags. Only least significant 4 bits are used.	36
CC	0x3C	Credential Constraints Specifies the debug access rights allowed for this certificate holder. - A 32-bit SOCU_PIN value. See Table 617 for bit encoding of this field. - A 32-bit SOCU_DFLT value. See Table 617 for bit encoding of this field. - A 32-bit DCFG_VENDOR_USAGE value. Only the lower 16 bits are used.	12
CV	0x48	Challenge Vector generated by the device Device provides 32 bytes random value generated using TRNG block.	32

60.6.4 Debug Authentication Response (DAR)

Before the debugger can formulate a response to the challenge, it should perform some checks to confirm correctness of VER, SoCC, UUID, RoTID and CC; it should find a matching DC.

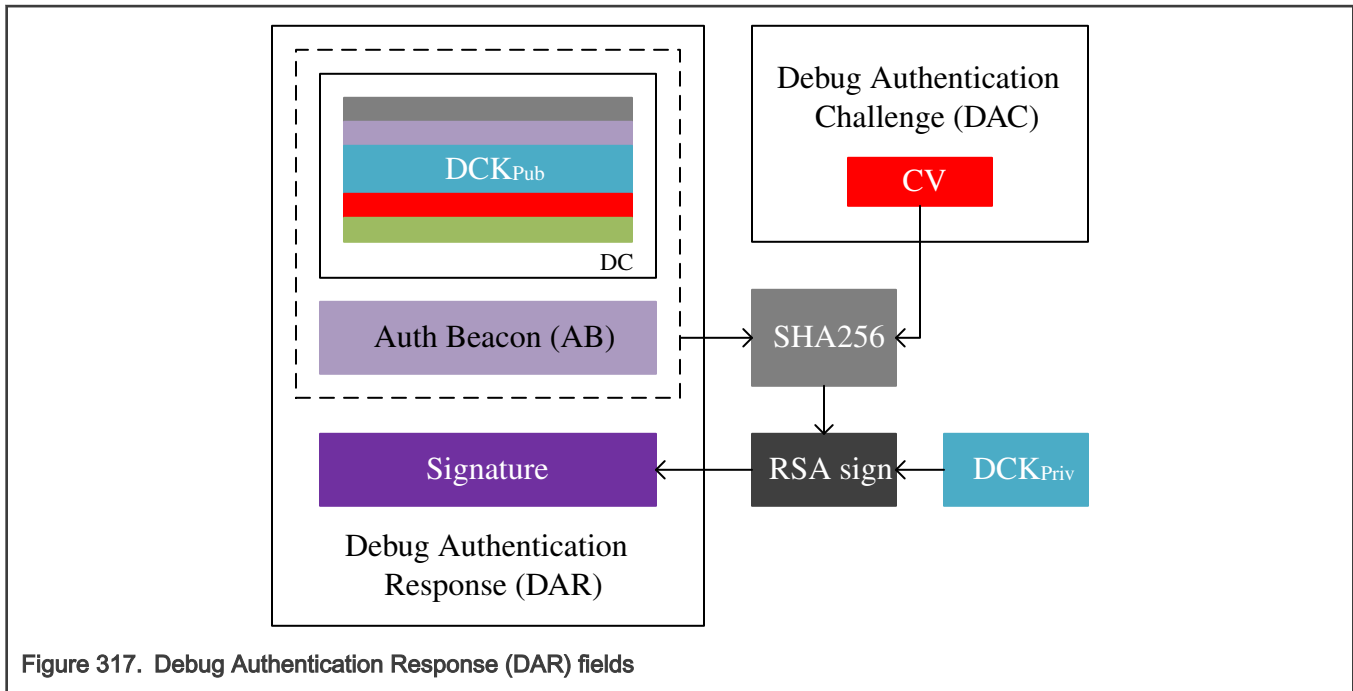


Figure 317. Debug Authentication Response (DAR) fields

Table 621. Debug Authentication Response (DAR) fields

Name	Offset	Description	Size in bytes
DC	0x000	DC, provides the debugger's credential, RoT public key and more, described in Debug Credential (Certificate). See Debug Credential Certificate (DC) 940 bytes (v1.0) or 1708 bytes (v1.1)	940 or 1708
AB	0x3AC or 0x6AC	AB, the Authentication Beacon provided and signed by the debugger during the authentication process. Refer to the Credential Beacon (CB) field in Debug Credential Certificate (DC) structure for details.	4
SIG	0x3B0 or 0x6B0	A cryptographic signature by the debugger that binds the previous two fields with the challenge vector from the DAC. SIG = RSA_SIGN (DCKpriv, SHA256(DAR::DC II DAR::AB II DAR::CV)) - Uses the private key corresponding to the public key (DCK) of the selected DC (proves that debugger has possession of debugger private key). - RSASSA-PKCS1-v1_5 is used for SIG function.	256 or 512

60.6.5 Device processing the DAR

The device Boot ROM will process the DAR received from the debugger. As part of the validation process, the device will:

1. Verify the DC: Validate DC version, SoCC, UUID, RoT, VU, and DC signature.
2. Verify that the DAR has a valid signature that binds it to the CV from the DAC.

If all the steps are successfully completed, it can be deduced that:

- The debugger possesses the private key corresponding to the vendor/RoT-signed credential.

- The credential satisfies the constraints specified in the device configuration.
- The response of the debugger to the challenge from the device is produced and signed in response to the challenge (because of its cryptographic dependency on the challenge vector). The response is not replayed from a previous authentication where a different challenge vector is used.

After completion of processing the DAR:

If authentication is successful, Debug Access will be granted.

If authentication fails, no special response is issued but further debug access request will be ignored and device will enter in a failure loop.

60.6.5.1 Successful authentication

ROM executes the following steps after successful debug authentication:

1. ROM determines the final enable states of the debug ports based on pinned state from DCFG_CC_SOCU and the DC::CC fields.
2. ROM evaluates part enables using the following logic:
 - Uses pinned states based on DCFG_CC_SOCU and DCFG_CC_SOCU_NS OTP words
 - Evaluates SOCUPIN and SOCUDFLT
 - Evaluates debug port enables for ports which are not pinned using authentication protocol
 - $\text{Debug\_State} = (\text{SOCUPIN} \& \text{SOCUDFLT}) \mid (! \text{SOCUPIN} \& \text{DC::CC\_SOCU})$
 - Enables debug ports for bits which are set in above evaluation
3. The Debug Mailbox handler allows the following commands only if the enable bit is set in the final evaluation of Debug_State.
 - Handle 'ENTER_ISP_MODE' command only if default ISP_CMD_EN bit is set in Debug_State.
 - Handles 'SET_FA_MODE' and 'ERASE_FLASH' commands only if default FA_CMD_EN bit is set in Debug_State.
4. ROM stores the beacons in the write lockable register DBG_AUTH_SCRATCH.
 - $\text{DBG\_AUTH\_SCRATCH}[15:0] = \text{DAR::DC::CB}[15:0]$
 - $\text{DBG\_AUTH\_SCRATCH}[31:16] = \text{DAR::AB}[15:0]$
5. On receiving an EXIT_DBG_MB command, ROM exits the debug mailbox handler loop and continues normal boot flow.

60.6.6 Debug Authentication use cases

60.6.6.1 Return Material Analysis (RMA) use case

The diagram shows RMA flow using debug authentication scheme, where a debug credential certificate is issued for each field technician.

1. Vendor generates RoT key pairs and programs the device with SHA256 hash of RoT public key hashes before shipping.
2. Field technician generates his own key pair and provides public key to vendor for authorization.
3. Vendor attests the field technician's public key. In the debug credential certificate, vendor assigns the access rights.
4. End customer having issues with a locked product takes it to field technician.
 - Field technician uses his credentials to authenticate with device and un-locks the product for debugging.

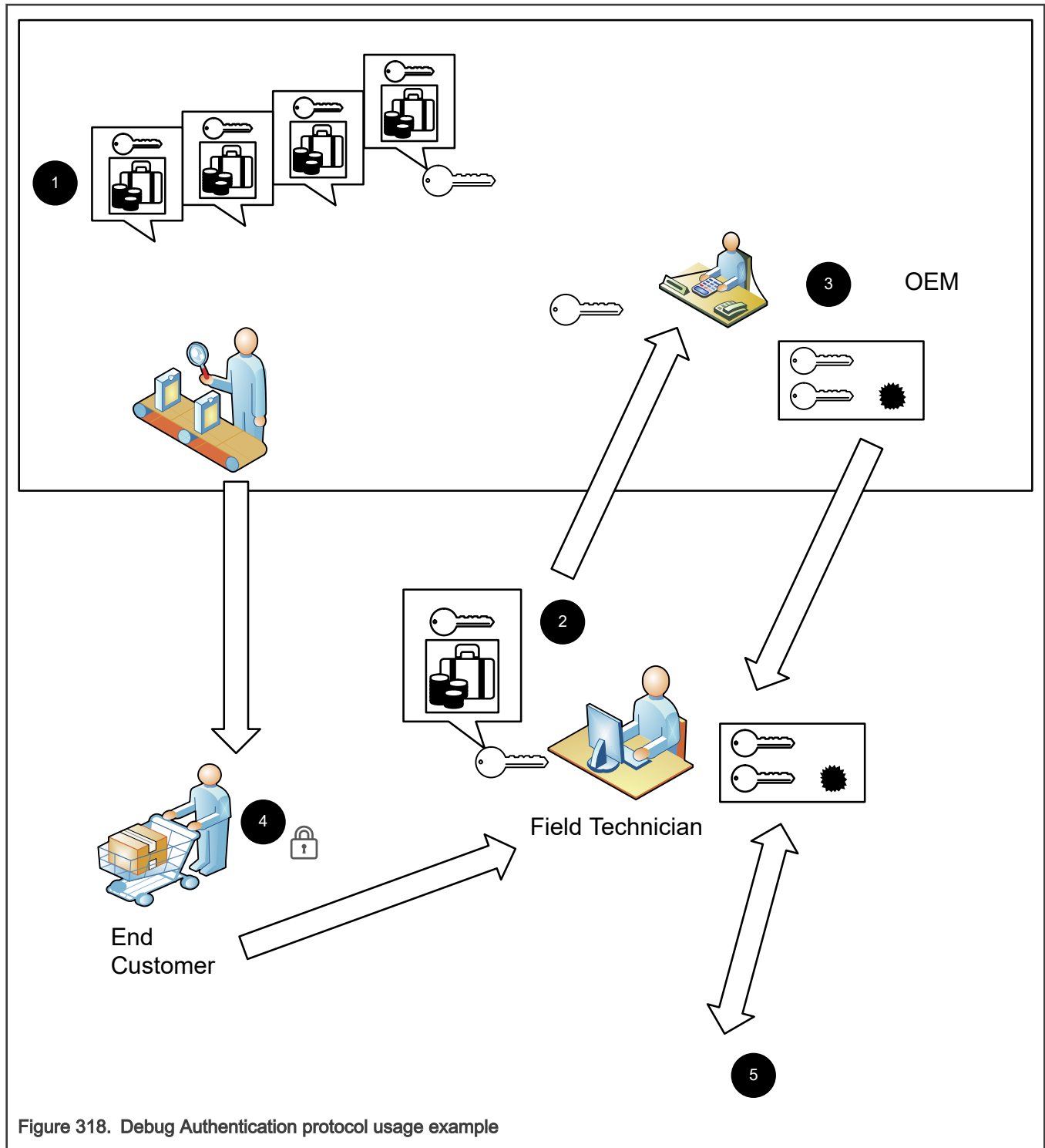


Figure 318. Debug Authentication protocol usage example

60.6.6.2 Module use case with OEM tier 1 and tier 2 lifecycle states

The DCFG_CC_SOCU_NS is provided to allow module-maker and OEM using the module to implement a tiered protection approach.

- The module maker who is referred to as a Tier-1 developer can develop secure code and define access rights to the module using DCFG_CC_SOCU.

- Configuration can be such that debug access to the secure module is blocked but non-secure debug is always allowed.
- Once the module is ready, a Tier-1 developer can release the module to OEM (a.k.a. tier-2 developer), but block debug access to secure mode and enable debug access to non-secure mode.
- Tier-2 developer can develop non-secure module and extend access rights configuration to that module using DCFG_CC_SOCU_NS.

60.6.7 Debug port states before entering user code

ROM executes the following steps before entering into user code or failure loop.

1. ROM checks if the CS_PROTCPU0 is set to enable value (0x12345678).
 - If yes, ROM entry is due to non-POR reset. Therefore ROM copies the DBG_FEATURES register value to the DBG_FEATURES_DP register and exits to user code/failure loop.
2. If the DBG_AUTH_SCRATCH register is equal to zero:
 - Check PINNED_TAPEN bit in DCFG_CC_SOCU.
 - Check PINNED_DSPEN bit in DCFG_CC_SOCU.
 - If set, unlock value (0x12345678) is written to the CS_PROTCPU1 register.
 - If not set, disable port value (0x0) is written to the CS_PROTCPU1 register.
 - Check states of PINNED_NIDEN, PINNED_DBGEN, PINNED_SPNIDEN, PINNED_SPIIDEN bits in DCFG_CC_SOCU.
 - If any of the bits is set, the unlock value (0x12345678) is written to the CS_PROTCPU0 register.
 - For bits which are set, enable value (b'10) is written in corresponding bit fields of the DBG_FEATURES register.
 - For bits which are not-set, disable value (b'01) is written in corresponding bit fields of the DBG_FEATURES register.
 - If all bits are zero, then disable port value (0x0) is written to the CS_PROTCPU0 register.
 - The DBG_FEATURES register value is copied to the DBG_FEATURES_DP register.
 - Debug control registers are write-locked by writing to the DBG_LOCKEN register.
 - Exits to user code/failure loop.
3. If the DBG_AUTH_SCRATCH is non-zero then the unlocking/enabling of the debug ports to user code is deferred.
 - Check DFLT_TAPEN bit in DCFG_CC_SOCU.
 - Check DFLT_CPU1EN bit in DCFG_CC_SOCU.
 - If set, then the CS_PROTCPU1 register is not written.
 - If not set, then disable port value (0x0) is written to the CS_PROTCPU1 register.
 - If all of the following bits are set in DCFG_CC_SOCU, then the CS_PROTCPU0 and DBG_FEATURES_DP registers are not written. Else the registers are written as per DCFG_CC_SOCU value and write access is locked by setting the DBG_LOCKEN register.
 DFLT_NIDEN, DFLT_DBGEN, DFLT_SPNIDEN, DFLT_SPIIDEN
 - In all cases ROM writes to the DBG_FEATURES register based on DCFG_CC_SOCU.

60.6.8 Test access port

The xxx_TAPEN OTP bits (DCFG_CC_SOCU_NS[PINNED_TAPEN] and DCFG_CC_SOCU_NS[DFLT_TAPEN]) control access settings for the Test Access Port (TAP). TAP is used with JTAG boundary scan, and used by NXP testers. The reason this TAP access is customer programmable is to support RMA analysis. If a customer wants NXP to be able to re-run production tests on the MCU at some point, say during a Failure Analysis (FA), the customer can leave the TAP access enabled. But for security reasons,

customers have control to disable TAP access. Even if the xxx_TAPEN OTP bits prevent TAP access, TAP access can be granted to NXP for returns using the Fault Analysis mode.

60.6.9 Fault Analysis (FA) mode

The ROM on this device has an FA Mode (SET_FA_MODE) command handler to enable the deletion of sensitive information (for example, Keys) before handing over the device to NXP for fault analysis. ROM allows the SET_FA_MODE command only when a corresponding flag in the Debug_State is set.

The FA_MODE sequence when activated by Boot ROM:

- Creates a new version of the OTP page.
- Sets ENABLE_FA_MODE word in the page to value 0xC33CA55A.
- Erases all KEYS and IVs in KEYSTORE Flash page.
- Flushes all temporary key registers.
- Blocks PUF indexes
- Opens all debug ports
- Enters while (1) loop

60.7 Signals

Serial Wire Debug is the default function on pins PIO2_25 and PIO2_26, allowing for debug through reset. The JTAG mode for boundary scan is selected by a certain combination of pin setting at reset.

Serial Wire Debug functions are integrated into the CM33 and the Fusion DSP. Trace on the CM33 is supported via the serial wire output only. Device boundary scan is also available.

[Table 622](#) indicate the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 622. Serial wire debug signals

Signal	I/O	Pin name	Description
SWCLK	In	PIO2_25	Serial Wire Clock. This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). At reset release, this pin is pulled down internally. SWCLK is the default function of this pin.
SWDIO	I/O	PIO2_26	Serial Wire Debug Data input/output. The SWDIO pin is used by an external debug tool to communicate with and control the part. At reset release, this pin is pulled up internally. SWDIO is the default function of this pin.
SWO	Out	PIO2_24 or PIO2_31	Serial Wire Output. The SWO pin optionally provides data from the ITM for an external debug tool to evaluate. SWO must be selected as a function on one of these pins prior to use.

The following setups are required to enable SWO output:

1. Write 0x0 to CLKCTL0_PFC0DIV to enable the trace clock divider.
2. Connect the SWO function to a pin. This could be function 1 of PIO2_24 or function 5 of PIO2_31.

The JTAG boundary pin functions are selected by hardware at reset. See [Table 623](#).

Table 623. JTAG boundary scan signals

Signal	I/O	Pin name	Description
TCK	In	PIO0_8	JTAG Test Clock. This pin is the clock for JTAG boundary scan when the JTAG mode active.
TMS	In	PIO0_9	JTAG Test Mode Select. The TMS pin selects the next state in the TAP state machine. This pin is used for JTAG boundary scan when the JTAG mode is active.
TDI	In	PIO0_10	JTAG Test Data In. This is the serial data input for the shift register. It is used for JTAG boundary scan when the JTAG mode is active. This pin has an internal pull-up and input receiver enabled at default.
TDO	Out	PIO0_11	JTAG Test Data Output. This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the JTAG mode is active
TRST	In	PIO0_7	JTAG Test Reset. The TRST pin can be used to reset the test logic within the debug logic. It is used for JTAG boundary scan when the JTAG mode is active. This pin has an internal pull-down and input receiver enabled at default.

The following Trace signals are available on the chip.

Table 624. Trace signals

Signal	I/O	Pin name	Description
TRACECLK	Out	PIO0_21 or PIO4_23	Trace Clock
TRACEDATA0	Out	PIO0_22 or PIO4_24	Trace Data 0
TRACEDATA1	Out	PIO0_23 or PIO4_25	Trace Data 1
TRACEDATA2	Out	PIO0_24 or PIO4_26	Trace Data 2
TRACEDATA3	Out	PIO0_25 or PIO4_27	Trace Data 3

60.8 Memory map and register definition

60.8.1 Debug register descriptions

60.8.1.1 Debug memory map

DEBUGGER_MAILBOX0 base address: 4010_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Command and status word (CSW)	32	RW	0000_0000h
4h	Request Value (REQUEST)	32	RW	0000_0000h
8h	Return Value (RETURN)	32	RW	0000_0000h
FCh	Identification (ID)	32	R	002A_0000h

60.8.1.2 Command and status word (CSW)

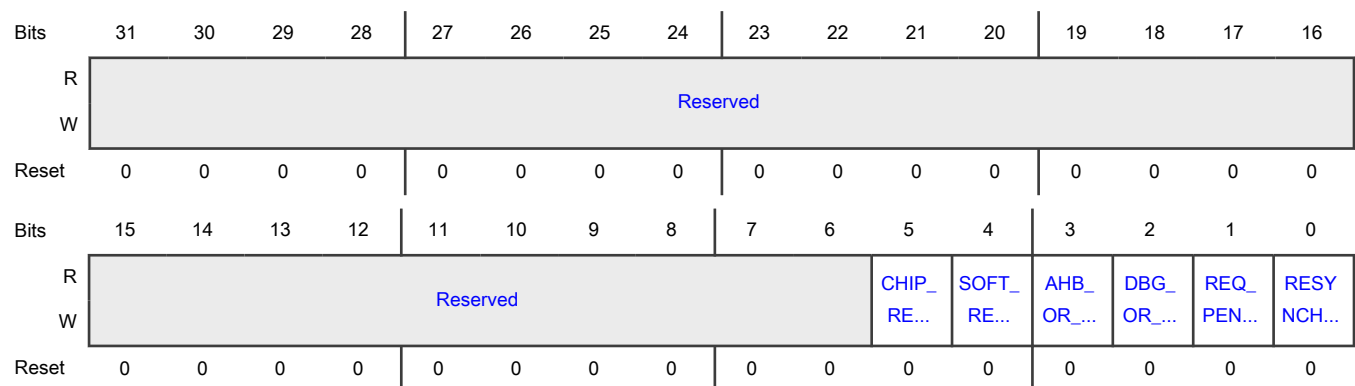
Offset

Register	Offset
CSW	0h

Function

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Diagram



Fields

Field	Function
31-6 —	Reserved
5 CHIP_RESET_ REQ	Chip Reset Request This write-only bit causes the device (but not the DM-AP) to be reset.
4	Soft Reset

Table continues on the next page...

Table continued from the previous page...

Field	Function
SOFT_RESET	The SOFT_RESET bit is write-only by the device and resets the DM-AP.
3 AHB_OR_ERR	AHB Overrun Error When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before the RETURN value was read by the debugger. 0b - No AHB Overrun Error 1b - AHB Overrun Error. An AHB overrun occurred.
2 DBG_OR_ERR	Debug Overrun Error When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before the REQUEST value was read by the device. 0b - No Debug Overrun error 1b - Debug Overrun Error. A debug overrun occurred.
1 REQ_PENDING	Request Pending A request is pending for the debugger: a value is waiting to be read from the REQUEST register. 0b - No Request Pending 1b - Request for Re-synchronization Pending
0 RESYNCH_REQ	Re-synchronization Request The debugger sets RESYNCH_REQ to requests a re-synchronization. 0b - No Request 1b - Request for re-synchronization

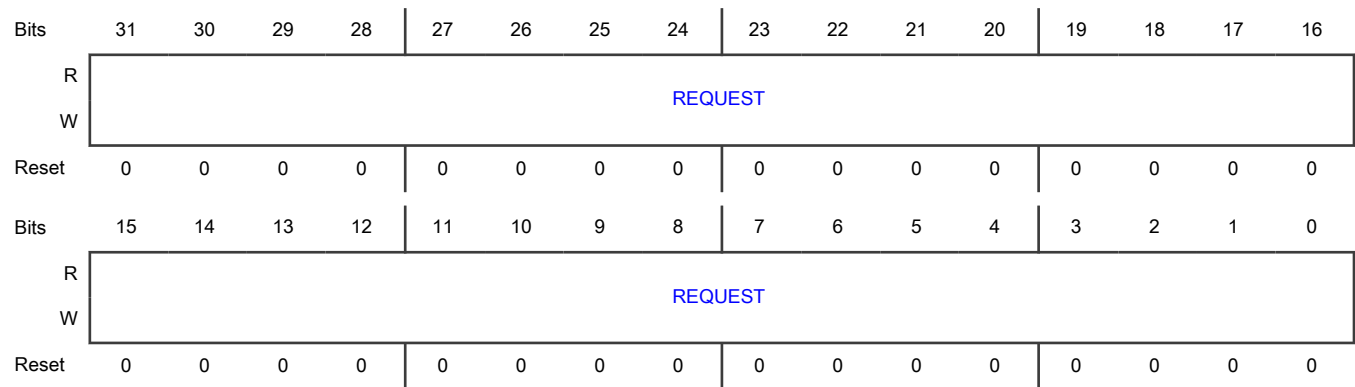
60.8.1.3 Request Value (REQUEST)

Offset

Register	Offset
REQUEST	4h

Function

The REQUEST register is used by a debugger to send action requests to the device.

Diagram**Fields**

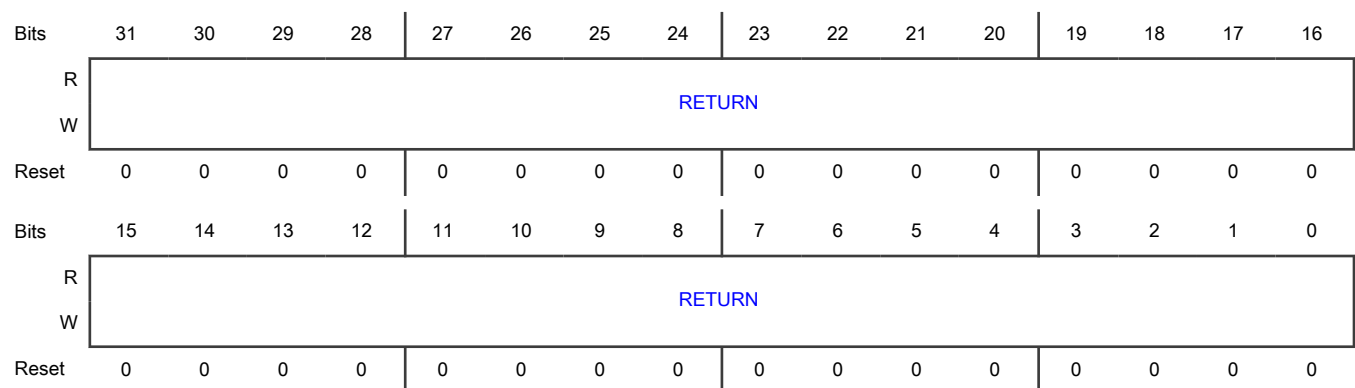
Field	Function
31-0 REQUEST	Request Value Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.

60.8.1.4 Return Value (RETURN)**Offset**

Register	Offset
RETURN	8h

Function

The RETURN register provides the responses from the device to the debugger.

Diagram

Fields

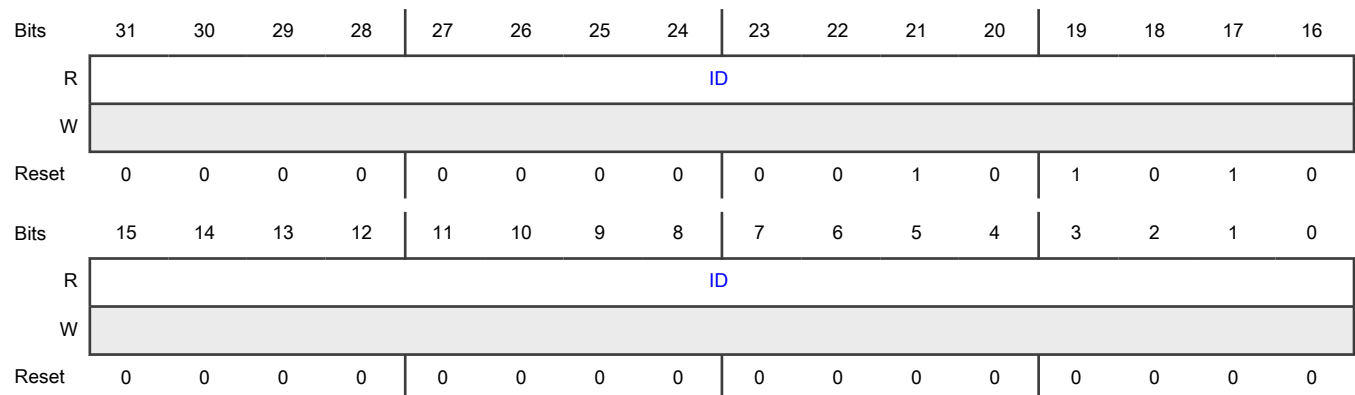
Field	Function
31-0 RETURN	Return Value This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available.

60.8.1.5 Identification (ID)**Offset**

Register	Offset
ID	FCh

Function

The ID register provides an identification of the DM-AP interface.

Diagram**Fields**

Field	Function
31-0 ID	Identification Value

Chapter 61

JTAG

61.1 Introduction

The JTAG block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAG block is communicated in serial format.

61.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAG) block. See the chip-specific configuration information as well as [Register description](#) for more information about the JTAG registers.

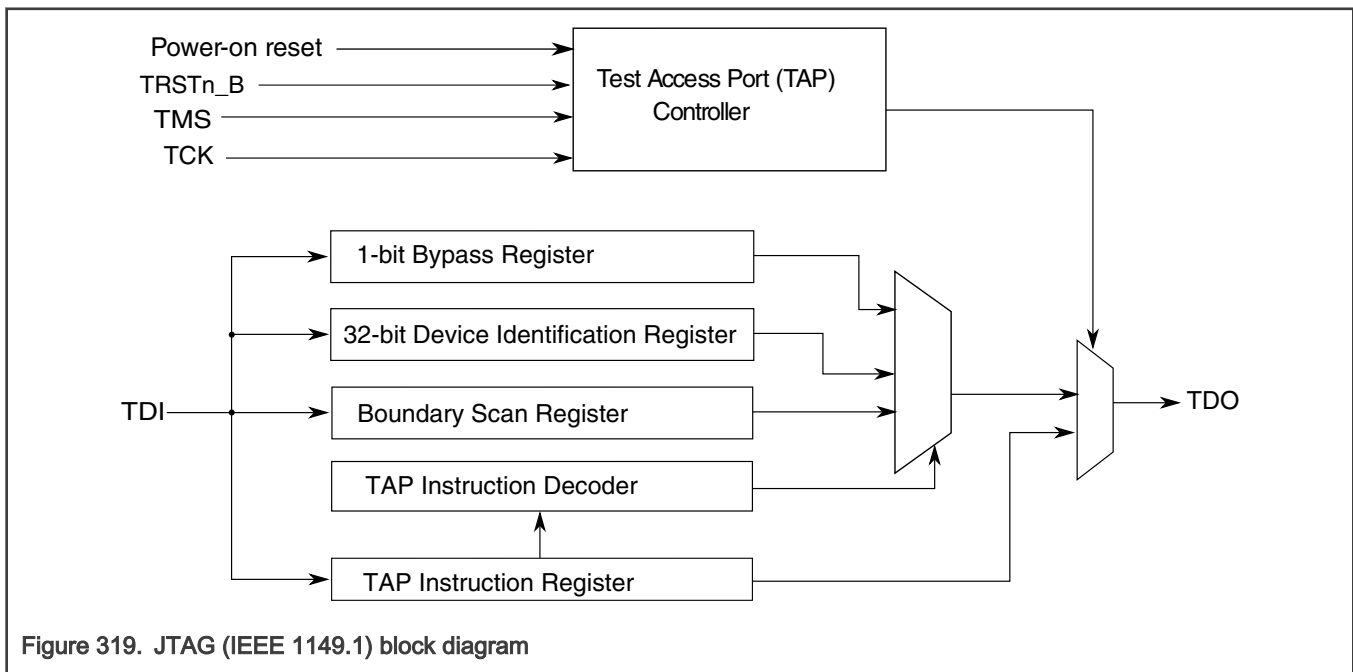


Figure 319. JTAG (IEEE 1149.1) block diagram

61.1.2 Features

The JTAG block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
 - Five pins (TDI, TMS, TCK, TRSTn_B and TDO)
- TRST_B input that provides reset control
- Instruction register that supports several IEEE 1149.1–2001 defined instructions as well as several public and private device-specific instructions (see [JTAG block instructions](#) for a list of supported instructions).
- Bypass register, boundary scan register, and device identification register. The size of the boundary scan register is 675
- TAP controller state machine that controls the operation of the data registers, instruction register, and associated circuitry

61.1.3 Modes of operation

The JTAG block uses TRSTn_B and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

61.1.3.1 Reset

The JTAG block is placed in reset when:

- Power-on reset is asserted.
- TRSTn_B is asserted
- TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state

Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or TRSTn_B results in asynchronous entry into the reset state.

When in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

61.1.3.2 IEEE 1149.1–2001 defined test modes

The JTAG block supports several IEEE 1149.1–2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register when the JTAG is enabled. Supported test instructions include EXTEST, BYPASS, HIGHZ, CLAMP, SAMPLE/PRELOAD.

Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic when the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the following instructions are active:

- BYPASS
- HIGHZ
- CLAMP

The functionality of each test mode is explained in more detail in [JTAG block instructions](#).

61.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAG block into bypass mode. When in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

61.2 External signal description

The JTAGC consists of a set of signals that connect to off-chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

NOTE

JTAG pins are High Z until JTAG Boundary Scan mode is entered.

Table 625. JTAG signal properties in Boundary Scan mode

Signal Name	Alternate Name	I/O	Function	Pin state
TCK	PIO_10	Input	Test clock	Weak pulldown

Table continues on the next page...

Table 625. JTAG signal properties in Boundary Scan mode (continued)

Signal Name	Alternate Name	I/O	Function	Pin state
TDI	PIO_11	Input	Test data in	High Z
TDO	PIO_11	Output	Test data out	High Z
TMS	PIO_9	Input	Test mode select	Weak pullup
TRST_B	PIO_7	Input/ Output	Test Reset	Weak Pullup

61.2.1 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

61.2.2 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

61.2.3 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is tristateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [TAP controller state machine](#).

61.2.4 Test mode select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

61.2.5 Test Reset (TRST_B)

Test Reset (TRST_B) is an optional active-low reset to the test logic, usually asynchronous, but sometimes synchronous. It resets the TAP controller's state machine..

61.3 Functional description

This section explains the JTAG functional description.

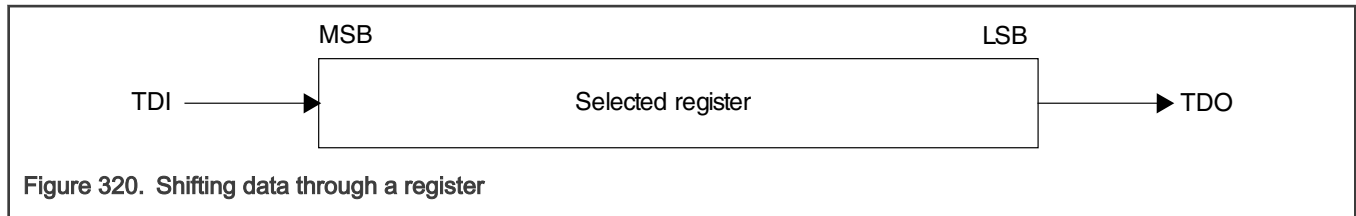
61.3.1 JTAG reset configuration

When in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

61.3.2 IEEE 1149.1-2001 (JTAG) TAP

The JTAG block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction.

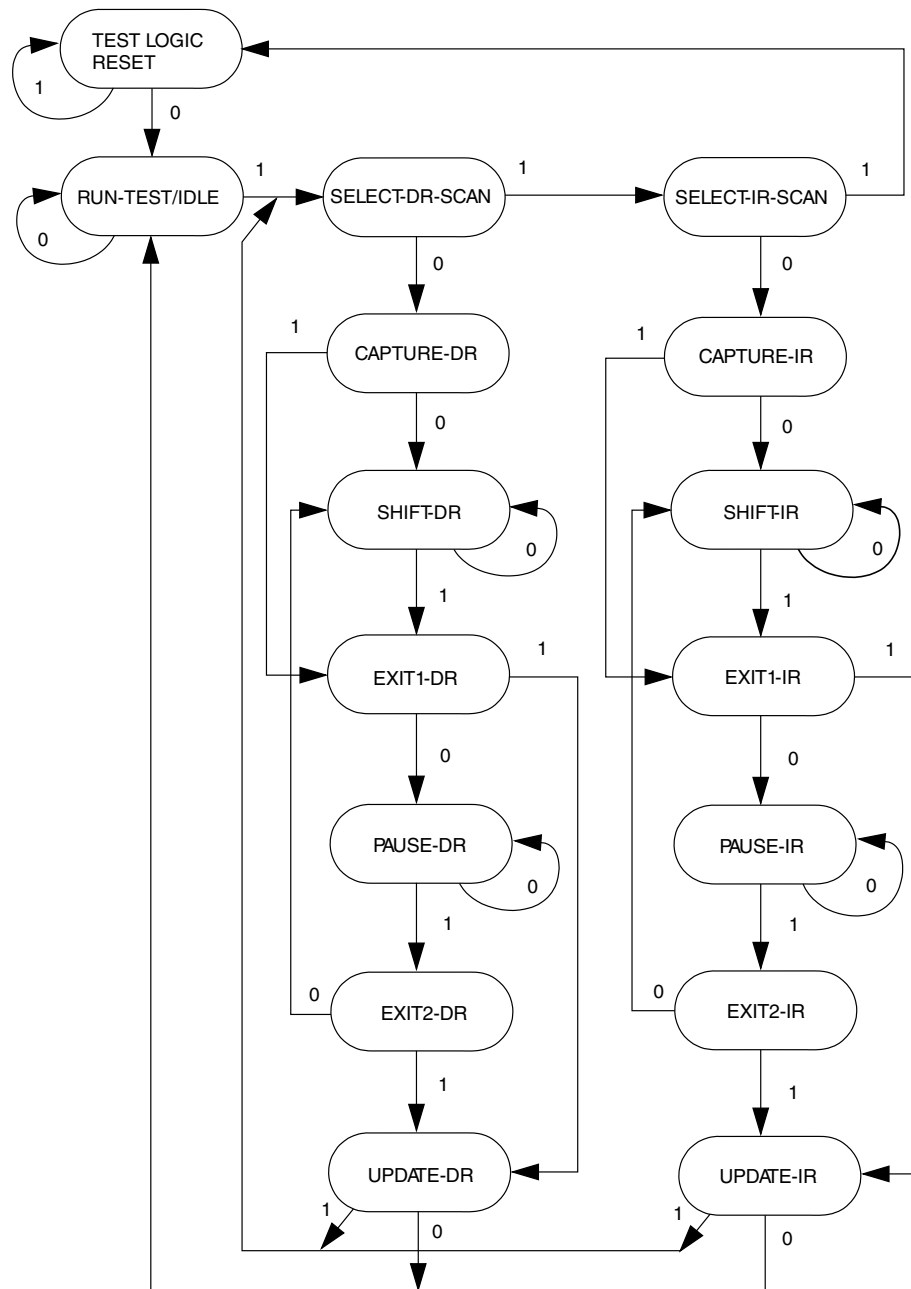
Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.



61.3.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin.

The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the figure shows, holding TMS at logic 1 when clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 321. IEEE 1149.1-2001 TAP controller finite state machine

61.3.3.1 Enabling the TAP controller

To enable JTAG operation TRSTn should off-state (1'b1) and RESETN = 1'b0.

61.3.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAG data registers is achieved by loading the instruction register with any of the JTAG block instructions when the JTAG is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated after the required number of bits have been acquired.

61.3.4 JTAG block instructions

The JTAG block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction. See the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 626. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
EXTEST	00000	Selects boundary scan register and applies preloaded values to output pins.
SAMPLE/PRELOAD	00001	Selects boundary scan register for shifting and sampling without disturbing functional operation
HIGHZ	00010	Selects bypass register and tristates all output pins NOTE Execution of this instruction asserts functional reset.
CLAMP	00011	Selects bypass register and applies preloaded values to output pins NOTE Execution of this instruction asserts functional reset.
IDCODE	00100	Selects device identification register for shift
Factory debug reserved	00101	Intended for factory debug only
Not used	00110 - 01000	Not used
Factory debug reserved	01001	Intended for factory debug only
Not used	01010	Not used
Factory debug reserved	01011	Intended for factory debug only
Not used	01100 - 11110	Not used
BYPASS	11111	Selects bypass register for data operations

61.3.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAG block is reset.

61.3.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction obtains a sample of the system data and control signals present at the MCU input pins, and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

61.3.4.3 EXTEST external test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins.

61.3.4.4 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. When HIGHZ is active all output drivers are placed in an inactive drive state (for example, high impedance).

61.3.4.5 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register when the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) when conducting an EXTEST type of instruction through the boundary scan register.

61.3.4.6 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions.

- The bit pattern of all ones in the Instruction Register must decode to the BYPASS instruction.
- The standard also states that all unused instruction codes not declared to be private must also decode to BYPASS.
- When the BYPASS is in effect, the Bypass Register is parallel loaded with a 0 upon passing through the CaptureDR state. This initializes the register with known, predictable data.

61.3.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

61.4 Initialization

The RESETN pin must be asserted and consequently System Logic will in reset. The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAG block and enable access to registers, the following sequence is required:

1. RESETN must be asserted (1'b0) before starting JTAG operations.
2. Place the JTAG in reset through TAP controller state machine transitions controlled by TMS.
3. Load the appropriate instruction for the test or action to be performed.

61.5 Register description

This section provides a detailed description of the JTAG block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

61.5.1 Instruction register

The JTAG block uses a 5-bit instruction register as shown in the following figure.

The instruction register allows instructions to be loaded into the block to select the test to be performed, or the test data register to be accessed, or both. Instructions are shifted in through TDI when the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states.

Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. According the Boundary Scan Handbook, in Capture-IR the Instruction Register is parallel loaded and the two least significant bits loaded must be "01". These bits will be used later for testing the integrity of the 1149.1 logic.

	4	3	2	1	0
R	0	0	0	0	1
W	Instruction Code				
Reset:	0	0	1	0	0

Figure 322. Instruction register

61.5.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the following instructions are active:

- BYPASS
- HIGHZ
- CLAMP

After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

61.5.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP.

The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state when the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

Table 627. Device identification register (JTAG ID)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number (PRN)				Design Center (DC)								Part Identification Number (PIN)			
W																

Table continues on the next page...

Table 627. Device identification register (JTAG ID) (continued)

Reset	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number (PIN)				Manufacturer Identity Code (MIC)											1
W																
Reset	1	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1

The following table describes the device identification register functions.

Table 628. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part.
DC	Design Center. Indicates the design center.
PIN	Part Identification Number. Contains the part number of the device.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID .
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

61.5.4 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE instructions are active. It is used to:

- Capture input pin data
- Force fixed values on output pins
- Select a logic value and direction for bidirectional pins

Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Boundary scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

Appendix A

Revision History

A.1 Reference manual changes

Table 629. Reference manual changes

Document ID	Release date	Description
IMXRT500RM v.2	12/2023	• Memory Maps — Updated section "Shared RAM Address Map" • System Controller (SYSCTL) — In section "Memory map and register definition"- Updated note to "Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out." — In PDSLEEPCFG0, PDSLEEPCFG1, PDRUNCFG0, PDRUNCFG1, PDRUNCFG0_SET, PDRUNCFG0_CLR, PDRUNCFG1_SET and PDRUNCFG1_CLR registers- Added HSPAD_*_REF_* bits. • Clock Control (CLKCTL) — In section "Memory map and register definition"- Updated note to "Flexcomm modules 0-8 and 10-12 are available on this chip. Please disregard references to Flexcomm 9 and 13 as they are not pinned out." — Updated Figure 14. RT500 clock diagram page 1 — Updated Figure 17. RT500 clock diagram page 4 • Pin Interrupt and Pattern Match (PINT) — Updated note in section "Pin Interrupts Initialization" • Non-Secure Boot ROM — Added a note in section "Overview" — Updated Table 95 in section "eMMC Configuration Block". — Added note "Flash Block erase is not supported for FlexSPI memory, when flash-erase-region command is issued from blhost." in sections "FlexSPI NOR Configuration Option Block", "1-bit SPI NOR Configuration Option Block", and FLEXSPI FLASH Driver APIs

Table continues on the next page...

Table 629. Reference manual changes

Document ID	Release date	Description
		— In section "ROM API layout"- Updated figure Figure 79. ROM API Tree structure. — Changed IAP API to KB API throughout the chapter. — In section "API to invoke into ROM" updated Table 141. Location of runboot ROM API in ROM API. — Moved section "IAP APIs" and its subsection as section 18.7.6 — Updated section Lockupthechip mode. • Secure Boot ROM — Added ISP commands table in section "Boot flow diagram". — Updated section Secure ROM API — In section "skboot_authenticate_api"- Added note "kb_init must be called beforehand; whereas, skboot_authneticate can be called directly instead of kb_execute." — In section "skboot_authenticate_api", "HASH_IRQHandler", and "skboot_otfad_encrypt"- Added a table — Split the section "skboot_authenticate_api" into three sections "RAM usage and Hashcrypt", "Memory alignment", and "Return values/error codes". • Security — Added section "RT500 Life cycle state" • PowerQuad DSP Coprocessor and Accelerator (PowerQuad) — Extensive changes throughout chapter, although no changes to register or field names. • Windowed Watchdog Timer (WWDT) — Added Module instance topic • MIPI DSI (MIPI_DSI) Interface — Changed mnemonic to MIPI_DSI_HOST throughout. — Updated DPI-2 Interface Ports (-minor change from "g" to "active"). — Updated DPI-2 Video Timing Recreation over DSI (reformatted content).

Table continues on the next page...

Table 629. Reference manual changes

Document ID	Release date	Description
		— Throughout register section, added introduction/ function and reformatted bitfield descriptions. • DMIC Subsystem — In section "Clocking and Outputs"- Added text "Data from DMIC channels 0 and 1 can optionally be sent directly to the Flexcomm0 I2S function for output." • Improved Inter-Integrated Circuit (I3C) — Changed field location of MDATACTRL[UNLOCK] — Changed field width of SCAPABILITIES[HDRSUPP] • Flexcomm Serial Communication — In USART- Added bit field values in OSR[OSRVAL]

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Cadence — the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All rights reserved worldwide.

EdgeLock — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

NXP SECURE CONNECTIONS FOR A SMARTER WORLD — is a trademark of NXP B.V.



Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2023.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12/2023

Document identifier: IMXRT500RM