CC13x4x10, CC26x4x10 SimpleLink™ Wireless MCU

Technical Reference Manual

Literature Number: SWCU194
MARCH 2023

Table of Contents

Read This First 37
About This Manual 37
Devices 37
Register, Field, and Bit Calls 37
Related Documentation 38
1 Architectural Overview 41
1.1 Target Applications 42
1.2 Overview 42
1.3 Functional Overview 45
2 Arm® Cortex®-M33 Processor with FPU 57
2.1 Arm® Cortex®-M33 Processor Introduction 58
2.2 Block Diagram 58
2.3 Overview 59
2.4 Programming Model 62
2.5 Arm® Cortex®-M33 Registers 69
3 Memory Map 267
3.1 Introduction 267
3.2 Memory Map (Secure and Non-secure) 267
3.3 Memory Map 272
4 Arm® Cortex®-M33 Peripherals 275
4.1 Arm® Cortex®-M33 Peripherals Introduction 276
5 Interrupts and Events 279
5.1 Exception Model 280
5.2 Fault Handling 288
5.3 Security State Switches 290
5.4 Event Fabric 291
5.5 AON Event Fabric 295
5.6 MCU Event Fabric 295
5.7 AON Events 300
5.8 Interrupts and Events Registers 301
6 JTAG Interface 461
6.1 Overview 462
6.2 cJTAG 464
6.3 ICEPick 465
6.4 ICEMelter 475
6.5 Serial Wire Viewer (SWV) 476
6.6 Halt In Boot (HIB) 476
6.7 Debug and Shutdown 476
6.8 Boundary Scan 476
7 Power, Reset, and Clock Management (PRCM) 479
7.1 Introduction 480
7.2 System CPU Mode 481
7.3 Supply System 481
7.4 Digital Power Partitioning 482
7.5 Clock Management 484
7.6 Power Modes 490
7.7 Reset 494
7.8 PRCM Registers 495
8 Versatile Instruction Memory System (VIMS) 615
8.1 Introduction 616

8.2 VIMS Configurations ... 617
8.3 VIMS Software Remarks ... 619
8.4 FLASH ... 621
8.5 ROM Functions ... 623
8.6 VIMS Registers ... 623
9 SRAM ... 717
9.1 Introduction ... 718
9.2 Main Features ... 718
9.3 Data Retention ... 718
9.4 Parity and SRAM Error Support ... 718
9.5 SRAM Auto-Initialization ... 718
9.6 Parity Debug Behavior ... 718
9.7 SRAM Registers ... 719
10 Bootloader ... 735
10.1 Bootloader Functionality ... 736
10.2 Bootloader Interfaces ... 736
11 Device Configuration ... 753
11.1 Customer Configuration (CCFG) ... 754
11.2 CCFG Registers ... 756
11.3 Factory Configuration (FCFG) ... 788
11.4 FCFG1 Registers ... 789
12 AES and Hash Cryptoprocessor ... 859
12.1 Introduction ... 860
12.2 Functional Description ... 861
12.3 DMA Controller ... 872
12.4 AES and Hash Cryptoprocessor Performance ... 876
12.5 Programming Guidelines ... 877
12.6 Conventions and Compliances ... 907
12.7 CRYPTO Registers ... 910
13 PKA Engine ... 1013
13.1 Introduction ... 1014
13.2 Functional Description ... 1015
13.3 PKA Engine Performance ... 1026
13.4 PKA Registers ... 1029
14 True Random Number Generator (TRNG) ... 1047
14.1 Introduction ... 1048
14.2 Block Diagram ... 1048
14.3 TRNG Software Reset ... 1049
14.4 Interrupt Requests ... 1049
14.5 TRNG Operation Description ... 1050
14.6 TRNG Low-Level Programming Guide ... 1052
14.7 TRNG Registers ... 1055
15 I/O Controller (IOC) ... 1079
15.1 Introduction ... 1080
15.2 IOC Overview ... 1080
15.3 I/O Mapping and Configuration ... 1081
15.4 Edge Detection on DIO Pins ... 1082
15.5 Unused I/O Pins ... 1083
15.6 GPIO ... 1083
15.7 I/O Pin Capability ... 1084
15.8 Peripheral PORT_IDs ... 1086
15.9 I/O Pins ... 1088
15.10 IOC Registers ... 1090
16 Micro Direct Memory Access (µDMA) ... 1319
16.1 Introduction ... 1320
16.2 Block Diagram ... 1321
16.3 Functional Description ... 1321
16.4 Initialization and Configuration ... 1348
16.5 UDMA Registers ... 1350
17 Timers ... 1371
17.1 Introduction ... 1372

17.2 Block Diagram....1373
17.3 Functional Description....1373
17.4 Initialization and Configuration....1383
17.5 GPT Registers....1387
18 Real-Time Clock (RTC)....1421
18.1 Introduction....1422
18.2 Functional Specifications....1422
18.3 RTC Register Information....1424
18.4 RTC Registers....1426
19 Watchdog Timer (WDT)....1441
19.1 Introduction....1442
19.2 Functional Description....1442
19.3 Initialization and Configuration....1443
19.4 WDT Registers....1444
20 AUX Domain Sensor Controller and Peripherals....1455
20.1 Introduction....1456
20.2 Power and Clock Management....1458
20.3 Sensor Controller....1461
20.4 Digital Peripheral Modules....1479
20.5 Analog Peripheral Modules....1505
20.6 Event Routing and Usage....1521
20.7 Sensor Controller Alias Register Space....1527
20.8 AUX Domain Sensor Controller and Peripherals Registers....1532
21 Battery Monitor and Temperature Sensor (BATMON)....1821
21.1 Introduction....1822
21.2 Functional Description....1822
21.3 AON_BATMON Registers....1823
22 Universal Asynchronous Receiver/Transmitter (UART)....1843
22.1 Introduction....1844
22.2 Block Diagram....1845
22.3 Signal Description....1845
22.4 Functional Description....1845
22.5 Interface to µDMA....1850
22.6 Initialization and Configuration....1851
22.7 UART Registers....1852
23 Serial Peripheral Interface (SPI)....1869
23.1 Introduction....1870
23.2 Block Diagram....1871
23.3 Signal Description....1872
23.4 Functional Description....1872
23.5 µDMA Operation....1882
23.6 Initialization and Configuration....1882
23.7 SPI Registers....1884
24 Inter-Integrated Circuit (I^2C)....1905
24.1 Introduction....1906
24.2 Block Diagram....1906
24.3 Functional Description....1907
24.4 Initialization and Configuration....1918
24.5 I2C Registers....1919
25 Inter-IC Sound (I^2S)....1939
25.1 Introduction....1940
25.2 Block Diagram....1941
25.3 Signal Description....1942
25.4 Functional Description....1942
25.5 Memory Interface....1948
25.6 Samplestamp Generator....1951
25.7 Error Detection....1954
25.8 Usage....1954
25.9 I2S Registers....1955
26 Radio....1985
26.1 RF Core....1986

26.2 Radio Doorbell 1988
26.3 RF Core HAL 1992
26.4 Data Queue Usage 2029
26.5 IEEE 802.15.4 2033
26.6 Bluetooth® Low Energy 2056
26.7 Data Handling 2077
26.8 Radio Operation Command Descriptions 2078
26.9 Immediate Commands 2117
26.10 Proprietary Radio 2118
26.11 Radio Registers 2140
27 Revision History 2168

List of Figures

Figure 1-1. CC13x4x10 and CC26x4x10 Block Diagram 43
Figure 1-2. CC13x4x10 and CC26x4x10 Supply System 54
Figure 2-1. Block Diagram 59
Figure 2-2. Simple Arm Debug Interface MEM-AP Implementation 60
Figure 2-3. TPIU Block Diagram 60
Figure 2-4. Bus Topology 67
Figure 3-1. Accessible Memory per Bus Master for Different Runtime Security Configurations 269
Figure 4-1. Arm® Cortex®-M33 Processor Memory Map 276
Figure 5-1. Event Fabric Concept 292
Figure 5-2. Event Fabric Overview (Simplified) 293
Figure 6-1. JTAG Subsystem 463
Figure 6-2. cJTAG Conceptual Diagram 464
Figure 6-3. Data Shift Register 468
Figure 6-4. Instruction Register 468
Figure 6-5. Bypass Register 468
Figure 6-6. Device Identification Register 468
Figure 6-7. User Code Register 469
Figure 6-8. ICEPick Identification Register 469
Figure 6-9. Connect Register 470
Figure 6-10. ROUTER DR Scan Chain 470
Figure 6-11. Boundary Scan Cell 477
Figure 7-1. Hierarchy of Power Saving Features 480
Figure 7-2. CC13x4x10 and CC26x4x10 Supply System 482
Figure 7-3. Digital Power Partitioning in CC13x4x10 and CC26x4x10 483
Figure 7-4. Clock Sources 484
Figure 7-5. System Clock Muxing 487
Figure 7-6. Clocks in MCU_VD 489
Figure 8-1. VIMS Overview 616
Figure 8-2. VIMS Mode Switching Flowchart 617
Figure 8-3. VIMS Module in GPRAM Mode 617
Figure 8-4. VIMS Module in Off Mode 618
Figure 8-5. VIMS Module in Cache Mode 618
Figure 8-6. Software Precautions with No RAM Retention 620
Figure 8-7. GPRAM Retention 621
Figure 10-1. Sequence Diagram for Send and Receive Protocol 737
Figure 10-2. Serial Bus Packet Format 738
Figure 12-1. DMA Controller and Integration 873
Figure 12-2. Symmetric Crypto Processing Steps 876
Figure 12-3. HMAC Steps 883
Figure 13-1. PKA Engine 1015
Figure 13-2. Operation Sequence 1025
Figure 13-3. Interleaved Operation Sequence 1025
Figure 13-4. Basic PKCP Operation Sequence 1026
Figure 14-1. Random Number Generator Block Diagram 1048
Figure 14-2. TRNG Polling Mode 1053
Figure 14-3. Interrupt Service Routine 1054
Figure 15-1. IOC Overview (Simplified) 1080
Figure 15-2. Generic I/O Pin (Simplified) 1088

Figure 16-1. µDMA Block Diagram....1321
Figure 16-2. Example of Ping-Pong µDMA Transaction....1328
Figure 16-3. Memory Scatter-Gather, Setup, and Configuration....1330
Figure 16-4. Memory Scatter-Gather, µDMA Copy Sequence....1331
Figure 16-5. Peripheral Scatter-Gather, Setup, and Configuration....1332
Figure 16-6. Peripheral Scatter-Gather, µDMA Copy Sequence....1333
Figure 17-1. GPTM Module Block Diagram....1373
Figure 17-2. Input Edge-Count Mode Example, Counting Down....1377
Figure 17-3. Input Edge-Time Mode Example....1378
Figure 17-4. 16-Bit PWM Mode Example....1380
Figure 17-5. CCP Output, GPT:TnMATCHR > GPT:TnILR....1380
Figure 17-6. CCP Output, GPT:TnMATCHR = GPT:TnILR....1381
Figure 17-7. CCP Output, GPT:TnILR > GPT:TnMATCHR....1381
Figure 17-8. Timer Daisy-Chain....1382
Figure 18-1. AON_RTC Channels....1423
Figure 19-1. WDT Block Diagram....1443
Figure 20-1. AUX Domain Block Diagram....1457
Figure 20-2. AUX State Diagram....1458
Figure 20-3. ADC Window Monitor - Execution Code....1462
Figure 20-4. ADC Window Monitor....1463
Figure 20-5. Task Testing....1464
Figure 20-6. Help Viewer....1465
Figure 20-7. MAC Block Diagram....1473
Figure 20-8. MAC Timing Diagram....1474
Figure 20-9. SCE Wake-Up and Event Interface....1475
Figure 20-10. Avoid VDDR Recharge Power Noise....1478
Figure 20-11. AUX I/O Block Diagram....1481
Figure 20-12. AUX_SPIM Block Diagram....1483
Figure 20-13. SPI Timing Diagram: PHA = 0....1485
Figure 20-14. SPI Timing Diagram: PHA = 1....1485
Figure 20-15. AUX_TDC Block Diagram....1487
Figure 20-16. Phase Width Timing Requirements....1491
Figure 20-17. Frequency Measurement Waveform....1491
Figure 20-18. Arbitrary Time Measurement 1....1492
Figure 20-19. Arbitrary Time Measurement 2....1493
Figure 20-20. Arbitrary Time Measurement 3....1494
Figure 20-21. Arbitrary Time Measurement 4....1495
Figure 20-22. Pulse Counting....1496
Figure 20-23. AUX_TIMER01 Block Diagram....1497
Figure 20-24. AUX_TIMER2 Block Diagram....1498
Figure 20-25. Period Pulse Width Measurement....1502
Figure 20-26. Center-Aligned PWM....1503
Figure 20-27. Edge-Aligned PWM....1504
Figure 20-28. AUX Analog Block Diagram....1506
Figure 20-29. ADC Block Diagram....1507
Figure 20-30. COMPA Block Diagram....1511
Figure 20-31. COMPB Block Diagram....1513
Figure 20-32. Reference DAC Block Diagram....1515
Figure 20-33. DAC Sample Clock Phases....1516
Figure 20-34. Sample Clock Period Configuration....1516
Figure 20-35. ISRC Block Diagram....1520
Figure 22-1. UART Module Block Diagram....1845
Figure 22-2. UART Character Frame....1846
Figure 22-3. µDMA Example....1850
Figure 23-1. SPI Module Block Diagram (Master Mode)....1871
Figure 23-2. TI Synchronous Serial Frame Format (Single Transfer)....1876
Figure 23-3. TI Synchronous Serial Frame Format (Continuous Transfer)....1876
Figure 23-4. Motorola™ SPI Format (Single Transfer) with SPO = 0 and SPH = 0....1877
Figure 23-5. Motorola SPI Format (Continuous Transfer) with SPO = 0 and SPH = 0....1877
Figure 23-6. Motorola SPI Frame Format with SPO = 0 and SPH = 1....1878
Figure 23-7. Motorola SPI Frame Format (Single Transfer) with SPO = 1 and SPH = 0....1879

Figure 23-8. Motorola SPI Frame Format (Continuous Transfer) with SPO = 1 and SPH = 0 1879
Figure 23-9. Motorola SPI Frame Format with SPO = 1 and SPH = 1 1880
Figure 23-10. MICROWIRE Frame Format (Single Frame) 1880
Figure 23-11. MICROWIRE Frame Format (Continuous Transfer) 1881
Figure 24-1. I2C Block Diagram 1906
Figure 24-2. I2C Bus Configuration 1907
Figure 24-3. Start and Stop Conditions 1907
Figure 24-4. Complete Data Transfer with a 7-Bit Address 1908
Figure 24-5. R/S Bit in First Byte 1908
Figure 24-6. Data Validity During Bit Transfer on the I2C Bus 1908
Figure 24-7. Master Single Transmit 1911
Figure 24-8. Master Single Receive 1912
Figure 24-9. Master Transmit with Repeated Start Condition 1913
Figure 24-10. Master Receive with Repeated Start Condition 1914
Figure 24-11. Master Receive with Repeated Start after Transmit with Repeated Start Condition 1915
Figure 24-12. Master Transmit with Repeated Start after Receive with Repeated Start Condition 1916
Figure 24-13. Slave Command Sequence 1917
Figure 25-1. Simplified I2S Module Block Diagram 1941
Figure 25-2. I2S Serial Format 1943
Figure 25-3. LJF Serial Format 1944
Figure 25-4. RJF Serial Format 1945
Figure 25-5. DSP Serial Format (Zero Data Delay) 1946
Figure 25-6. 16-Bit Stereo I2S, LJF, and RJF Formats on One ADx Pin, Showing Three Frames in Memory 1949
Figure 25-7. 24-Bit Stereo I2S, LJF, and RJF Formats on One ADx Pin, Showing Two Frames in Memory 1949
Figure 25-8. 16-Bit Mono I2S, LJF, and RJF Formats on One ADx Pin, Showing Six Frames in Memory 1949
Figure 25-9. 16-Bit I2S Format on AD0 and AD1 Pins, Showing Two Frames in Memory 1950
Figure 25-10. 16-Bit DSP Format on AD0 and AD1 Pins, Showing Two Frames in Memory 1950
Figure 25-11. Samplestamp Generator 1952
Figure 26-1. Limited RF Core Overview with External Dependencies 1987
Figure 26-2. Hardware Support for the HAL 1988
Figure 26-3. CMDR Register for Radio Operation Commands and Immediate Commands 1992
Figure 26-4. CMDR Register for Direct Commands 1992
Figure 26-5. Format of RFC_DBELL:CMDSTA Register 1993
Figure 26-6. RX Queue Entry Element (Stapled Fields are Optional) 2042
Figure 26-7. CSMA-CA Operation 2051
Figure 26-8. Receive Buffer Entry Element 2077
Figure 26-9. Standard Packet Format 2118
Figure 26-10. Advanced Packet Format 2118
Figure 26-11. Receive Buffer Entry Element 2126

List of Tables

Table 2-1. Processor Core Register Set Summary 64
Table 2-2. CPU_ITM Registers 69
Table 2-3. CPU_ITM Access Type Codes 71
Table 2-4. STIM0 Register Field Descriptions 72
Table 2-5. STIM1 Register Field Descriptions 73
Table 2-6. STIM2 Register Field Descriptions 74
Table 2-7. STIM3 Register Field Descriptions 75
Table 2-8. STIM4 Register Field Descriptions 76
Table 2-9. STIM5 Register Field Descriptions 77
Table 2-10. STIM6 Register Field Descriptions 78
Table 2-11. STIM7 Register Field Descriptions 79
Table 2-12. STIM8 Register Field Descriptions 80
Table 2-13. STIM9 Register Field Descriptions 81
Table 2-14. STIM10 Register Field Descriptions 82
Table 2-15. STIM11 Register Field Descriptions 83
Table 2-16. STIM12 Register Field Descriptions 84
Table 2-17. STIM13 Register Field Descriptions 85
Table 2-18. STIM14 Register Field Descriptions 86
Table 2-19. STIM15 Register Field Descriptions 87
Table 2-20. STIM16 Register Field Descriptions 88

Table 2-21. STIM17 Register Field Descriptions ... 89
Table 2-22. STIM18 Register Field Descriptions ... 90
Table 2-23. STIM19 Register Field Descriptions ... 91
Table 2-24. STIM20 Register Field Descriptions ... 92
Table 2-25. STIM21 Register Field Descriptions ... 93
Table 2-26. STIM22 Register Field Descriptions ... 94
Table 2-27. STIM23 Register Field Descriptions ... 95
Table 2-28. STIM24 Register Field Descriptions ... 96
Table 2-29. STIM25 Register Field Descriptions ... 97
Table 2-30. STIM26 Register Field Descriptions ... 98
Table 2-31. STIM27 Register Field Descriptions ... 99
Table 2-32. STIM28 Register Field Descriptions ... 100
Table 2-33. STIM29 Register Field Descriptions ... 101
Table 2-34. STIM30 Register Field Descriptions ... 102
Table 2-35. STIM31 Register Field Descriptions ... 103
Table 2-36. TER0 Register Field Descriptions ... 104
Table 2-37. TPR Register Field Descriptions ... 105
Table 2-38. TCR Register Field Descriptions ... 106
Table 2-39. INT_ATREADY Register Field Descriptions ... 107
Table 2-40. INT_ATVALID Register Field Descriptions ... 108
Table 2-41. ITCTRL Register Field Descriptions ... 109
Table 2-42. DEVARCH Register Field Descriptions ... 110
Table 2-43. DEVTYPE Register Field Descriptions ... 111
Table 2-44. PIDR4 Register Field Descriptions ... 112
Table 2-45. PIDR5 Register Field Descriptions ... 113
Table 2-46. PIDR6 Register Field Descriptions ... 114
Table 2-47. PIDR7 Register Field Descriptions ... 115
Table 2-48. PIDR0 Register Field Descriptions ... 116
Table 2-49. PIDR1 Register Field Descriptions ... 117
Table 2-50. PIDR2 Register Field Descriptions ... 118
Table 2-51. PIDR3 Register Field Descriptions ... 119
Table 2-52. CIDR0 Register Field Descriptions ... 120
Table 2-53. CIDR1 Register Field Descriptions ... 121
Table 2-54. CIDR2 Register Field Descriptions ... 122
Table 2-55. CIDR3 Register Field Descriptions ... 123
Table 2-56. CPU_DWT Registers ... 124
Table 2-57. CPU_DWT Access Type Codes ... 124
Table 2-58. CTRL Register Field Descriptions ... 125
Table 2-59. CYCCNT Register Field Descriptions ... 126
Table 2-60. CPICNT Register Field Descriptions ... 127
Table 2-61. EXCCNT Register Field Descriptions ... 128
Table 2-62. SLEEPCNT Register Field Descriptions ... 129
Table 2-63. LSUCNT Register Field Descriptions ... 130
Table 2-64. FOLDCNT Register Field Descriptions ... 131
Table 2-65. PCSR Register Field Descriptions ... 132
Table 2-66. FUNCTION0 Register Field Descriptions ... 133
Table 2-67. FUNCTION1 Register Field Descriptions ... 134
Table 2-68. FUNCTION2 Register Field Descriptions ... 135
Table 2-69. FUNCTION3 Register Field Descriptions ... 136
Table 2-70. DEVARCH Register Field Descriptions ... 137
Table 2-71. DEVTYPE Register Field Descriptions ... 138
Table 2-72. PIDR4 Register Field Descriptions ... 139
Table 2-73. PIDR5 Register Field Descriptions ... 140
Table 2-74. PIDR6 Register Field Descriptions ... 141
Table 2-75. PIDR7 Register Field Descriptions ... 142
Table 2-76. PIDR0 Register Field Descriptions ... 143
Table 2-77. PIDR1 Register Field Descriptions ... 144
Table 2-78. PIDR2 Register Field Descriptions ... 145
Table 2-79. PIDR3 Register Field Descriptions ... 146
Table 2-80. CIDR0 Register Field Descriptions ... 147
Table 2-81. CIDR1 Register Field Descriptions ... 148

Table 2-82. CIDR2 Register Field Descriptions.... 149
Table 2-83. CIDR3 Register Field Descriptions.... 150
Table 2-84. CPU_SYSTICK Registers.... 151
Table 2-85. CPU_SYSTICK Access Type Codes.... 151
Table 2-86. CSR Register Field Descriptions.... 152
Table 2-87. RVR Register Field Descriptions.... 153
Table 2-88. CVR Register Field Descriptions.... 154
Table 2-89. CALIB Register Field Descriptions.... 155
Table 2-90. CPU_NVIC Registers.... 156
Table 2-91. CPU_NVIC Access Type Codes.... 157
Table 2-92. ISER0 Register Field Descriptions.... 158
Table 2-93. ISER1 Register Field Descriptions.... 159
Table 2-94. ICER0 Register Field Descriptions.... 160
Table 2-95. ICER1 Register Field Descriptions.... 161
Table 2-96. ISPR0 Register Field Descriptions.... 162
Table 2-97. ISPR1 Register Field Descriptions.... 163
Table 2-98. ICPR0 Register Field Descriptions.... 164
Table 2-99. ICPR1 Register Field Descriptions.... 165
Table 2-100. IABR0 Register Field Descriptions.... 166
Table 2-101. IABR1 Register Field Descriptions.... 167
Table 2-102. ITNS0 Register Field Descriptions.... 168
Table 2-103. ITNS1 Register Field Descriptions.... 169
Table 2-104. IPR0 Register Field Descriptions.... 170
Table 2-105. IPR1 Register Field Descriptions.... 171
Table 2-106. IPR2 Register Field Descriptions.... 172
Table 2-107. IPR3 Register Field Descriptions.... 173
Table 2-108. IPR4 Register Field Descriptions.... 174
Table 2-109. IPR5 Register Field Descriptions.... 175
Table 2-110. IPR6 Register Field Descriptions.... 176
Table 2-111. IPR7 Register Field Descriptions.... 177
Table 2-112. IPR8 Register Field Descriptions.... 178
Table 2-113. IPR9 Register Field Descriptions.... 179
Table 2-114. IPR10 Register Field Descriptions.... 180
Table 2-115. IPR11 Register Field Descriptions.... 181
Table 2-116. CPU_SCS Registers.... 182
Table 2-117. CPU_SCS Access Type Codes.... 183
Table 2-118. CPUID Register Field Descriptions.... 184
Table 2-119. ICSR Register Field Descriptions.... 185
Table 2-120. VTOR Register Field Descriptions.... 186
Table 2-121. AIRCR Register Field Descriptions.... 187
Table 2-122. SCR Register Field Descriptions.... 188
Table 2-123. CCR Register Field Descriptions.... 189
Table 2-124. SHPR1 Register Field Descriptions.... 190
Table 2-125. SHPR2 Register Field Descriptions.... 191
Table 2-126. SHPR3 Register Field Descriptions.... 192
Table 2-127. SHCSR Register Field Descriptions.... 193
Table 2-128. CFSR Register Field Descriptions.... 195
Table 2-129. HFSR Register Field Descriptions.... 197
Table 2-130. DFSR Register Field Descriptions.... 198
Table 2-131. MMFAR Register Field Descriptions.... 199
Table 2-132. BFAR Register Field Descriptions.... 200
Table 2-133. AFSR Register Field Descriptions.... 201
Table 2-134. ID_PFR0 Register Field Descriptions.... 202
Table 2-135. ID_PFR1 Register Field Descriptions.... 203
Table 2-136. ID_DFR0 Register Field Descriptions.... 204
Table 2-137. ID_AFR0 Register Field Descriptions.... 205
Table 2-138. ID_MMFR0 Register Field Descriptions.... 206
Table 2-139. ID_MMFR1 Register Field Descriptions.... 207
Table 2-140. ID_MMFR2 Register Field Descriptions.... 208
Table 2-141. ID_MMFR3 Register Field Descriptions.... 209
Table 2-142. ID_ISAR0 Register Field Descriptions.... 210

Table 2-143. ID_ISAR1 Register Field Descriptions....211
Table 2-144. ID_ISAR2 Register Field Descriptions....212
Table 2-145. ID_ISAR3 Register Field Descriptions....213
Table 2-146. ID_ISAR4 Register Field Descriptions....214
Table 2-147. CPU_MPU Registers....215
Table 2-148. CPU_MPU Access Type Codes....215
Table 2-149. TYPE Register Field Descriptions....216
Table 2-150. CTRL Register Field Descriptions....217
Table 2-151. RNR Register Field Descriptions....218
Table 2-152. RBAR Register Field Descriptions....219
Table 2-153. RLAR Register Field Descriptions....220
Table 2-154. RBAR_A1 Register Field Descriptions....221
Table 2-155. RLAR_A1 Register Field Descriptions....222
Table 2-156. RBAR_A2 Register Field Descriptions....223
Table 2-157. RLAR_A2 Register Field Descriptions....224
Table 2-158. RBAR_A3 Register Field Descriptions....225
Table 2-159. RLAR_A3 Register Field Descriptions....226
Table 2-160. MAIR0 Register Field Descriptions....227
Table 2-161. MAIR1 Register Field Descriptions....228
Table 2-162. CPU_SAU Registers....229
Table 2-163. CPU_SAU Access Type Codes....229
Table 2-164. CTRL Register Field Descriptions....230
Table 2-165. TYPE Register Field Descriptions....231
Table 2-166. RNR Register Field Descriptions....232
Table 2-167. RBAR Register Field Descriptions....233
Table 2-168. RLAR Register Field Descriptions....234
Table 2-169. SFSR Register Field Descriptions....235
Table 2-170. SFAR Register Field Descriptions....236
Table 2-171. CPU_DCB Registers....237
Table 2-172. CPU_DCB Access Type Codes....237
Table 2-173. DHCSR Register Field Descriptions....238
Table 2-174. DCRSR Register Field Descriptions....239
Table 2-175. DCRDR Register Field Descriptions....240
Table 2-176. DEMCR Register Field Descriptions....241
Table 2-177. DAUTHCTRL Register Field Descriptions....242
Table 2-178. DSCSR Register Field Descriptions....243
Table 2-179. CPU_SIG Registers....244
Table 2-180. CPU_SIG Access Type Codes....244
Table 2-181. STIR Register Field Descriptions....245
Table 2-182. CPU_FPU Registers....246
Table 2-183. CPU_FPU Access Type Codes....246
Table 2-184. FPCCR Register Field Descriptions....247
Table 2-185. FPCAR Register Field Descriptions....248
Table 2-186. FPDSCR Register Field Descriptions....249
Table 2-187. MVFR0 Register Field Descriptions....250
Table 2-188. MVFR1 Register Field Descriptions....251
Table 2-189. MVFR2 Register Field Descriptions....252
Table 2-190. CPU_TPIU Registers....253
Table 2-191. CPU_TPIU Access Type Codes....253
Table 2-192. SSPSR Register Field Descriptions....254
Table 2-193. CSPSR Register Field Descriptions....255
Table 2-194. ACPR Register Field Descriptions....256
Table 2-195. SPPR Register Field Descriptions....257
Table 2-196. FFSR Register Field Descriptions....258
Table 2-197. FFCR Register Field Descriptions....259
Table 2-198. PSCR Register Field Descriptions....260
Table 2-199. CLAIMMASK Register Field Descriptions....261
Table 2-200. CLAIMSET Register Field Descriptions....262
Table 2-201. CLAIMTAG Register Field Descriptions....263
Table 2-202. CLAIMCLR Register Field Descriptions....264
Table 2-203. DEVID Register Field Descriptions....265

Table 2-204. DEVTYPE Register Field Descriptions.... 266
Table 3-1. Memory Map (Secure and Non-secure).... 267
Table 3-2. Granularity.... 268
Table 3-3. Determining the Security Attribution for a Given Address.... 270
Table 3-4. BUS_CFG Configuration.... 270
Table 3-5. System Management Registers.... 271
Table 3-6. Apertures with all Registers Mapped to Secure Address Space.... 271
Table 3-7. Memory Map.... 272
Table 4-1. Core Peripheral Register Regions.... 276
Table 5-1. Exception Types.... 282
Table 5-2. Vector Table With Security Extension.... 284
Table 5-3. Extended Priority.... 285
Table 5-4. Exception Return Behavior.... 287
Table 5-5. Faults.... 288
Table 5-6. Security State Transitions.... 290
Table 5-7. MCU Events.... 295
Table 5-8. Freeze Subscriber Event Selection.... 299
Table 5-9. AON Events.... 300
Table 5-10. AON_EVENT Registers.... 301
Table 5-11. AON_EVENT Access Type Codes.... 301
Table 5-12. MCUWUSEL Register Field Descriptions.... 302
Table 5-13. MCUWUSEL1 Register Field Descriptions.... 306
Table 5-14. EVTOMCUSEL Register Field Descriptions.... 310
Table 5-15. RTCSEL Register Field Descriptions.... 313
Table 5-16. EVENT Registers.... 314
Table 5-17. EVENT Access Type Codes.... 317
Table 5-18. CPUIRQSEL0 Register Field Descriptions.... 318
Table 5-19. CPUIRQSEL1 Register Field Descriptions.... 319
Table 5-20. CPUIRQSEL2 Register Field Descriptions.... 320
Table 5-21. CPUIRQSEL3 Register Field Descriptions.... 321
Table 5-22. CPUIRQSEL4 Register Field Descriptions.... 322
Table 5-23. CPUIRQSEL5 Register Field Descriptions.... 323
Table 5-24. CPUIRQSEL6 Register Field Descriptions.... 324
Table 5-25. CPUIRQSEL7 Register Field Descriptions.... 325
Table 5-26. CPUIRQSEL8 Register Field Descriptions.... 326
Table 5-27. CPUIRQSEL9 Register Field Descriptions.... 327
Table 5-28. CPUIRQSEL10 Register Field Descriptions.... 328
Table 5-29. CPUIRQSEL11 Register Field Descriptions.... 329
Table 5-30. CPUIRQSEL12 Register Field Descriptions.... 330
Table 5-31. CPUIRQSEL13 Register Field Descriptions.... 331
Table 5-32. CPUIRQSEL14 Register Field Descriptions.... 332
Table 5-33. CPUIRQSEL15 Register Field Descriptions.... 333
Table 5-34. CPUIRQSEL16 Register Field Descriptions.... 334
Table 5-35. CPUIRQSEL17 Register Field Descriptions.... 335
Table 5-36. CPUIRQSEL18 Register Field Descriptions.... 336
Table 5-37. CPUIRQSEL19 Register Field Descriptions.... 337
Table 5-38. CPUIRQSEL20 Register Field Descriptions.... 338
Table 5-39. CPUIRQSEL21 Register Field Descriptions.... 339
Table 5-40. CPUIRQSEL22 Register Field Descriptions.... 340
Table 5-41. CPUIRQSEL23 Register Field Descriptions.... 341
Table 5-42. CPUIRQSEL24 Register Field Descriptions.... 342
Table 5-43. CPUIRQSEL25 Register Field Descriptions.... 343
Table 5-44. CPUIRQSEL26 Register Field Descriptions.... 344
Table 5-45. CPUIRQSEL27 Register Field Descriptions.... 345
Table 5-46. CPUIRQSEL28 Register Field Descriptions.... 346
Table 5-47. CPUIRQSEL29 Register Field Descriptions.... 347
Table 5-48. CPUIRQSEL30 Register Field Descriptions.... 348
Table 5-49. CPUIRQSEL31 Register Field Descriptions.... 349
Table 5-50. CPUIRQSEL32 Register Field Descriptions.... 350
Table 5-51. CPUIRQSEL33 Register Field Descriptions.... 351
Table 5-52. CPUIRQSEL34 Register Field Descriptions.... 352

Table 5-53. CPUIRQSEL35 Register Field Descriptions... 353
Table 5-54. CPUIRQSEL36 Register Field Descriptions... 354
Table 5-55. CPUIRQSEL37 Register Field Descriptions... 355
Table 5-56. CPUIRQSEL38 Register Field Descriptions... 356
Table 5-57. CPUIRQSEL39 Register Field Descriptions... 357
Table 5-58. CPUIRQSEL40 Register Field Descriptions... 358
Table 5-59. CPUIRQSEL41 Register Field Descriptions... 359
Table 5-60. CPUIRQSEL42 Register Field Descriptions... 360
Table 5-61. RFCSEL0 Register Field Descriptions... 361
Table 5-62. RFCSEL1 Register Field Descriptions... 362
Table 5-63. RFCSEL2 Register Field Descriptions... 363
Table 5-64. RFCSEL3 Register Field Descriptions... 364
Table 5-65. RFCSEL4 Register Field Descriptions... 365
Table 5-66. RFCSEL5 Register Field Descriptions... 366
Table 5-67. RFCSEL6 Register Field Descriptions... 367
Table 5-68. RFCSEL7 Register Field Descriptions... 368
Table 5-69. RFCSEL8 Register Field Descriptions... 369
Table 5-70. RFCSEL9 Register Field Descriptions... 370
Table 5-71. GPT0ACAPTSEL Register Field Descriptions... 373
Table 5-72. GPT0BCAPTSEL Register Field Descriptions... 376
Table 5-73. GPT1ACAPTSEL Register Field Descriptions... 379
Table 5-74. GPT1BCAPTSEL Register Field Descriptions... 382
Table 5-75. GPT2ACAPTSEL Register Field Descriptions... 385
Table 5-76. GPT2BCAPTSEL Register Field Descriptions... 388
Table 5-77. UDMACH1SSEL Register Field Descriptions... 391
Table 5-78. UDMACH1BSEL Register Field Descriptions... 392
Table 5-79. UDMACH2SSEL Register Field Descriptions... 393
Table 5-80. UDMACH2BSEL Register Field Descriptions... 394
Table 5-81. UDMACH3SSEL Register Field Descriptions... 395
Table 5-82. UDMACH3BSEL Register Field Descriptions... 396
Table 5-83. UDMACH4SSEL Register Field Descriptions... 397
Table 5-84. UDMACH4BSEL Register Field Descriptions... 398
Table 5-85. UDMACH5SSEL Register Field Descriptions... 399
Table 5-86. UDMACH5BSEL Register Field Descriptions... 400
Table 5-87. UDMACH6SSEL Register Field Descriptions... 401
Table 5-88. UDMACH6BSEL Register Field Descriptions... 402
Table 5-89. UDMACH7SSEL Register Field Descriptions... 403
Table 5-90. UDMACH7BSEL Register Field Descriptions... 404
Table 5-91. UDMACH8SSEL Register Field Descriptions... 405
Table 5-92. UDMACH8BSEL Register Field Descriptions... 406
Table 5-93. UDMACH9SSEL Register Field Descriptions... 407
Table 5-94. UDMACH9BSEL Register Field Descriptions... 408
Table 5-95. UDMACH10SSEL Register Field Descriptions... 409
Table 5-96. UDMACH10BSEL Register Field Descriptions... 410
Table 5-97. UDMACH11SSEL Register Field Descriptions... 411
Table 5-98. UDMACH11BSEL Register Field Descriptions... 412
Table 5-99. UDMACH12SSEL Register Field Descriptions... 413
Table 5-100. UDMACH12BSEL Register Field Descriptions... 414
Table 5-101. UDMACH13BSEL Register Field Descriptions... 415
Table 5-102. UDMACH14BSEL Register Field Descriptions... 416
Table 5-103. UDMACH15BSEL Register Field Descriptions... 421
Table 5-104. UDMACH16SSEL Register Field Descriptions... 422
Table 5-105. UDMACH16BSEL Register Field Descriptions... 423
Table 5-106. UDMACH17SSEL Register Field Descriptions... 424
Table 5-107. UDMACH17BSEL Register Field Descriptions... 425
Table 5-108. UDMACH19SSEL Register Field Descriptions... 426
Table 5-109. UDMACH19BSEL Register Field Descriptions... 427
Table 5-110. UDMACH20SSEL Register Field Descriptions... 428
Table 5-111. UDMACH20BSEL Register Field Descriptions... 429
Table 5-112. UDMACH21SSEL Register Field Descriptions... 430
Table 5-113. UDMACH21BSEL Register Field Descriptions... 431

Table 5-114. UDMACH22SSEL Register Field Descriptions... 432
Table 5-115. UDMACH22BSEL Register Field Descriptions... 433
Table 5-116. UDMACH23SSEL Register Field Descriptions... 434
Table 5-117. UDMACH23BSEL Register Field Descriptions... 435
Table 5-118. UDMACH24SSEL Register Field Descriptions... 436
Table 5-119. UDMACH24BSEL Register Field Descriptions... 437
Table 5-120. UDMACH25SSEL Register Field Descriptions... 438
Table 5-121. UDMACH25BSEL Register Field Descriptions... 439
Table 5-122. UDMACH26SSEL Register Field Descriptions... 440
Table 5-123. UDMACH26BSEL Register Field Descriptions... 441
Table 5-124. UDMACH28SSEL Register Field Descriptions... 442
Table 5-125. UDMACH28BSEL Register Field Descriptions... 443
Table 5-126. UDMACH29SSEL Register Field Descriptions... 444
Table 5-127. UDMACH29BSEL Register Field Descriptions... 445
Table 5-128. UDMACH30SSEL Register Field Descriptions... 446
Table 5-129. UDMACH30BSEL Register Field Descriptions... 447
Table 5-130. UDMACH31SSEL Register Field Descriptions... 448
Table 5-131. UDMACH31BSEL Register Field Descriptions... 449
Table 5-132. GPT3ACAPTSEL Register Field Descriptions... 450
Table 5-133. GPT3BCAPTSEL Register Field Descriptions... 453
Table 5-134. AUXSEL0 Register Field Descriptions... 456
Table 5-135. CM3NMISEL0 Register Field Descriptions... 457
Table 5-136. I2SSTMPSEL0 Register Field Descriptions... 458
Table 5-137. FRZSEL0 Register Field Descriptions... 459
Table 5-138. SWEV Register Field Descriptions... 460
Table 6-1. References... 461
Table 6-2. IEEE 1149.7 Feature Subset... 464
Table 6-3. OScan Scan Packet Contents... 465
Table 6-4. Slave TAP Order... 466
Table 6-5. Register Summary... 467
Table 6-6. Instruction Register Opcodes... 467
Table 6-7. Device Identification Register Description... 469
Table 6-8. User Code Register Description... 469
Table 6-9. ICEPick Identification Register Description... 469
Table 6-10. Connect Register Description... 470
Table 6-11. ROUTER DR Scan Chain Description... 470
Table 6-12. Control Block Registers... 471
Table 6-13. All0s Register... 472
Table 6-14. ICEPick Control Register... 472
Table 6-15. ICEPick Linking Mode Register... 472
Table 6-16. ICEPick TAP Link Mode... 472
Table 6-17. Test TAP Linking Registers... 473
Table 6-18. Secondary Test TAP Register (STTR)... 473
Table 6-19. Debug TAP Linking Registers... 473
Table 6-20. Secondary Debug TAP Register (SDTR)... 474
Table 6-21. Reset Control... 475
Table 6-22. Boundary Scan I/O for Different Devices... 477
Table 7-1. Power Saving Features... 480
Table 7-2. Power Modes in TI's Power Manager... 480
Table 7-3. System CPU Modes... 481
Table 7-4. System Clocks... 485
Table 7-5. Power Modes as Defined in TI's Power Manager... 490
Table 7-6. Example Sequence for Setting CC13x4x10 and CC26x4x10 in Standby Mode... 492
Table 7-7. Example Sequence for Going to Shutdown... 494
Table 7-8. PRCM Registers... 496
Table 7-9. PRCM Access Type Codes... 498
Table 7-10. INFRCLKDIVR Register Field Descriptions... 499
Table 7-11. INFRCLKDIVS Register Field Descriptions... 500
Table 7-12. INFRCLKDIVDS Register Field Descriptions... 501
Table 7-13. VDCTL Register Field Descriptions... 502
Table 7-14. CLKLOADCTL Register Field Descriptions... 503

Table 7-15. RFCCLKG Register Field Descriptions....504
Table 7-16. VIMSCLKG Register Field Descriptions....505
Table 7-17. SECDMACLKGR Register Field Descriptions....506
Table 7-18. SECDMACLKGS Register Field Descriptions....508
Table 7-19. SECDMACLKGDS Register Field Descriptions....509
Table 7-20. GPIOCLKGR Register Field Descriptions....510
Table 7-21. GPIOCLKGS Register Field Descriptions....511
Table 7-22. GPIOCLKGDS Register Field Descriptions....512
Table 7-23. GPTCLKGR Register Field Descriptions....513
Table 7-24. GPTCLKGS Register Field Descriptions....514
Table 7-25. GPTCLKGDS Register Field Descriptions....515
Table 7-26. I2CCLKGR Register Field Descriptions....516
Table 7-27. I2CCLKGS Register Field Descriptions....517
Table 7-28. I2CCLKGDS Register Field Descriptions....518
Table 7-29. UARTCLKGR Register Field Descriptions....519
Table 7-30. UARTCLKGS Register Field Descriptions....520
Table 7-31. UARTCLKGDS Register Field Descriptions....521
Table 7-32. SPICLKGR Register Field Descriptions....522
Table 7-33. SPICLKGS Register Field Descriptions....523
Table 7-34. SPICLKGDS Register Field Descriptions....524
Table 7-35. I2SCLKGR Register Field Descriptions....525
Table 7-36. I2SCLKGS Register Field Descriptions....526
Table 7-37. I2SCLKGDS Register Field Descriptions....527
Table 7-38. SYSBUSCLKDIV Register Field Descriptions....528
Table 7-39. CPUCLKDIV Register Field Descriptions....529
Table 7-40. PERBUSCPUCLKDIV Register Field Descriptions....530
Table 7-41. PERDMACLKDIV Register Field Descriptions....531
Table 7-42. I2SBCLKSEL Register Field Descriptions....532
Table 7-43. GPTCLKDIV Register Field Descriptions....533
Table 7-44. I2SCLKCTL Register Field Descriptions....534
Table 7-45. I2SMCLKDIV Register Field Descriptions....535
Table 7-46. I2SBCLKDIV Register Field Descriptions....536
Table 7-47. I2SWCLKDIV Register Field Descriptions....537
Table 7-48. RESETSECDMA Register Field Descriptions....538
Table 7-49. RESETGPIO Register Field Descriptions....539
Table 7-50. RESETGPT Register Field Descriptions....540
Table 7-51. RESETI2C Register Field Descriptions....541
Table 7-52. RESETUART Register Field Descriptions....542
Table 7-53. RESETSPI Register Field Descriptions....543
Table 7-54. RESETI2S Register Field Descriptions....544
Table 7-55. PDCTL0 Register Field Descriptions....545
Table 7-56. PDCTL0RFC Register Field Descriptions....546
Table 7-57. PDCTL0SERIAL Register Field Descriptions....547
Table 7-58. PDCTL0PERIPH Register Field Descriptions....548
Table 7-59. PDSTAT0 Register Field Descriptions....549
Table 7-60. PDSTAT0RFC Register Field Descriptions....550
Table 7-61. PDSTAT0SERIAL Register Field Descriptions....551
Table 7-62. PDSTAT0PERIPH Register Field Descriptions....552
Table 7-63. PDCTL1 Register Field Descriptions....553
Table 7-64. PDCTL1CPU Register Field Descriptions....554
Table 7-65. PDCTL1RFC Register Field Descriptions....555
Table 7-66. PDCTL1VIMS Register Field Descriptions....556
Table 7-67. PDSTAT1 Register Field Descriptions....557
Table 7-68. PDSTAT1BUS Register Field Descriptions....558
Table 7-69. PDSTAT1RFC Register Field Descriptions....559
Table 7-70. PDSTAT1CPU Register Field Descriptions....560
Table 7-71. PDSTAT1VIMS Register Field Descriptions....561
Table 7-72. RFCBITS Register Field Descriptions....562
Table 7-73. RFCMODESEL Register Field Descriptions....563
Table 7-74. RFCMODEHWOPT Register Field Descriptions....564
Table 7-75. PWRPROFSTAT Register Field Descriptions....565

Table 7-76. MCUSRAMCFG Register Field Descriptions....566
Table 7-77. RAMRETEN Register Field Descriptions....567
Table 7-78. OSCIMSC Register Field Descriptions....568
Table 7-79. OSCRIS Register Field Descriptions....569
Table 7-80. OSCICR Register Field Descriptions....572
Table 7-81. NVMNSCADDR Register Field Descriptions....573
Table 7-82. NVMNSADDR Register Field Descriptions....574
Table 7-83. SRAMNSCADDR Register Field Descriptions....575
Table 7-84. SRAMNSADDR Register Field Descriptions....576
Table 7-85. BUSSECCFG Register Field Descriptions....577
Table 7-86. CPULOCK Register Field Descriptions....578
Table 7-87. AON_PMCTL Registers....579
Table 7-88. AON_PMCTL Access Type Codes....579
Table 7-89. AUXSCECLK Register Field Descriptions....580
Table 7-90. RAMCFG Register Field Descriptions....581
Table 7-91. PWRCTL Register Field Descriptions....582
Table 7-92. PWRSTAT Register Field Descriptions....583
Table 7-93. SHUTDOWN Register Field Descriptions....584
Table 7-94. RECHARGECFG Register Field Descriptions....585
Table 7-95. RECHARGESTAT Register Field Descriptions....586
Table 7-96. OSCCFG Register Field Descriptions....587
Table 7-97. RESETCTL Register Field Descriptions....588
Table 7-98. SLEEPCTL Register Field Descriptions....590
Table 7-99. JTAGCFG Register Field Descriptions....591
Table 7-100. JTAGUSERCODE Register Field Descriptions....592
Table 7-101. WDTLOAD Register Field Descriptions....593
Table 7-102. WDTTEST Register Field Descriptions....594
Table 7-103. WDTLOCK Register Field Descriptions....595
Table 7-104. DDI_0_OSC Registers....596
Table 7-105. DDI_0_OSC Access Type Codes....596
Table 7-106. CTL0 Register Field Descriptions....597
Table 7-107. CTL1 Register Field Descriptions....599
Table 7-108. RADCEXTCFG Register Field Descriptions....600
Table 7-109. AMPCOMPCTL Register Field Descriptions....601
Table 7-110. AMPCOMPTH1 Register Field Descriptions....602
Table 7-111. AMPCOMPTH2 Register Field Descriptions....603
Table 7-112. ANABYPASSVAL1 Register Field Descriptions....604
Table 7-113. ANABYPASSVAL2 Register Field Descriptions....605
Table 7-114. ATESTCTL Register Field Descriptions....606
Table 7-115. ADCDOUBLERNANOAMPCTL Register Field Descriptions....607
Table 7-116. XOSCHFCTL Register Field Descriptions....608
Table 7-117. LFOSCCTL Register Field Descriptions....609
Table 7-118. RCOSCHFCTL Register Field Descriptions....610
Table 7-119. RCOSCMFCTL Register Field Descriptions....611
Table 7-120. STAT0 Register Field Descriptions....612
Table 7-121. STAT1 Register Field Descriptions....613
Table 7-122. STAT2 Register Field Descriptions....614
Table 8-1. Valid Retention Combination for VIMS Memory....620
Table 8-2. CC13x4x10 and CC26x4x10 Memory Write/Erase Protection....622
Table 8-3. FLASH Registers....623
Table 8-4. FLASH Access Type Codes....624
Table 8-5. WEPROT_B0_31_0_BY1 Register Field Descriptions....625
Table 8-6. WEPROT_AUX_BY1 Register Field Descriptions....626
Table 8-7. STAT Register Field Descriptions....627
Table 8-8. CFG Register Field Descriptions....628
Table 8-9. FLASH_SIZE Register Field Descriptions....629
Table 8-10. FWLOCK Register Field Descriptions....630
Table 8-11. FWFLAG Register Field Descriptions....631
Table 8-12. BANK0_TRIM_CFG_3 Register Field Descriptions....632
Table 8-13. BANK0_TRIM_CFG_2 Register Field Descriptions....633
Table 8-14. BANK0_TRIM_CFG_1 Register Field Descriptions....634

Table 8-15. BANK0_TRIM_CFG_0 Register Field Descriptions....635
Table 8-16. BANK1_TRIM_CFG_3 Register Field Descriptions....636
Table 8-17. BANK1_TRIM_CFG_2 Register Field Descriptions....637
Table 8-18. BANK1_TRIM_CFG_1 Register Field Descriptions....638
Table 8-19. BANK1_TRIM_CFG_0 Register Field Descriptions....639
Table 8-20. PUMP_TRIM_CFG_2 Register Field Descriptions....640
Table 8-21. PUMP_TRIM_CFG_1 Register Field Descriptions....641
Table 8-22. PUMP_TRIM_CFG_0 Register Field Descriptions....642
Table 8-23. EFUSE Register Field Descriptions....643
Table 8-24. EFUSEADDR Register Field Descriptions....644
Table 8-25. DATAUPPER Register Field Descriptions....645
Table 8-26. DATALOWER Register Field Descriptions....646
Table 8-27. EFUSECFG Register Field Descriptions....647
Table 8-28. EFUSESTAT Register Field Descriptions....648
Table 8-29. ACC Register Field Descriptions....649
Table 8-30. BOUNDARY Register Field Descriptions....650
Table 8-31. EFUSEFLAG Register Field Descriptions....651
Table 8-32. EFUSEKEY Register Field Descriptions....652
Table 8-33. EFUSERELEASE Register Field Descriptions....653
Table 8-34. EFUSEPINS Register Field Descriptions....654
Table 8-35. EFUSECRA Register Field Descriptions....655
Table 8-36. EFUSEREAD Register Field Descriptions....656
Table 8-37. EFUSEPROGRAM Register Field Descriptions....657
Table 8-38. EFUSEERROR Register Field Descriptions....658
Table 8-39. SINGLEBIT Register Field Descriptions....659
Table 8-40. TWOBIT Register Field Descriptions....660
Table 8-41. SELFTESTCYC Register Field Descriptions....661
Table 8-42. SELFTESTSIGN Register Field Descriptions....662
Table 8-43. VIMS Registers....663
Table 8-44. VIMS Access Type Codes....663
Table 8-45. STAT Register Field Descriptions....664
Table 8-46. CTL Register Field Descriptions....665
Table 8-47. NVMNW Registers....666
Table 8-48. NVMNW Access Type Codes....667
Table 8-49. IIDX Register Field Descriptions....668
Table 8-50. IMASK Register Field Descriptions....669
Table 8-51. RIS Register Field Descriptions....670
Table 8-52. MIS Register Field Descriptions....671
Table 8-53. ISET Register Field Descriptions....672
Table 8-54. ICLR Register Field Descriptions....673
Table 8-55. EVT_MODE Register Field Descriptions....674
Table 8-56. DESC Register Field Descriptions....675
Table 8-57. CMDEXEC Register Field Descriptions....676
Table 8-58. CMDTYPE Register Field Descriptions....677
Table 8-59. CMDCTL Register Field Descriptions....678
Table 8-60. CMDADDR Register Field Descriptions....679
Table 8-61. CMDBYTEN Register Field Descriptions....680
Table 8-62. CMDDATAINDEX Register Field Descriptions....681
Table 8-63. CMDDATA0 Register Field Descriptions....682
Table 8-64. CMDDATA1 Register Field Descriptions....683
Table 8-65. CMDDATA2 Register Field Descriptions....684
Table 8-66. CMDDATA3 Register Field Descriptions....685
Table 8-67. CMDDATA4 Register Field Descriptions....686
Table 8-68. CMDDATA5 Register Field Descriptions....687
Table 8-69. CMDDATA6 Register Field Descriptions....688
Table 8-70. CMDDATA7 Register Field Descriptions....689
Table 8-71. CMDDATA8 Register Field Descriptions....690
Table 8-72. CMDDATA9 Register Field Descriptions....691
Table 8-73. CMDDATA10 Register Field Descriptions....692
Table 8-74. CMDDATA11 Register Field Descriptions....693
Table 8-75. CMDDATA12 Register Field Descriptions....694

Table 8-76. CMDDATA13 Register Field Descriptions....695
Table 8-77. CMDDATA14 Register Field Descriptions....696
Table 8-78. CMDDATA15 Register Field Descriptions....697
Table 8-79. CMDWEPROTA Register Field Descriptions....698
Table 8-80. CMDWEPROTB Register Field Descriptions....699
Table 8-81. CMDWEPROTNM Register Field Descriptions....700
Table 8-82. CMDWEPROTTR Register Field Descriptions....701
Table 8-83. CMDWEPROTEN Register Field Descriptions....702
Table 8-84. CFGCMD Register Field Descriptions....703
Table 8-85. CFGPCNT Register Field Descriptions....704
Table 8-86. STATCMD Register Field Descriptions....705
Table 8-87. STATADDR Register Field Descriptions....706
Table 8-88. STATPCNT Register Field Descriptions....707
Table 8-89. STATMODE Register Field Descriptions....708
Table 8-90. GBLINFO0 Register Field Descriptions....709
Table 8-91. GBLINFO1 Register Field Descriptions....710
Table 8-92. GBLINFO2 Register Field Descriptions....711
Table 8-93. BANK0INFO0 Register Field Descriptions....712
Table 8-94. BANK0INFO1 Register Field Descriptions....713
Table 8-95. BANK1INFO0 Register Field Descriptions....714
Table 8-96. BANK1INFO1 Register Field Descriptions....715
Table 9-1. SRAM_MMR Registers....719
Table 9-2. SRAM_MMR Access Type Codes....719
Table 9-3. PER_CTL Register Field Descriptions....720
Table 9-4. PER_CHK Register Field Descriptions....721
Table 9-5. PER_DBG Register Field Descriptions....722
Table 9-6. MEM_CTL Register Field Descriptions....723
Table 9-7. MEM_STA Register Field Descriptions....724
Table 9-8. SRAM Registers....725
Table 9-9. SRAM Access Type Codes....725
Table 9-10. BANK0_y Register Field Descriptions....726
Table 9-11. BANK1_y Register Field Descriptions....727
Table 9-12. BANK2_y Register Field Descriptions....728
Table 9-13. BANK3_y Register Field Descriptions....729
Table 9-14. BANK4_y Register Field Descriptions....730
Table 9-15. BANK5_y Register Field Descriptions....731
Table 9-16. BANK6_y Register Field Descriptions....732
Table 9-17. BANK7_y Register Field Descriptions....733
Table 10-1. Protocol Acknowledge and Not-Acknowledge Bytes....738
Table 10-2. DIO Configuration of Serial Interfaces....739
Table 10-3. Supported Bootloader Commands....740
Table 10-4. Defined Status Values....742
Table 10-5. Defined CCFG Field IDs and Field Values....749
Table 11-1. CCFG Registers....756
Table 11-2. CCFG Access Type Codes....756
Table 11-3. SIZE_AND_DIS_FLAGS Register Field Descriptions....757
Table 11-4. MODE_CONF Register Field Descriptions....758
Table 11-5. MODE_CONF_1 Register Field Descriptions....760
Table 11-6. VOLT_LOAD_0 Register Field Descriptions....761
Table 11-7. VOLT_LOAD_1 Register Field Descriptions....762
Table 11-8. EXT_LF_CLK Register Field Descriptions....763
Table 11-9. IEEE_MAC_0 Register Field Descriptions....764
Table 11-10. IEEE_MAC_1 Register Field Descriptions....765
Table 11-11. IEEE_BLE_0 Register Field Descriptions....766
Table 11-12. IEEE_BLE_1 Register Field Descriptions....767
Table 11-13. BL_CONFIG Register Field Descriptions....768
Table 11-14. ERASE_CONF Register Field Descriptions....769
Table 11-15. ERASE_CONF_1 Register Field Descriptions....770
Table 11-16. CCFG_TI_OPTIONS Register Field Descriptions....771
Table 11-17. CCFG_TAP_DAP_0 Register Field Descriptions....772
Table 11-18. CCFG_TAP_DAP_1 Register Field Descriptions....773

Table 11-19. IMAGE_VALID_CONF Register Field Descriptions ... 774
Table 11-20. CCFG_WEPROT_31_0_BY2K Register Field Descriptions ... 775
Table 11-21. CCFG_WEPROT_SPARE_1 Register Field Descriptions ... 776
Table 11-22. CCFG_WEPROT_SPARE_2 Register Field Descriptions ... 777
Table 11-23. CCFG_WEPROT_SPARE_3 Register Field Descriptions ... 778
Table 11-24. TRUSTZONE_FLASH_CFG Register Field Descriptions ... 779
Table 11-25. TRUSTZONE_SRAM_CFG Register Field Descriptions ... 780
Table 11-26. SRAM_CFG Register Field Descriptions ... 781
Table 11-27. CPU_LOCK_CFG Register Field Descriptions ... 782
Table 11-28. DEB_AUTH_CFG Register Field Descriptions ... 783
Table 11-29. CKEY0 Register Field Descriptions ... 784
Table 11-30. CKEY1 Register Field Descriptions ... 785
Table 11-31. CKEY2 Register Field Descriptions ... 786
Table 11-32. CKEY3 Register Field Descriptions ... 787
Table 11-33. FCFG1 Registers ... 789
Table 11-34. FCFG1 Access Type Codes ... 790
Table 11-35. MISC_CONF_1 Register Field Descriptions ... 791
Table 11-36. MISC_CONF_2 Register Field Descriptions ... 792
Table 11-37. HPOSC_MEAS_5 Register Field Descriptions ... 793
Table 11-38. HPOSC_MEAS_4 Register Field Descriptions ... 794
Table 11-39. HPOSC_MEAS_3 Register Field Descriptions ... 795
Table 11-40. HPOSC_MEAS_2 Register Field Descriptions ... 796
Table 11-41. HPOSC_MEAS_1 Register Field Descriptions ... 797
Table 11-42. CONFIG_CC26_FE Register Field Descriptions ... 798
Table 11-43. CONFIG_CC13_FE Register Field Descriptions ... 799
Table 11-44. CONFIG_RF_COMMON Register Field Descriptions ... 800
Table 11-45. CONFIG_SYNTH_DIV2_CC26_2G4 Register Field Descriptions ... 801
Table 11-46. CONFIG_SYNTH_DIV2_CC13_2G4 Register Field Descriptions ... 802
Table 11-47. CONFIG_SYNTH_DIV2_CC26_1G Register Field Descriptions ... 803
Table 11-48. CONFIG_SYNTH_DIV2_CC13_1G Register Field Descriptions ... 804
Table 11-49. CONFIG_SYNTH_DIV4_CC26 Register Field Descriptions ... 805
Table 11-50. CONFIG_SYNTH_DIV4_CC13 Register Field Descriptions ... 806
Table 11-51. CONFIG_SYNTH_DIV5 Register Field Descriptions ... 807
Table 11-52. CONFIG_SYNTH_DIV6_CC26 Register Field Descriptions ... 808
Table 11-53. CONFIG_SYNTH_DIV6_CC13 Register Field Descriptions ... 809
Table 11-54. CONFIG_SYNTH_DIV10 Register Field Descriptions ... 810
Table 11-55. CONFIG_SYNTH_DIV12_CC26 Register Field Descriptions ... 811
Table 11-56. CONFIG_SYNTH_DIV12_CC13 Register Field Descriptions ... 812
Table 11-57. CONFIG_SYNTH_DIV15 Register Field Descriptions ... 813
Table 11-58. CONFIG_SYNTH_DIV30 Register Field Descriptions ... 814
Table 11-59. IOCONF Register Field Descriptions ... 815
Table 11-60. USER_ID Register Field Descriptions ... 816
Table 11-61. FLASH_OTP_DATA3 Register Field Descriptions ... 817
Table 11-62. ANA2_TRIM Register Field Descriptions ... 818
Table 11-63. LDO_TRIM Register Field Descriptions ... 819
Table 11-64. MAC_BLE_0 Register Field Descriptions ... 820
Table 11-65. MAC_BLE_1 Register Field Descriptions ... 821
Table 11-66. MAC_15_4_0 Register Field Descriptions ... 822
Table 11-67. MAC_15_4_1 Register Field Descriptions ... 823
Table 11-68. MISC_TRIM Register Field Descriptions ... 824
Table 11-69. RCOSC_HF_TEMPCOMP Register Field Descriptions ... 825
Table 11-70. ICEPICK_DEVICE_ID Register Field Descriptions ... 826
Table 11-71. FCFG1_REVISION Register Field Descriptions ... 827
Table 11-72. MISC_OTP_DATA Register Field Descriptions ... 828
Table 11-73. CONFIG_IF_ADC Register Field Descriptions ... 829
Table 11-74. CONFIG_OSC_TOP Register Field Descriptions ... 830
Table 11-75. SOC_ADC_ABS_GAIN Register Field Descriptions ... 831
Table 11-76. SOC_ADC_REL_GAIN Register Field Descriptions ... 832
Table 11-77. SOC_ADC_OFFSET_INT Register Field Descriptions ... 833
Table 11-78. SOC_ADC_REF_TRIM_AND_OFFSET_EXT Register Field Descriptions ... 834
Table 11-79. AMPCOMP_TH1 Register Field Descriptions ... 835

Table 11-80. AMPCOMP_TH2 Register Field Descriptions....836
Table 11-81. AMPCOMP_CTRL1 Register Field Descriptions....837
Table 11-82. ANABYPASS_VALUE2 Register Field Descriptions....838
Table 11-83. VOLT_TRIM Register Field Descriptions....839
Table 11-84. OSC_CONF Register Field Descriptions....840
Table 11-85. FREQ_OFFSET Register Field Descriptions....841
Table 11-86. MISC_OTP_DATA_1 Register Field Descriptions....842
Table 11-87. SHDW_DIE_ID_0 Register Field Descriptions....843
Table 11-88. SHDW_DIE_ID_1 Register Field Descriptions....844
Table 11-89. SHDW_DIE_ID_2 Register Field Descriptions....845
Table 11-90. SHDW_DIE_ID_3 Register Field Descriptions....846
Table 11-91. SHDW_SCAN_MCU3_SEC Register Field Descriptions....847
Table 11-92. SHDW_SCAN_DATA1_CRC Register Field Descriptions....848
Table 11-93. SHDW_ANA_TRIM Register Field Descriptions....849
Table 11-94. OSC_CONF1 Register Field Descriptions....850
Table 11-95. DAC_BIAS_CNF Register Field Descriptions....851
Table 11-96. TFW_PROBE Register Field Descriptions....852
Table 11-97. TFW_FT Register Field Descriptions....853
Table 11-98. DAC_CAL0 Register Field Descriptions....854
Table 11-99. DAC_CAL1 Register Field Descriptions....855
Table 11-100. DAC_CAL2 Register Field Descriptions....856
Table 11-101. DAC_CAL3 Register Field Descriptions....857
Table 12-1. Detailed Memory Map(1) 862
Table 12-2. Valid Combinations for ALGSEL Flags....865
Table 12-3. AES_KEY....866
Table 12-4. AES_KEY....867
Table 12-5. For CCM....867
Table 12-6. For CBC-MAC....867
Table 12-7. For GCM....867
Table 12-8. AES Initialization Vector Registers....867
Table 12-9. Initialization Vector, Used for CBC and CTR Modes....868
Table 12-10. For GCM....868
Table 12-11. Initialization Vector, Used for CCM....868
Table 12-12. Initialization Vector, Used for CBC-MAC....868
Table 12-13. Input/Output Block Format per Operating Mode....869
Table 12-14. AES Tag Output Register....870
Table 12-15. For GCM, CCM, and CBC-MAC....870
Table 12-16. Input/Digest Block Format for Hash Algorithms....872
Table 12-17. Supported DMAC Operations....875
Table 12-18. Performance Table for DMA-Based Operations....877
Table 12-19. Acronyms....907
Table 12-20. Formulas and Nomenclature....908
Table 12-21. CRYPTO Registers....910
Table 12-22. CRYPTO Access Type Codes....912
Table 12-23. DMACH0CTL Register Field Descriptions....913
Table 12-24. DMACH0EXTADDR Register Field Descriptions....914
Table 12-25. DMACH0LEN Register Field Descriptions....915
Table 12-26. DMASTAT Register Field Descriptions....916
Table 12-27. DMASWRESET Register Field Descriptions....917
Table 12-28. DMACH1CTL Register Field Descriptions....918
Table 12-29. DMACH1EXTADDR Register Field Descriptions....919
Table 12-30. DMACH1LEN Register Field Descriptions....920
Table 12-31. DMABUSCFG Register Field Descriptions....921
Table 12-32. DMAPORTERR Register Field Descriptions....922
Table 12-33. DMAHWVER Register Field Descriptions....923
Table 12-34. KEYWRITEAREA Register Field Descriptions....924
Table 12-35. KEYWRITTENAREA Register Field Descriptions....926
Table 12-36. KEYSIZE Register Field Descriptions....928
Table 12-37. KEYREADAREA Register Field Descriptions....929
Table 12-38. AESKEY2_y Register Field Descriptions....930
Table 12-39. AESKEY3_y Register Field Descriptions....931

Table 12-40. AESIV_y Register Field Descriptions....932
Table 12-41. AESCTL Register Field Descriptions....933
Table 12-42. AESDATALEN0 Register Field Descriptions....936
Table 12-43. AESDATALEN1 Register Field Descriptions....937
Table 12-44. AESAUTHLEN Register Field Descriptions....938
Table 12-45. AESDATAOUT0 Register Field Descriptions....939
Table 12-46. AESDATAIN0 Register Field Descriptions....940
Table 12-47. AESDATAOUT1 Register Field Descriptions....941
Table 12-48. AESDATAIN1 Register Field Descriptions....942
Table 12-49. AESDATAOUT2 Register Field Descriptions....943
Table 12-50. AESDATAIN2 Register Field Descriptions....944
Table 12-51. AESDATAOUT3 Register Field Descriptions....945
Table 12-52. AESDATAIN3 Register Field Descriptions....946
Table 12-53. AESTAGOUT_y Register Field Descriptions....947
Table 12-54. AESCCMALNWRD Register Field Descriptions....948
Table 12-55. AESBLKCNT0 Register Field Descriptions....949
Table 12-56. AESBLKCNT1 Register Field Descriptions....950
Table 12-57. HASHDATAIN0 Register Field Descriptions....951
Table 12-58. HASHDATAIN1 Register Field Descriptions....952
Table 12-59. HASHDATAIN2 Register Field Descriptions....953
Table 12-60. HASHDATAIN3 Register Field Descriptions....954
Table 12-61. HASHDATAIN4 Register Field Descriptions....955
Table 12-62. HASHDATAIN5 Register Field Descriptions....956
Table 12-63. HASHDATAIN6 Register Field Descriptions....957
Table 12-64. HASHDATAIN7 Register Field Descriptions....958
Table 12-65. HASHDATAIN8 Register Field Descriptions....959
Table 12-66. HASHDATAIN9 Register Field Descriptions....960
Table 12-67. HASHDATAIN10 Register Field Descriptions....961
Table 12-68. HASHDATAIN11 Register Field Descriptions....962
Table 12-69. HASHDATAIN12 Register Field Descriptions....963
Table 12-70. HASHDATAIN13 Register Field Descriptions....964
Table 12-71. HASHDATAIN14 Register Field Descriptions....965
Table 12-72. HASHDATAIN15 Register Field Descriptions....966
Table 12-73. HASHDATAIN16 Register Field Descriptions....967
Table 12-74. HASHDATAIN17 Register Field Descriptions....968
Table 12-75. HASHDATAIN18 Register Field Descriptions....969
Table 12-76. HASHDATAIN19 Register Field Descriptions....970
Table 12-77. HASHDATAIN20 Register Field Descriptions....971
Table 12-78. HASHDATAIN21 Register Field Descriptions....972
Table 12-79. HASHDATAIN22 Register Field Descriptions....973
Table 12-80. HASHDATAIN23 Register Field Descriptions....974
Table 12-81. HASHDATAIN24 Register Field Descriptions....975
Table 12-82. HASHDATAIN25 Register Field Descriptions....976
Table 12-83. HASHDATAIN26 Register Field Descriptions....977
Table 12-84. HASHDATAIN27 Register Field Descriptions....978
Table 12-85. HASHDATAIN28 Register Field Descriptions....979
Table 12-86. HASHDATAIN29 Register Field Descriptions....980
Table 12-87. HASHDATAIN30 Register Field Descriptions....981
Table 12-88. HASHDATAIN31 Register Field Descriptions....982
Table 12-89. HASHIOBUFCTRL Register Field Descriptions....983
Table 12-90. HASHMODE Register Field Descriptions....985
Table 12-91. HASHINLENL Register Field Descriptions....986
Table 12-92. HASHINLENH Register Field Descriptions....987
Table 12-93. HASHDIGESTA Register Field Descriptions....988
Table 12-94. HASHDIGESTB Register Field Descriptions....989
Table 12-95. HASHDIGESTC Register Field Descriptions....990
Table 12-96. HASHDIGESTD Register Field Descriptions....991
Table 12-97. HASHDIGESTE Register Field Descriptions....992
Table 12-98. HASHDIGESTF Register Field Descriptions....993
Table 12-99. HASHDIGESTG Register Field Descriptions....994
Table 12-100. HASHDIGESTH Register Field Descriptions....995

Table 12-101. HASHDIGESTI Register Field Descriptions....996
Table 12-102. HASHDIGESTJ Register Field Descriptions....997
Table 12-103. HASHDIGESTK Register Field Descriptions....998
Table 12-104. HASHDIGESTL Register Field Descriptions....999
Table 12-105. HASHDIGESTM Register Field Descriptions....1000
Table 12-106. HASHDIGESTN Register Field Descriptions....1001
Table 12-107. HASHDIGESTO Register Field Descriptions....1002
Table 12-108. HASHDIGESTP Register Field Descriptions....1003
Table 12-109. ALGSEL Register Field Descriptions....1004
Table 12-110. DMAPROTCTL Register Field Descriptions....1005
Table 12-111. SWRESET Register Field Descriptions....1006
Table 12-112. IRQTYPE Register Field Descriptions....1007
Table 12-113. IRQEN Register Field Descriptions....1008
Table 12-114. IRQCLR Register Field Descriptions....1009
Table 12-115. IRQSET Register Field Descriptions....1010
Table 12-116. IRQSTAT Register Field Descriptions....1011
Table 12-117. HWVER Register Field Descriptions....1012
Table 13-1. PKCP Operations(1) 1016
Table 13-2. Restrictions on Input Vectors for PKCP Operations....1016
Table 13-3. Result Vector Memory Allocation....1017
Table 13-4. PKCP Result Vector and Vector Overlap Restrictions....1017
Table 13-5. ExpMod Operations....1018
Table 13-6. Operational Restrictions....1018
Table 13-7. Result Vector and Scratchpad Area Memory Allocation (Starting at PKA_DPTR)....1019
Table 13-8. Overlap Restrictions of Result and Input Vectors....1019
Table 13-9. Required RAM Sizes....1019
Table 13-10. Maximum Number of Odd-Numbered Powers....1020
Table 13-11. Example PKA RAM Vector Allocations....1020
Table 13-12. Extension of Basic PKCP Functions....1021
Table 13-13. PKA_SHIFT Result Values....1021
Table 13-14. Operational Restrictions....1021
Table 13-15. Overlap Restrictions of Result and Input Vectors....1021
Table 13-16. Result and Vector Scratchpad Area Memory Allocation(1) 1021
Table 13-17. ECC Operations....1022
Table 13-18. PKA_SHIFT Result Values....1023
Table 13-19. Operational Restrictions....1023
Table 13-20. Overlap Restrictions of Input and Output Vectors....1023
Table 13-21. Memory Allocation of Result Vector and Scratchpad Area(1) (2) 1023
Table 13-22. PKA RAM Vector Allocations....1024
Table 13-23. Clocks for PKCP Operations....1026
Table 13-24. ExpMod Performance Values....1027
Table 13-25. Performance Values of Five Simulations....1027
Table 13-26. ECC Operation Performance Values....1028
Table 13-27. ECC-MUL Operation Performance Values....1028
Table 13-28. PKA Registers....1029
Table 13-29. PKA Access Type Codes....1029
Table 13-30. APTR Register Field Descriptions....1030
Table 13-31. BPTR Register Field Descriptions....1031
Table 13-32. CPTR Register Field Descriptions....1032
Table 13-33. DPTR Register Field Descriptions....1033
Table 13-34. ALENGTH Register Field Descriptions....1034
Table 13-35. BLENGTH Register Field Descriptions....1035
Table 13-36. SHIFT Register Field Descriptions....1036
Table 13-37. FUNCTION Register Field Descriptions....1037
Table 13-38. COMPARE Register Field Descriptions....1039
Table 13-39. MSW Register Field Descriptions....1040
Table 13-40. DIVMSW Register Field Descriptions....1041
Table 13-41. SEQCTRL Register Field Descriptions....1042
Table 13-42. OPTIONS Register Field Descriptions....1043
Table 13-43. FWREV Register Field Descriptions....1044
Table 13-44. HWREV Register Field Descriptions....1045

Table 14-1. Events....1049
Table 14-2. Initialization of Surrounding Modules....1052
Table 14-3. TRNG Initialization Sequence....1052
Table 14-4. TRNG Interrupt Mode....1054
Table 14-5. TRNG Registers....1055
Table 14-6. TRNG Access Type Codes....1055
Table 14-7. OUT0 Register Field Descriptions....1056
Table 14-8. OUT1 Register Field Descriptions....1057
Table 14-9. IRQFLAGSTAT Register Field Descriptions....1058
Table 14-10. IRQFLAGMASK Register Field Descriptions....1059
Table 14-11. IRQFLAGCLR Register Field Descriptions....1060
Table 14-12. CTL Register Field Descriptions....1061
Table 14-13. CFG0 Register Field Descriptions....1062
Table 14-14. ALARMCNT Register Field Descriptions....1063
Table 14-15. FROEN Register Field Descriptions....1064
Table 14-16. FRODETUNE Register Field Descriptions....1065
Table 14-17. ALARMMASK Register Field Descriptions....1066
Table 14-18. ALARMSTOP Register Field Descriptions....1067
Table 14-19. LFSR0 Register Field Descriptions....1068
Table 14-20. LFSR1 Register Field Descriptions....1069
Table 14-21. LFSR2 Register Field Descriptions....1070
Table 14-22. HWOPT Register Field Descriptions....1071
Table 14-23. HWVER0 Register Field Descriptions....1072
Table 14-24. IRQSTATMASK Register Field Descriptions....1073
Table 14-25. HWVER1 Register Field Descriptions....1074
Table 14-26. IRQSET Register Field Descriptions....1075
Table 14-27. SWRESET Register Field Descriptions....1076
Table 14-28. IRQSTAT Register Field Descriptions....1077
Table 15-1. RF Core Data Signals for PA and LNA....1081
Table 15-2. CC13x4x10 and CC26x4x10 Pin Mapping....1084
Table 15-3. CC13x4x10 and CC26x4x10 PORT_IDs....1086
Table 15-4. AON_IOC Registers....1090
Table 15-5. AON_IOC Access Type Codes....1090
Table 15-6. IOSTRMIN Register Field Descriptions....1091
Table 15-7. IOSTRMED Register Field Descriptions....1092
Table 15-8. IOSTRMAX Register Field Descriptions....1093
Table 15-9. IOCLATCH Register Field Descriptions....1094
Table 15-10. CLK32KCTL Register Field Descriptions....1095
Table 15-11. TCKCTL Register Field Descriptions....1096
Table 15-12. GPIO Registers....1097
Table 15-13. GPIO Access Type Codes....1097
Table 15-14. DOUT3_0 Register Field Descriptions....1098
Table 15-15. DOUT7_4 Register Field Descriptions....1099
Table 15-16. DOUT11_8 Register Field Descriptions....1100
Table 15-17. DOUT15_12 Register Field Descriptions....1101
Table 15-18. DOUT19_16 Register Field Descriptions....1102
Table 15-19. DOUT23_20 Register Field Descriptions....1103
Table 15-20. DOUT27_24 Register Field Descriptions....1104
Table 15-21. DOUT31_28 Register Field Descriptions....1105
Table 15-22. DOUT35_32 Register Field Descriptions....1106
Table 15-23. DOUT39_36 Register Field Descriptions....1107
Table 15-24. DOUT43_40 Register Field Descriptions....1108
Table 15-25. DOUT47_44 Register Field Descriptions....1109
Table 15-26. DOUT31_0 Register Field Descriptions....1110
Table 15-27. DOUT47_32 Register Field Descriptions....1111
Table 15-28. DOUTSET31_0 Register Field Descriptions....1112
Table 15-29. DOUTSET47_32 Register Field Descriptions....1113
Table 15-30. DOUTCLR31_0 Register Field Descriptions....1114
Table 15-31. DOUTCLR47_32 Register Field Descriptions....1115
Table 15-32. DOUTTGL31_0 Register Field Descriptions....1116
Table 15-33. DOUTTGL47_32 Register Field Descriptions....1117

Table 15-34. DIN31_0 Register Field Descriptions.... 1118
Table 15-35. DIN47_32 Register Field Descriptions.... 1119
Table 15-36. DOE31_0 Register Field Descriptions.... 1120
Table 15-37. DOE47_32 Register Field Descriptions.... 1121
Table 15-38. EVFLAGS31_0 Register Field Descriptions.... 1122
Table 15-39. EVFLAGS47_32 Register Field Descriptions.... 1123
Table 15-40. IOC Registers.... 1124
Table 15-41. IOC Access Type Codes.... 1125
Table 15-42. IOCFG0 Register Field Descriptions.... 1126
Table 15-43. IOCFG1 Register Field Descriptions.... 1130
Table 15-44. IOCFG2 Register Field Descriptions.... 1134
Table 15-45. IOCFG3 Register Field Descriptions.... 1138
Table 15-46. IOCFG4 Register Field Descriptions.... 1142
Table 15-47. IOCFG5 Register Field Descriptions.... 1146
Table 15-48. IOCFG6 Register Field Descriptions.... 1150
Table 15-49. IOCFG7 Register Field Descriptions.... 1154
Table 15-50. IOCFG8 Register Field Descriptions.... 1158
Table 15-51. IOCFG9 Register Field Descriptions.... 1162
Table 15-52. IOCFG10 Register Field Descriptions.... 1166
Table 15-53. IOCFG11 Register Field Descriptions.... 1170
Table 15-54. IOCFG12 Register Field Descriptions.... 1174
Table 15-55. IOCFG13 Register Field Descriptions.... 1178
Table 15-56. IOCFG14 Register Field Descriptions.... 1182
Table 15-57. IOCFG15 Register Field Descriptions.... 1186
Table 15-58. IOCFG16 Register Field Descriptions.... 1190
Table 15-59. IOCFG17 Register Field Descriptions.... 1194
Table 15-60. IOCFG18 Register Field Descriptions.... 1198
Table 15-61. IOCFG19 Register Field Descriptions.... 1202
Table 15-62. IOCFG20 Register Field Descriptions.... 1206
Table 15-63. IOCFG21 Register Field Descriptions.... 1210
Table 15-64. IOCFG22 Register Field Descriptions.... 1214
Table 15-65. IOCFG23 Register Field Descriptions.... 1218
Table 15-66. IOCFG24 Register Field Descriptions.... 1222
Table 15-67. IOCFG25 Register Field Descriptions.... 1226
Table 15-68. IOCFG26 Register Field Descriptions.... 1230
Table 15-69. IOCFG27 Register Field Descriptions.... 1234
Table 15-70. IOCFG28 Register Field Descriptions.... 1238
Table 15-71. IOCFG29 Register Field Descriptions.... 1242
Table 15-72. IOCFG30 Register Field Descriptions.... 1246
Table 15-73. IOCFG31 Register Field Descriptions.... 1250
Table 15-74. IOCFG32 Register Field Descriptions.... 1254
Table 15-75. IOCFG33 Register Field Descriptions.... 1258
Table 15-76. IOCFG34 Register Field Descriptions.... 1262
Table 15-77. IOCFG35 Register Field Descriptions.... 1266
Table 15-78. IOCFG36 Register Field Descriptions.... 1270
Table 15-79. IOCFG37 Register Field Descriptions.... 1274
Table 15-80. IOCFG38 Register Field Descriptions.... 1278
Table 15-81. IOCFG39 Register Field Descriptions.... 1282
Table 15-82. IOCFG40 Register Field Descriptions.... 1286
Table 15-83. IOCFG41 Register Field Descriptions.... 1290
Table 15-84. IOCFG42 Register Field Descriptions.... 1294
Table 15-85. IOCFG43 Register Field Descriptions.... 1298
Table 15-86. IOCFG44 Register Field Descriptions.... 1302
Table 15-87. IOCFG45 Register Field Descriptions.... 1306
Table 15-88. IOCFG46 Register Field Descriptions.... 1310
Table 15-89. IOCFG47 Register Field Descriptions.... 1314
Table 16-1. Channel Assignments.... 1322
Table 16-2. Request Type Support.... 1324
Table 16-3. Control Structure Memory Map.... 1325
Table 16-4. Channel Control Structure.... 1326
Table 16-5. µDMA Read Example: 8-Bit Peripheral.... 1334

Table 16-6. µDMA Interrupt Assignments....1335
Table 16-7. UDMA Registers....1350
Table 16-8. UDMA Access Type Codes....1350
Table 16-9. STATUS Register Field Descriptions....1351
Table 16-10. CFG Register Field Descriptions....1352
Table 16-11. CTRL Register Field Descriptions....1353
Table 16-12. ALTCTRL Register Field Descriptions....1354
Table 16-13. WAITONREQ Register Field Descriptions....1355
Table 16-14. SOFTREQ Register Field Descriptions....1356
Table 16-15. SETBURST Register Field Descriptions....1357
Table 16-16. CLEARBURST Register Field Descriptions....1358
Table 16-17. SETREQMASK Register Field Descriptions....1359
Table 16-18. CLEARREQMASK Register Field Descriptions....1360
Table 16-19. SETCHANNELEN Register Field Descriptions....1361
Table 16-20. CLEARCHANNELEN Register Field Descriptions....1362
Table 16-21. SETCHNLPRIALT Register Field Descriptions....1363
Table 16-22. CLEARCHNLPRIALT Register Field Descriptions....1364
Table 16-23. SETCHNLPRIORITY Register Field Descriptions....1365
Table 16-24. CLEARCHNLPRIORITY Register Field Descriptions....1366
Table 16-25. ERROR Register Field Descriptions....1367
Table 16-26. REQDONE Register Field Descriptions....1368
Table 16-27. DONEMASK Register Field Descriptions....1369
Table 17-1. General-Purpose Timer Capabilities....1374
Table 17-2. 16-Bit Timer With Prescaler Configurations....1376
Table 17-3. Counter Values When the Timer is Enabled in Input Edge-Count Mode....1376
Table 17-4. Counter Values When the Timer is Enabled in Input Event-Count Mode....1377
Table 17-5. Counter Values When the Timer is Enabled in PWM Mode....1379
Table 17-6. Time-Out Actions for GPTM Modes....1382
Table 17-7. GPT Registers....1387
Table 17-8. GPT Access Type Codes....1388
Table 17-9. CFG Register Field Descriptions....1389
Table 17-10. TAMR Register Field Descriptions....1390
Table 17-11. TBMR Register Field Descriptions....1392
Table 17-12. CTL Register Field Descriptions....1394
Table 17-13. SYNC Register Field Descriptions....1396
Table 17-14. IMR Register Field Descriptions....1397
Table 17-15. RIS Register Field Descriptions....1398
Table 17-16. MIS Register Field Descriptions....1400
Table 17-17. ICLR Register Field Descriptions....1401
Table 17-18. TAILR Register Field Descriptions....1402
Table 17-19. TBILR Register Field Descriptions....1403
Table 17-20. TAMATCHR Register Field Descriptions....1404
Table 17-21. TBMATCHR Register Field Descriptions....1405
Table 17-22. TAPR Register Field Descriptions....1406
Table 17-23. TBPR Register Field Descriptions....1407
Table 17-24. TAPMR Register Field Descriptions....1408
Table 17-25. TBPMR Register Field Descriptions....1409
Table 17-26. TAR Register Field Descriptions....1410
Table 17-27. TBR Register Field Descriptions....1411
Table 17-28. TAV Register Field Descriptions....1412
Table 17-29. TBV Register Field Descriptions....1413
Table 17-30. TAPS Register Field Descriptions....1414
Table 17-31. TBPS Register Field Descriptions....1415
Table 17-32. TAPV Register Field Descriptions....1416
Table 17-33. TBPV Register Field Descriptions....1417
Table 17-34. DMAEV Register Field Descriptions....1418
Table 17-35. VERSION Register Field Descriptions....1419
Table 17-36. ANDCCP Register Field Descriptions....1420
Table 18-1. AON_RTC Registers....1426
Table 18-2. AON_RTC Access Type Codes....1426
Table 18-3. CTL Register Field Descriptions....1427

Table 18-4. EVFLAGS Register Field Descriptions ... 1428
Table 18-5. SEC Register Field Descriptions ... 1429
Table 18-6. SUBSEC Register Field Descriptions ... 1430
Table 18-7. SUBSECINC Register Field Descriptions ... 1431
Table 18-8. CHCTL Register Field Descriptions ... 1432
Table 18-9. CH0CMP Register Field Descriptions ... 1433
Table 18-10. CH1CMP Register Field Descriptions ... 1434
Table 18-11. CH2CMP Register Field Descriptions ... 1435
Table 18-12. CH2CMPINC Register Field Descriptions ... 1436
Table 18-13. CH1CAPT Register Field Descriptions ... 1437
Table 18-14. SYNC Register Field Descriptions ... 1438
Table 18-15. TIME Register Field Descriptions ... 1439
Table 18-16. SYNCLF Register Field Descriptions ... 1440
Table 19-1. WDT Registers ... 1444
Table 19-2. WDT Access Type Codes ... 1444
Table 19-3. LOAD Register Field Descriptions ... 1445
Table 19-4. VALUE Register Field Descriptions ... 1446
Table 19-5. CTL Register Field Descriptions ... 1447
Table 19-6. ICR Register Field Descriptions ... 1448
Table 19-7. RIS Register Field Descriptions ... 1449
Table 19-8. MIS Register Field Descriptions ... 1450
Table 19-9. TEST Register Field Descriptions ... 1451
Table 19-10. INT_CAUS Register Field Descriptions ... 1452
Table 19-11. LOCK Register Field Descriptions ... 1453
Table 20-1. AUX Operational Modes ... 1459
Table 20-2. AUX Operational Clock Rates ... 1460
Table 20-3. Task Testing and Task Debugging ... 1464
Table 20-4. Memory Word Access ... 1468
Table 20-5. I/O Word Access ... 1468
Table 20-6. I/O Bit Access ... 1468
Table 20-7. Register Access ... 1468
Table 20-8. Logical Operations ... 1468
Table 20-9. Arithmetic Operations ... 1470
Table 20-10. Shift Operations ... 1470
Table 20-11. Program Flow Control ... 1470
Table 20-12. Program Flow Conditions ... 1471
Table 20-13. Loop Flow Control ... 1471
Table 20-14. Power Management ... 1471
Table 20-15. Miscellaneous ... 1471
Table 20-16. SCE Event Interface ... 1472
Table 20-17. Wake-Up Event Interface ... 1476
Table 20-18. Wake-Up Vector Priority ... 1476
Table 20-19. Digital Peripherals ... 1479
Table 20-20. AUX I/O Control ... 1480
Table 20-21. Semaphore Resource Allocation ... 1482
Table 20-22. TDC Counter Clock Source ... 1489
Table 20-23. TDC Reference Clock Source ... 1489
Table 20-24. Phase Width Timing Requirements ... 1491
Table 20-25. Timing Requirements for Frequency Measurements ... 1491
Table 20-26. Arbitrary Time Measurement 1 ... 1492
Table 20-27. Arbitrary Time Measurement 2 ... 1493
Table 20-28. Arbitrary Time Measurement 3 ... 1494
Table 20-29. Arbitrary Time Measurement 4 ... 1495
Table 20-30. Channel Actions ... 1500
Table 20-31. Analog Peripherals ... 1505
Table 20-32. ADC Clock Source ... 1509
Table 20-33. AUX Event Bus ... 1522
Table 20-34. AUX Event Subscribers ... 1523
Table 20-35. Asynchronous Event Detection ... 1523
Table 20-36. Events to MCU ... 1525
Table 20-37. Events to AON ... 1526

Table 20-38. Alias Register Mapping....1527
Table 20-39. AUX Domain Sensor Controller and Peripherals Registers....1532
Table 20-40. ADI_4_AUX Registers....1533
Table 20-41. ADI_4_AUX Access Type Codes....1533
Table 20-42. MUX0 Register Field Descriptions....1534
Table 20-43. MUX1 Register Field Descriptions....1535
Table 20-44. MUX2 Register Field Descriptions....1536
Table 20-45. MUX3 Register Field Descriptions....1537
Table 20-46. ISRC Register Field Descriptions....1538
Table 20-47. COMP Register Field Descriptions....1539
Table 20-48. MUX4 Register Field Descriptions....1540
Table 20-49. ADC0 Register Field Descriptions....1541
Table 20-50. ADC1 Register Field Descriptions....1542
Table 20-51. ADCREF0 Register Field Descriptions....1543
Table 20-52. ADCREF1 Register Field Descriptions....1544
Table 20-53. LPMBIAS Register Field Descriptions....1545
Table 20-54. AUX_AIODIO Registers....1546
Table 20-55. AUX_AIODIO Access Type Codes....1546
Table 20-56. IOMODE Register Field Descriptions....1547
Table 20-57. GPIODIE Register Field Descriptions....1552
Table 20-58. IOPOE Register Field Descriptions....1553
Table 20-59. GPIODOUT Register Field Descriptions....1554
Table 20-60. GPIODIN Register Field Descriptions....1555
Table 20-61. GPIODOUTSET Register Field Descriptions....1556
Table 20-62. GPIODOUTCLR Register Field Descriptions....1557
Table 20-63. GPIODOUTTGL Register Field Descriptions....1558
Table 20-64. IO0PSEL Register Field Descriptions....1559
Table 20-65. IO1PSEL Register Field Descriptions....1560
Table 20-66. IO2PSEL Register Field Descriptions....1561
Table 20-67. IO3PSEL Register Field Descriptions....1562
Table 20-68. IO4PSEL Register Field Descriptions....1563
Table 20-69. IO5PSEL Register Field Descriptions....1564
Table 20-70. IO6PSEL Register Field Descriptions....1565
Table 20-71. IO7PSEL Register Field Descriptions....1566
Table 20-72. IOMODEL Register Field Descriptions....1567
Table 20-73. IOMODEH Register Field Descriptions....1568
Table 20-74. AUX_EVCTL Registers....1569
Table 20-75. AUX_EVCTL Access Type Codes....1569
Table 20-76. EVSTAT0 Register Field Descriptions....1570
Table 20-77. EVSTAT1 Register Field Descriptions....1571
Table 20-78. EVSTAT2 Register Field Descriptions....1572
Table 20-79. EVSTAT3 Register Field Descriptions....1573
Table 20-80. SCEWEVCFG0 Register Field Descriptions....1575
Table 20-81. SCEWEVCFG1 Register Field Descriptions....1578
Table 20-82. DMACTL Register Field Descriptions....1581
Table 20-83. SWEVSET Register Field Descriptions....1582
Table 20-84. EVTOAONFLAGS Register Field Descriptions....1583
Table 20-85. EVTOAONPOL Register Field Descriptions....1584
Table 20-86. EVTOAONFLAGSCLR Register Field Descriptions....1585
Table 20-87. EVTOMCUFLAGS Register Field Descriptions....1586
Table 20-88. EVTOMCUPOL Register Field Descriptions....1588
Table 20-89. EVTOMCUFLAGSCLR Register Field Descriptions....1590
Table 20-90. COMBEVTOMCUMASK Register Field Descriptions....1591
Table 20-91. EVOBSCFG Register Field Descriptions....1593
Table 20-92. PROGDLY Register Field Descriptions....1596
Table 20-93. MANUAL Register Field Descriptions....1597
Table 20-94. EVSTAT0L Register Field Descriptions....1598
Table 20-95. EVSTAT0H Register Field Descriptions....1599
Table 20-96. EVSTAT1L Register Field Descriptions....1600
Table 20-97. EVSTAT1H Register Field Descriptions....1601
Table 20-98. EVSTAT2L Register Field Descriptions....1602

Table 20-99. EVSTAT2H Register Field Descriptions....1603
Table 20-100. EVSTAT3L Register Field Descriptions....1604
Table 20-101. EVSTAT3H Register Field Descriptions....1605
Table 20-102. AUX_SMPH Registers....1606
Table 20-103. AUX_SMPH Access Type Codes....1606
Table 20-104. SMPH0 Register Field Descriptions....1607
Table 20-105. SMPH1 Register Field Descriptions....1608
Table 20-106. SMPH2 Register Field Descriptions....1609
Table 20-107. SMPH3 Register Field Descriptions....1610
Table 20-108. SMPH4 Register Field Descriptions....1611
Table 20-109. SMPH5 Register Field Descriptions....1612
Table 20-110. SMPH6 Register Field Descriptions....1613
Table 20-111. SMPH7 Register Field Descriptions....1614
Table 20-112. AUTOTAKE Register Field Descriptions....1615
Table 20-113. AUX_TDC Registers....1616
Table 20-114. AUX_TDC Access Type Codes....1616
Table 20-115. CTL Register Field Descriptions....1617
Table 20-116. STAT Register Field Descriptions....1618
Table 20-117. RESULT Register Field Descriptions....1619
Table 20-118. SATCFG Register Field Descriptions....1620
Table 20-119. TRIGSRC Register Field Descriptions....1621
Table 20-120. TRIGCNT Register Field Descriptions....1626
Table 20-121. TRIGCNTLOAD Register Field Descriptions....1627
Table 20-122. TRIGCNTCFG Register Field Descriptions....1628
Table 20-123. PRECTL Register Field Descriptions....1629
Table 20-124. PRECNTR Register Field Descriptions....1632
Table 20-125. AUX_TIMER01 Registers....1633
Table 20-126. AUX_TIMER01 Access Type Codes....1633
Table 20-127. T0CFG Register Field Descriptions....1634
Table 20-128. T0CTL Register Field Descriptions....1637
Table 20-129. T0TARGET Register Field Descriptions....1638
Table 20-130. T0CNTR Register Field Descriptions....1639
Table 20-131. T1CFG Register Field Descriptions....1640
Table 20-132. T1CTL Register Field Descriptions....1643
Table 20-133. T1TARGET Register Field Descriptions....1644
Table 20-134. T1CNTR Register Field Descriptions....1645
Table 20-135. AUX_TIMER2 Registers....1646
Table 20-136. AUX_TIMER2 Access Type Codes....1646
Table 20-137. CTL Register Field Descriptions....1647
Table 20-138. TARGET Register Field Descriptions....1648
Table 20-139. SHDWTARGET Register Field Descriptions....1649
Table 20-140. CNTR Register Field Descriptions....1650
Table 20-141. PRECFG Register Field Descriptions....1651
Table 20-142. EVCTL Register Field Descriptions....1652
Table 20-143. PULSETRIG Register Field Descriptions....1653
Table 20-144. CH0EVCFG Register Field Descriptions....1654
Table 20-145. CH0CCFG Register Field Descriptions....1657
Table 20-146. CH0PCC Register Field Descriptions....1660
Table 20-147. CH0CC Register Field Descriptions....1661
Table 20-148. CH1EVCFG Register Field Descriptions....1662
Table 20-149. CH1CCFG Register Field Descriptions....1665
Table 20-150. CH1PCC Register Field Descriptions....1668
Table 20-151. CH1CC Register Field Descriptions....1669
Table 20-152. CH2EVCFG Register Field Descriptions....1670
Table 20-153. CH2CCFG Register Field Descriptions....1673
Table 20-154. CH2PCC Register Field Descriptions....1676
Table 20-155. CH2CC Register Field Descriptions....1677
Table 20-156. CH3EVCFG Register Field Descriptions....1678
Table 20-157. CH3CCFG Register Field Descriptions....1681
Table 20-158. CH3PCC Register Field Descriptions....1684
Table 20-159. CH3CC Register Field Descriptions....1685

Table 20-160. AUX_ANAIF Registers....1686
Table 20-161. AUX_ANAIF Access Type Codes....1686
Table 20-162. ADCCTL Register Field Descriptions....1687
Table 20-163. ADCFIFOSTAT Register Field Descriptions....1690
Table 20-164. ADCFIFO Register Field Descriptions....1691
Table 20-165. ADCTRIG Register Field Descriptions....1692
Table 20-166. ISRCCTL Register Field Descriptions....1693
Table 20-167. DACCTL Register Field Descriptions....1694
Table 20-168. LPMBIASCTL Register Field Descriptions....1695
Table 20-169. DACSMPLCTL Register Field Descriptions....1696
Table 20-170. DACSMPLCFG0 Register Field Descriptions....1697
Table 20-171. DACSMPLCFG1 Register Field Descriptions....1698
Table 20-172. DACVALUE Register Field Descriptions....1699
Table 20-173. DACSTAT Register Field Descriptions....1700
Table 20-174. AUX_SYSIF Registers....1701
Table 20-175. AUX_SYSIF Access Type Codes....1702
Table 20-176. OPMODEREQ Register Field Descriptions....1703
Table 20-177. OPMODEACK Register Field Descriptions....1704
Table 20-178. PROGWU0CFG Register Field Descriptions....1705
Table 20-179. PROGWU1CFG Register Field Descriptions....1708
Table 20-180. PROGWU2CFG Register Field Descriptions....1711
Table 20-181. PROGWU3CFG Register Field Descriptions....1714
Table 20-182. SWWUTRIG Register Field Descriptions....1717
Table 20-183. WUFLAGS Register Field Descriptions....1718
Table 20-184. WUFLAGSCLR Register Field Descriptions....1719
Table 20-185. WUGATE Register Field Descriptions....1721
Table 20-186. VECCFG0 Register Field Descriptions....1722
Table 20-187. VECCFG1 Register Field Descriptions....1723
Table 20-188. VECCFG2 Register Field Descriptions....1724
Table 20-189. VECCFG3 Register Field Descriptions....1725
Table 20-190. VECCFG4 Register Field Descriptions....1726
Table 20-191. VECCFG5 Register Field Descriptions....1727
Table 20-192. VECCFG6 Register Field Descriptions....1728
Table 20-193. VECCFG7 Register Field Descriptions....1729
Table 20-194. EVSYNCRATE Register Field Descriptions....1730
Table 20-195. PEROPRATE Register Field Descriptions....1731
Table 20-196. ADCCLKCTL Register Field Descriptions....1732
Table 20-197. TDCCLKCTL Register Field Descriptions....1733
Table 20-198. TDCREFCLKCTL Register Field Descriptions....1734
Table 20-199. TIMER2CLKCTL Register Field Descriptions....1735
Table 20-200. TIMER2CLKSTAT Register Field Descriptions....1736
Table 20-201. TIMER2CLKSWITCH Register Field Descriptions....1737
Table 20-202. TIMER2DBGCTL Register Field Descriptions....1738
Table 20-203. CLKSHIFTDET Register Field Descriptions....1739
Table 20-204. RECHARGETRIG Register Field Descriptions....1740
Table 20-205. RECHARGEDET Register Field Descriptions....1741
Table 20-206. RTCSUBSECINC0 Register Field Descriptions....1742
Table 20-207. RTCSUBSECINC1 Register Field Descriptions....1743
Table 20-208. RTCSUBSECINCCTL Register Field Descriptions....1744
Table 20-209. RTCSEC Register Field Descriptions....1745
Table 20-210. RTCSUBSEC Register Field Descriptions....1746
Table 20-211. RTCEVCLR Register Field Descriptions....1747
Table 20-212. BATMONBAT Register Field Descriptions....1748
Table 20-213. BATMONTEMP Register Field Descriptions....1749
Table 20-214. TIMERHALT Register Field Descriptions....1750
Table 20-215. TIMER2BRIDGE Register Field Descriptions....1751
Table 20-216. SWPWRPROF Register Field Descriptions....1752
Table 20-217. AUX_SPIM Registers....1753
Table 20-218. AUX_SPIM Access Type Codes....1753
Table 20-219. SPIMCFG Register Field Descriptions....1754
Table 20-220. MISOCFG Register Field Descriptions....1755

Table 20-221. MOSICTL Register Field Descriptions....1756
Table 20-222. TX8 Register Field Descriptions....1757
Table 20-223. TX16 Register Field Descriptions....1758
Table 20-224. RX8 Register Field Descriptions....1759
Table 20-225. RX16 Register Field Descriptions....1760
Table 20-226. SCLKIDLE Register Field Descriptions....1761
Table 20-227. DATAIDLE Register Field Descriptions....1762
Table 20-228. AUX_MAC Registers....1763
Table 20-229. AUX_MAC Access Type Codes....1764
Table 20-230. OP0S Register Field Descriptions....1765
Table 20-231. OP0U Register Field Descriptions....1766
Table 20-232. OP1SMUL Register Field Descriptions....1767
Table 20-233. OP1UMUL Register Field Descriptions....1768
Table 20-234. OP1SMAC Register Field Descriptions....1769
Table 20-235. OP1UMAC Register Field Descriptions....1770
Table 20-236. OP1SADD16 Register Field Descriptions....1771
Table 20-237. OP1UADD16 Register Field Descriptions....1772
Table 20-238. OP1SADD32 Register Field Descriptions....1773
Table 20-239. OP1UADD32 Register Field Descriptions....1774
Table 20-240. CLZ Register Field Descriptions....1775
Table 20-241. CLS Register Field Descriptions....1776
Table 20-242. ACCSHIFT Register Field Descriptions....1777
Table 20-243. ACCRESET Register Field Descriptions....1778
Table 20-244. ACC15_0 Register Field Descriptions....1779
Table 20-245. ACC16_1 Register Field Descriptions....1780
Table 20-246. ACC17_2 Register Field Descriptions....1781
Table 20-247. ACC18_3 Register Field Descriptions....1782
Table 20-248. ACC19_4 Register Field Descriptions....1783
Table 20-249. ACC20_5 Register Field Descriptions....1784
Table 20-250. ACC21_6 Register Field Descriptions....1785
Table 20-251. ACC22_7 Register Field Descriptions....1786
Table 20-252. ACC23_8 Register Field Descriptions....1787
Table 20-253. ACC24_9 Register Field Descriptions....1788
Table 20-254. ACC25_10 Register Field Descriptions....1789
Table 20-255. ACC26_11 Register Field Descriptions....1790
Table 20-256. ACC27_12 Register Field Descriptions....1791
Table 20-257. ACC28_13 Register Field Descriptions....1792
Table 20-258. ACC29_14 Register Field Descriptions....1793
Table 20-259. ACC30_15 Register Field Descriptions....1794
Table 20-260. ACC31_16 Register Field Descriptions....1795
Table 20-261. ACC32_17 Register Field Descriptions....1796
Table 20-262. ACC33_18 Register Field Descriptions....1797
Table 20-263. ACC34_19 Register Field Descriptions....1798
Table 20-264. ACC35_20 Register Field Descriptions....1799
Table 20-265. ACC36_21 Register Field Descriptions....1800
Table 20-266. ACC37_22 Register Field Descriptions....1801
Table 20-267. ACC38_23 Register Field Descriptions....1802
Table 20-268. ACC39_24 Register Field Descriptions....1803
Table 20-269. ACC39_32 Register Field Descriptions....1804
Table 20-270. AUX_SCE Registers....1805
Table 20-271. AUX_SCE Access Type Codes....1805
Table 20-272. CTL Register Field Descriptions....1806
Table 20-273. FETCHSTAT Register Field Descriptions....1807
Table 20-274. CPUSTAT Register Field Descriptions....1808
Table 20-275. WUSTAT Register Field Descriptions....1809
Table 20-276. REG1_0 Register Field Descriptions....1810
Table 20-277. REG3_2 Register Field Descriptions....1811
Table 20-278. REG5_4 Register Field Descriptions....1812
Table 20-279. REG7_6 Register Field Descriptions....1813
Table 20-280. LOOPADDR Register Field Descriptions....1814
Table 20-281. LOOPCNT Register Field Descriptions....1815

Table 20-282. NONSECDDIACC0 Register Field Descriptions....1816
Table 20-283. NONSECDDIACC1 Register Field Descriptions....1817
Table 20-284. NONSECDDIACC2 Register Field Descriptions....1818
Table 20-285. NONSECDDIACC3 Register Field Descriptions....1819
Table 21-1. AON_BATMON Registers....1823
Table 21-2. AON_BATMON Access Type Codes....1823
Table 21-3. CTL Register Field Descriptions....1824
Table 21-4. MEASCFG Register Field Descriptions....1825
Table 21-5. TEMPP0 Register Field Descriptions....1826
Table 21-6. TEMPP1 Register Field Descriptions....1827
Table 21-7. TEMPP2 Register Field Descriptions....1828
Table 21-8. BATMONP0 Register Field Descriptions....1829
Table 21-9. BATMONP1 Register Field Descriptions....1830
Table 21-10. IOSTRP0 Register Field Descriptions....1831
Table 21-11. FLASHPUMPP0 Register Field Descriptions....1832
Table 21-12. BAT Register Field Descriptions....1833
Table 21-13. BATUPD Register Field Descriptions....1834
Table 21-14. TEMP Register Field Descriptions....1835
Table 21-15. TEMPUPD Register Field Descriptions....1836
Table 21-16. EVENTMASK Register Field Descriptions....1837
Table 21-17. EVENT Register Field Descriptions....1838
Table 21-18. BATTUL Register Field Descriptions....1839
Table 21-19. BATTLL Register Field Descriptions....1840
Table 21-20. TEMPUL Register Field Descriptions....1841
Table 21-21. TEMPLL Register Field Descriptions....1842
Table 22-1. Signals for UART....1845
Table 22-2. Flow Control Mode....1847
Table 22-3. UART Registers....1852
Table 22-4. UART Access Type Codes....1852
Table 22-5. DR Register Field Descriptions....1853
Table 22-6. RSR Register Field Descriptions....1854
Table 22-7. ECR Register Field Descriptions....1855
Table 22-8. FR Register Field Descriptions....1856
Table 22-9. IBRD Register Field Descriptions....1857
Table 22-10. FBRD Register Field Descriptions....1858
Table 22-11. LCRH Register Field Descriptions....1859
Table 22-12. CTL Register Field Descriptions....1860
Table 22-13. IFLS Register Field Descriptions....1861
Table 22-14. IMSC Register Field Descriptions....1862
Table 22-15. RIS Register Field Descriptions....1863
Table 22-16. MIS Register Field Descriptions....1865
Table 22-17. ICR Register Field Descriptions....1866
Table 22-18. DMACTL Register Field Descriptions....1867
Table 23-1. SPI Signals....1872
Table 23-2. SPI Registers....1884
Table 23-3. SPI Access Type Codes....1885
Table 23-4. IIDX Register Field Descriptions....1886
Table 23-5. IMASK Register Field Descriptions....1887
Table 23-6. RIS Register Field Descriptions....1888
Table 23-7. MIS Register Field Descriptions....1889
Table 23-8. ISET Register Field Descriptions....1890
Table 23-9. ICLR Register Field Descriptions....1891
Table 23-10. EVT_MODE Register Field Descriptions....1892
Table 23-11. DESC Register Field Descriptions....1893
Table 23-12. CTL0 Register Field Descriptions....1894
Table 23-13. CTL1 Register Field Descriptions....1896
Table 23-14. CLKCTL Register Field Descriptions....1898
Table 23-15. IFLS Register Field Descriptions....1899
Table 23-16. STAT Register Field Descriptions....1900
Table 23-17. CLKDIV2 Register Field Descriptions....1901
Table 23-18. DMACR Register Field Descriptions....1902

Table 23-19. RXDATA Register Field Descriptions....1903
Table 23-20. TXDATA Register Field Descriptions....1904
Table 24-1. Examples of I2C Master Timer Period versus Speed Mode....1909
Table 24-2. I2C Registers....1919
Table 24-3. I2C Access Type Codes....1919
Table 24-4. SOAR Register Field Descriptions....1920
Table 24-5. SSTAT Register Field Descriptions....1921
Table 24-6. SCTL Register Field Descriptions....1922
Table 24-7. SDR Register Field Descriptions....1923
Table 24-8. SIMR Register Field Descriptions....1924
Table 24-9. SRIS Register Field Descriptions....1925
Table 24-10. SMIS Register Field Descriptions....1926
Table 24-11. SICR Register Field Descriptions....1927
Table 24-12. MSA Register Field Descriptions....1928
Table 24-13. MSTAT Register Field Descriptions....1929
Table 24-14. MCTRL Register Field Descriptions....1930
Table 24-15. MDR Register Field Descriptions....1931
Table 24-16. MTPR Register Field Descriptions....1932
Table 24-17. MIMR Register Field Descriptions....1933
Table 24-18. MRIS Register Field Descriptions....1934
Table 24-19. MMIS Register Field Descriptions....1935
Table 24-20. MICR Register Field Descriptions....1936
Table 24-21. MCR Register Field Descriptions....1937
Table 25-1. Serial Audio Pin Interface....1942
Table 25-2. I2S Registers....1955
Table 25-3. I2S Access Type Codes....1955
Table 25-4. AIFWCLKSRC Register Field Descriptions....1956
Table 25-5. AIFDMACFG Register Field Descriptions....1957
Table 25-6. AIFDIRCFG Register Field Descriptions....1958
Table 25-7. AIFFMTCFG Register Field Descriptions....1959
Table 25-8. AIFWMASK0 Register Field Descriptions....1960
Table 25-9. AIFWMASK1 Register Field Descriptions....1961
Table 25-10. AIFPWMVALUE Register Field Descriptions....1962
Table 25-11. AIFINPTRNEXT Register Field Descriptions....1963
Table 25-12. AIFINPTR Register Field Descriptions....1964
Table 25-13. AIFOUTPTRNEXT Register Field Descriptions....1965
Table 25-14. AIFOUTPTR Register Field Descriptions....1966
Table 25-15. STMPCTL Register Field Descriptions....1967
Table 25-16. STMPXCNTCAPT0 Register Field Descriptions....1968
Table 25-17. STMPXPER Register Field Descriptions....1969
Table 25-18. STMPWCNTCAPT0 Register Field Descriptions....1970
Table 25-19. STMPWPER Register Field Descriptions....1971
Table 25-20. STMPINTRIG Register Field Descriptions....1972
Table 25-21. STMPOUTTRIG Register Field Descriptions....1973
Table 25-22. STMPWSET Register Field Descriptions....1974
Table 25-23. STMPWADD Register Field Descriptions....1975
Table 25-24. STMPXPERMIN Register Field Descriptions....1976
Table 25-25. STMPWCNT Register Field Descriptions....1977
Table 25-26. STMPXCNT Register Field Descriptions....1978
Table 25-27. STMPXCNTCAPT1 Register Field Descriptions....1979
Table 25-28. STMPWCNTCAPT1 Register Field Descriptions....1980
Table 25-29. IRQMASK Register Field Descriptions....1981
Table 25-30. IRQFLAGS Register Field Descriptions....1982
Table 25-31. IRQSET Register Field Descriptions....1983
Table 25-32. IRQCLR Register Field Descriptions....1984
Table 26-1. Format of PRCM:RFCBITS for Enabling Selected Features as Part of Boot Process....1988
Table 26-2. Values of the Result Byte in the RFC_DBELL:CMDSTA Register....1993
Table 26-3. Common Radio Operation Status Codes....1994
Table 26-4. Format of Trigger Definition Byte....1997
Table 26-5. Supported Trigger Types....1997
Table 26-6. Fields of Time Parameter for External Event Trigger....1997

Table 26-7. Format of Condition Byte.... 1998
Table 26-8. Condition Rules.... 1999
Table 26-9. Radio Operation Command Format.... 2000
Table 26-10. Data Entry Queue Structure.... 2000
Table 26-11. General Data Entry Structure.... 2001
Table 26-12. Pointer Field in Pointer Entry Structure.... 2002
Table 26-13. Fields in a Partial Read RX Entry.... 2002
Table 26-14. End of Radio Operation Commands.... 2004
Table 26-15. CMD_RADIO_SETUP Command Format.... 2005
Table 26-16. Format of a Hardware Register Override Entry.... 2006
Table 26-17. Format of Array Initiator.... 2006
Table 26-18. Format of an ADI Register Override Entry.... 2007
Table 26-19. Format of a Firmware-Defined Parameter Override Entry.... 2007
Table 26-20. Format of an MCE/RFE Override Mode Entry.... 2007
Table 26-21. Format of a Center Frequency Entry.... 2008
Table 26-22. 20 dBm PA TX Power Entry.... 2008
Table 26-23. Format of an End of List Entry.... 2008
Table 26-24. CMD_FS_POWERUP Command Format.... 2009
Table 26-25. CMD_FS Command Format.... 2010
Table 26-26. CMD_RX_TEST Command Format.... 2011
Table 26-27. CMD_TX_TEST Command Format.... 2012
Table 26-28. CMD_SYNC_STOP_RAT Command Format.... 2013
Table 26-29. CMD_SYNC_START_RAT Command Format.... 2013
Table 26-30. CMD_COUNT Command Format.... 2014
Table 26-31. Additional End Causes for CMD_COUNT.... 2014
Table 26-32. CMD_SCH_IMM Command Format.... 2014
Table 26-33. End Statuses for CMD_SCH_IMM.... 2015
Table 26-34. CMD_COUNT_BRANCH Command Format.... 2015
Table 26-35. Additional End Causes for CMD_COUNT_BRANCH.... 2015
Table 26-36. CMD_PATTERN_CHECK Command Format.... 2016
Table 26-37. Additional End Causes for CMD_PATTERN_CHECK.... 2017
Table 26-38. CMD_UPDATE_RADIO_SETUP Command Format.... 2018
Table 26-39. CMD_TRIGGER Command Format.... 2018
Table 26-40. CMD_GET_FW_INFO Command Format.... 2019
Table 26-41. CMD_READ_RFREG Command Format.... 2020
Table 26-42. CMD_SET_RAT_CMP Command Format.... 2020
Table 26-43. CMD_SET_RAT_CPT Command Format.... 2021
Table 26-44. CMD_DISABLE_RAT_CH Command Format.... 2021
Table 26-45. CMD_SET_RAT_OUTPUT Command Format.... 2022
Table 26-46. CMD_ARM_RAT_CH Command Format.... 2022
Table 26-47. CMD_DISARM_RAT_CH Command Format.... 2023
Table 26-48. CMD_SET_TX_POWER Command Format.... 2023
Table 26-49. CMD_SET_TX20_POWER Command Format.... 2024
Table 26-50. CMD_MODIFY_FS Command Format.... 2025
Table 26-51. CMD_BUS_REQUEST Command Format.... 2025
Table 26-52. CMD_ADD_DATA_ENTRY Command Format.... 2026
Table 26-53. CMD_REMOVE_DATA_ENTRY Command Format.... 2026
Table 26-54. CMD_FLUSH_QUEUE Command Format.... 2027
Table 26-55. CMD_CLEAR_RX Command Format.... 2027
Table 26-56. CMD_REMOVE_PENDING_ENTRIES Command Format.... 2028
Table 26-57. IEEE 802.15.4 Radio Operation Commands on Background Level.... 2033
Table 26-58. IEEE 802.15.4 Radio Operation Commands on Foreground Level.... 2033
Table 26-59. IEEE 802.15.4 Immediate Commands.... 2033
Table 26-60. IEEE 802.15.4 RX Command Structure.... 2034
Table 26-61. IEEE 802.15.4 Energy Detect Scan Command Structure.... 2034
Table 26-62. IEEE 802.15.4 CSMA-CA Command Structure.... 2035
Table 26-63. IEEE 802.15.4 TX Command Structure.... 2035
Table 26-64. IEEE 802.15.4 Receive ACK Command Structure.... 2035
Table 26-65. IEEE 802.15.4 Modify CCA Immediate Command Structure.... 2036
Table 26-66. IEEE 802.15.4 Modify Frame Filtering Immediate Command Structure.... 2036
Table 26-67. IEEE 802.15.4 Enable or Disable Source Matching Entry Immediate Command Structure.... 2036

Table 26-68. IEEE 802.15.4 Request CCA State Immediate Command Structure 2037
Table 26-69. RX Command 2037
Table 26-70. Receive Queue Entry Configuration Bit Field 2038
Table 26-71. CCA Configuration Bit Field 2038
Table 26-72. Frame Filtering Configuration Bit Field 2039
Table 26-73. Frame Type Filtering Bit Field 2039
Table 26-74. Short Address Entry Structure 2040
Table 26-75. Extended Address List Structure 2040
Table 26-76. Short Address List Structure 2040
Table 26-77. Receive Correlation/CRC Result Bit Field 2040
Table 26-78. Interrupt Definitions Applicable to IEEE 802.15.4 2041
Table 26-79. Allowed Combinations of Foreground and Background Level Operations 2042
Table 26-80. IEEE 802.15.4 Radio Operation Status Codes 2043
Table 26-81. Conditions for Incrementing Counters and Raising Interrupts for RX Operation 2045
Table 26-82. End of Receive Operation 2047
Table 26-83. End of CSMA-CA Operation 2052
Table 26-84. End of Transmit Operation 2053
Table 26-85. End of Receive ACK Operation 2053
Table 26-86. End of ABORT Background-Level Operation 2054
Table 26-87. Legacy Bluetooth® Low Energy Radio Operation Commands 2056
Table 26-88. Bluetooth® Low Energy 5 Radio Operation Commands 2057
Table 26-89. Bluetooth® Low Energy Immediate Command 2057
Table 26-90. Legacy Bluetooth® Low Energy Radio Operation Command Structure(1) 2057
Table 26-91. Bluetooth® Low Energy 5 Radio Operation Command Structure(1) 2058
Table 26-92. Bluetooth® Low Energy 5 Radio Setup Command Structure(1) 2059
Table 26-93. Update Advertising Payload Command(1) 2060
Table 26-94. Legacy Slave Command(1) 2060
Table 26-95. Legacy Master Command(1) 2060
Table 26-96. Legacy Advertiser Commands(1) 2061
Table 26-97. Legacy Scanner Command(1) 2062
Table 26-98. Legacy Initiator Command(1) 2063
Table 26-99. Generic RX Command(1) 2064
Table 26-100. TX Test Command(1) 2064
Table 26-101. Bluetooth® 5 Slave Command(1) 2065
Table 26-102. Bluetooth® 5 Master Command(1) 2065
Table 26-103. Extended Advertiser Command(1) 2066
Table 26-104. Secondary Channel Advertiser Command(1) 2066
Table 26-105. Bluetooth® 5 Scanner Command(1) 2067
Table 26-106. Bluetooth® 5 Initiator Command(1) 2069
Table 26-107. Master and Slave Commands 2071
Table 26-108. Advertiser Commands 2071
Table 26-109. Legacy Scanner Command 2072
Table 26-110. Legacy Initiator Command 2072
Table 26-111. Bluetooth® Low Energy 5 Scanner and Initiator Command 2072
Table 26-112. Generic RX Command 2073
Table 26-113. Test TX Command 2073
Table 26-114. Receive Queue Entry Configuration Bit Field(1) 2074
Table 26-115. Sequence Number Status Bit Field(1) 2074
Table 26-116. Whitelist Structure 2074
Table 26-117. Advertising Data ID Entry Structure 2075
Table 26-118. Receive Status Byte Bit Field for Legacy Commands(1) 2075
Table 26-119. Receive Status Word Bit Field for Bluetooth® Low Energy 5 Commands(1) 2075
Table 26-120. Master and Slave Packet Status Byte 2075
Table 26-121. Common Extended Packet Entry Format(1) 2076
Table 26-122. Interrupt Definitions Applicable to Bluetooth® Low Energy 2076
Table 26-123. Bluetooth® Low Energy Radio Operation Status Codes 2080
Table 26-124. Coding Selection for Master and Slave Commands 2081
Table 26-125. Coding Selection for Advertiser, Scanner, and Initiator Commands 2081
Table 26-126. Actions on Received Packets 2082
Table 26-127. Conditions for Incrementing Counters and Raising Interrupts for Master and Slave Commands 2083
Table 26-128. End of Slave Operation 2086

Table 26-129. End of Master Operation....2087
Table 26-130. PDU Types for Different Advertiser Commands....2088
Table 26-131. Actions to Take Based on Received Packets for Advertisers....2089
Table 26-132. Descriptions of the Actions to Take on Received Packets....2089
Table 26-133. End of Connectable Undirected Advertiser Operation....2091
Table 26-134. End of Directed Advertiser Operation....2092
Table 26-135. End of Non-connectable Advertiser Operation....2092
Table 26-136. End of Scannable Undirected Advertiser Operation....2093
Table 26-137. End of Extended Advertiser Operation....2095
Table 26-138. Actions to Take Based on Received Packets for Scannable Advertiser (extHdrInfo.advMode = 2)(1) 2096
Table 26-139. Actions to Take Based on Received Packets for Connectable Advertiser (extHdrInfo.advMode = 1)(1) 2096
Table 26-140. Descriptions of the Actions to Take on Received Packets....2096
Table 26-141. End of Secondary Channel Advertiser Operation....2097
Table 26-142. Filtering on Received Advertiser Address....2098
Table 26-143. Filtering on Received Initiator Address of ADV_DIRECT_IND Packets....2098
Table 26-144. Actions on Received Packets by Scanner....2099
Table 26-145. Descriptions of the Actions to Take on Packets Received by Scanner....2099
Table 26-146. Actions on Packets Received by Scanner After Transmission of SCAN_REQ....2100
Table 26-147. Actions on Received Extended Advertiser Packets by Scanner....2101
Table 26-148. Descriptions of the Actions to Take on Extended Advertiser Packets Received by Scanner....2101
Table 26-149. Actions on Received Secondary Channel Packets by Scanner....2102
Table 26-150. Descriptions of the Actions to Take on Secondary Channel Packets Received by Scanner....2102
Table 26-151. End of Scanner Operation....2105
Table 26-152. Filtering on Received Advertiser Address....2106
Table 26-153. Actions on Received Packets by Initiator....2106
Table 26-154. Descriptions of the Actions to Take on Packets Received by Initiator....2106
Table 26-155. Actions on Received Extended Advertiser Packets by Initiator....2108
Table 26-156. Descriptions of the Actions to Take on Extended Advertiser Packets Received by Initiator....2108
Table 26-157. Actions on Received Secondary Channel Packets by Initiator....2109
Table 26-158. Descriptions of the Actions to Take on Secondary Channel Packets Received by Initiator....2109
Table 26-159. Actions on Received AUX_CONNECT_RSP Packets by Initiator....2109
Table 26-160. Descriptions of the Actions to Take on Secondary Channel Packets Received by Initiator....2110
Table 26-161. End of Initiator Operation....2111
Table 26-162. End of Generic RX Operation....2113
Table 26-163. Supported PHY Test Packet Types....2114
Table 26-164. End of PHY Test TX Operation....2114
Table 26-165. Update of Backoff Parameters....2115
Table 26-166. Proprietary Radio Operation Commands....2118
Table 26-167. Proprietary Immediate Commands....2118
Table 26-168. CMD_PROP_TX Command Structure....2119
Table 26-169. CMD_PROP_TX_ADV Command Structure....2119
Table 26-170. CMD_PROP_RX and CMD_PROP_RX_SNIFF Command Structure....2120
Table 26-171. CMD_PROP_RX_ADV and CMD_PROP_RX_ADV_SNIFF Command Structure....2121
Table 26-172. CMD_PROP_CS Command Structure....2121
Table 26-173. CMD_PROP_RADIO_SETUP and CMD_PROP_RADIO_DIV_SETUP Command Structure....2122
Table 26-174. CMD_PROP_SET_LEN Command Structure....2123
Table 26-175. Receive Commands....2123
Table 26-176. Carrier Sense Fields for CMD_PROP_RX_SNIFF, CMD_PROP_RX_ADV_SNIFF, and CMD_PROP_CS.. 2124
Table 26-177. Receive Queue Entry Configuration Bit Field(1) 2124
Table 26-178. Receive Status Byte Bit Field (1) 2124
Table 26-179. Interrupt Definitions....2125
Table 26-180. Proprietary Radio Operation Status Codes....2127
Table 26-181. Receiver Bandwidth Settings....2128
Table 26-182. End of Radio CMD_PROP_TX and CMD_PROP_TX_ADV Commands....2131
Table 26-183. Interrupt, Counter, and Result Field for Received Packets (1) 2132
Table 26-184. End of Radio CMD_PROP_RX and CMD_PROP_RX_ADV Commands....2133
Table 26-185. Channel State when both Sources are Enabled....2137
Table 26-186. End of CMD_PROP_CS Command....2137
Table 26-187. Additional End Statuses for CMD_PROP_RX_SNIFF and CMD_PROP_RX_ADV_SNIFF....2138
Table 26-188. RFC_RAT Registers....2140
Table 26-189. RFC_RAT Access Type Codes....2140

Table 26-190. RATCNT Register Field Descriptions....2141
Table 26-191. RATCH0VAL Register Field Descriptions....2142
Table 26-192. RATCH1VAL Register Field Descriptions....2143
Table 26-193. RATCH2VAL Register Field Descriptions....2144
Table 26-194. RATCH3VAL Register Field Descriptions....2145
Table 26-195. RATCH4VAL Register Field Descriptions....2146
Table 26-196. RATCH5VAL Register Field Descriptions....2147
Table 26-197. RATCH6VAL Register Field Descriptions....2148
Table 26-198. RATCH7VAL Register Field Descriptions....2149
Table 26-199. RFC_DBELL Registers....2150
Table 26-200. RFC_DBELL Access Type Codes....2150
Table 26-201. CMDR Register Field Descriptions....2151
Table 26-202. CMDSTA Register Field Descriptions....2152
Table 26-203. RFHWIFG Register Field Descriptions....2153
Table 26-204. RFHWIEN Register Field Descriptions....2154
Table 26-205. RFCPEIFG Register Field Descriptions....2155
Table 26-206. RFCPEIEN Register Field Descriptions....2158
Table 26-207. RFCPEISL Register Field Descriptions....2159
Table 26-208. RFACKIFG Register Field Descriptions....2163
Table 26-209. SYSGPOCTL Register Field Descriptions....2164
Table 26-210. RFC_PWR Registers....2166
Table 26-211. RFC_PWR Access Type Codes....2166
Table 26-212. PWMCLKEN Register Field Descriptions....2167

This technical reference manual provides information about how to use the CC13x4x10 and CC26x4x10 SimpleLink™ ultra-low power wireless microcontroller (MCU). The CC13x4x10 and the CC26x4x10 device platform share the same MCU architecture and most of the peripherals and features. The radio in the CC2674x10 device operates in the 2.4 GHz ISM frequency band, while the radio in the CC1314R10 device is designed for use in the Sub-1 GHz frequency bands. The CC1354x10 device is a multi-band wireless MCU and can operate both in the Sub-1 GHz and 2.4 GHz bands. This document covers the whole platform of devices, so refer to the individual device data sheets for supported modules and features.

About This Manual

This document is organized into sections that correspond to each major feature; it explains the features and functionality of each module and how to use them. For each feature, references are given to the driver documentation of the corresponding operating systems. Driver documentation does not contain performance characteristics for the device or modules, which are instead in the corresponding device data sheets. This manual is intended for system software developers, hardware designers, and application developers.

Devices

The CC13x4x10 and CC26x4x10 device platform includes both 2.4 GHz and Sub-1 GHz radios along with a variety of different memory sizes, peripherals, and package options. All devices are centered around an Arm® Cortex®-M33 with FPU series processor that handles the application layer and protocol stack, as well as an autonomous radio core centered around an Arm® Cortex®-M0 processor that handles all the low-level radio control and processing. Network processor options are available.

The availability of a wide range of different radio and MCU system combinations makes these device families very well suited for almost any low-power RF node implementation.

Register, Field, and Bit Calls

The naming convention applied for a call consists of:

- For a register call:
  - <Module name>:<Register name>
    - For example: UART:LCRH
- For a bit field call:
  - <Module name>:<Register name>.<Field name>
    - For example, UART:LCRH.FEN
  - <Field name> field of the <Module name>:<Register name> register
    - For example: FEN field of the UART:LCRH register

Related Documentation

The following related documents are available on the CC13x4x10 and CC26x4x10 device product pages at www.ti.com:

- CC1314R10:
  - CC1314R10 data sheet and errata (Technical Documents)
- CC1354x10:
  - CC1354R10 data sheet and errata (Technical Documents)
  - CC1354P10 data sheet and errata (Technical Documents)
- CC2674x10:
  - CC2674R10 data sheet and errata (Technical Documents)
  - CC2674P10 data sheet and errata (Technical Documents)

Note

This list of documents was current as of publication date. Check the website for additional documentation, application notes, and white papers.

Additional, related documentation follows:

- The Institute of Electrical and Electronic Engineers, Inc., IEEE Standard Test Access Port and Boundary Scan Architecture, IEEE Std 1149.1a 1993 and Supplement Std. 1149.1b 1994 (see IEEExplore.ieee.org)
- The Institute of Electrical and Electronic Engineers, Inc., IEEE 1149.7 Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture (see IEEExplore.ieee.org)
- National Institute of Standards and Technology, NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation Methods and Techniques (see NIST.gov)
- National Institute of Standards and Technology, NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (see NIST.gov)
- National Institute of Standards and Technology, FIPS 197, Advanced Encryption Standard (AES) (see NIST.gov)
- Bluetooth SIG, Inc., Bluetooth Specification versions 4.0, 4.1, 4.2, and 5.0 (see Bluetooth.com)
- Arm® Cortex® -M33 Processor Technical Reference Manual
- Arm® Cortex®-M33 Devices Generic User Guide
- Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2
- TrustZone® technology for Armv8-M Architecture

Trademarks

SimpleLink™ is a trademark of Texas Instruments.
CoreSight™ is a trademark of Arm Limited (or its subsidiaries).
Motorola™ is a trademark of Motorola Trademark Holdings, LLC.
Wi-SUN™ is a trademark of Wi-SUN Alliance, Inc..
Arm®, Cortex®, Thumb®, and PrimeCell® are registered trademarks of Arm Limited (or its subsidiaries).
TrustZone® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
Zigbee® is a registered trademark of Zigbee Alliance.
Bluetooth® is a registered trademark of Bluetooth Special Interest Group (SIG) .
All trademarks are the property of their respective owners.

This page intentionally left blank.

Chapter 1

Architectural Overview

The CC13x4x10 and CC26x4x10 device platform of the SimpleLink™ ultra-low-power wireless MCUs provides solutions for a wide range of applications. To help the user develop these applications, this user's guide focuses on the use of the different building blocks of the devices. For detailed device descriptions, complete feature lists, and performance numbers, see the data sheet for the specific device. The following subsections provide easy access to relevant information and guide the reader to the different chapters in this document.

The device platform system-on-chips (SoCs) are optimized for ultra-low power, while providing fast and capable MCU systems to enable short processing times and high integration. The combination of an Arm® Cortex®-M33 with FPU processing core running at 48 MHz, Flash memory, and a wide selection of peripherals makes the CC13x4x10 and CC26x4x10 device platform specifically designed for single-chip implementation or network processor implementations of lower-power RF nodes.

1.1 Target Applications 42
1.2 Overview 42
1.3 Functional Overview 45

1.1 Target Applications

The CC13x4x10 and CC26x4x10 SimpleLink™ ultra-low-power wireless MCU platform is positioned for low-power wireless applications, such as:

- Consumer electronics
- Mobile phone accessories
- Sports and fitness equipment
- HID applications
- Home and building automation
- Lighting control
- Alarm and security
- Electronic shelf labeling
- Proximity tags
- Medical
- Remote controls
- Smart metering
- Asset tracking
- Wireless sensor networks

1.2 Overview

Figure 1-1 shows the building blocks of the CC13x4x10 and CC26x4x10 device platform.

The CC13x4x10 and CC26x4x10 device platform has the following features:

- Arm® Cortex®-M33 Processor with DSP extensions, FPU and TrustZone®
  - 48-MHz RC oscillator and 48-MHz crystal oscillator
  - 32-kHz crystal oscillator, 32-kHz RC oscillator, or low-power 48-MHz crystal derive clock for timing maintenance while in low-power modes
  - Arm® Cortex® SysTick timer
  - Nested vectored interrupt controller (NVIC)
- On-chip memory
  - 1 MB of flash in two banks supporting read-while-write with 8 KB of 4-way set-associative cache RAM for speed and low power
  - Up to 256 KB of System RAM with parity, or 288 KB without
- Power management
  - Wide supply voltage range
  - Efficient on-chip DC/DC converter for reduced power consumption
  - High granularity clock gating and power gating of device parts
  - Flexible frequency of operation
    - Flexible low-power modes allowing low energy consumption in duty cycled applications
- Sensor interface
  - Autonomous, intelligent sensor interface that can wake up independently of the System CPU system to perform sensor readings, collect data, and determine if the System CPU must be woken
  - 12-bit analog-to-digital converter (ADC) with eight analog input channels
  - Low-power analog comparator
  - Serial communication interfaces

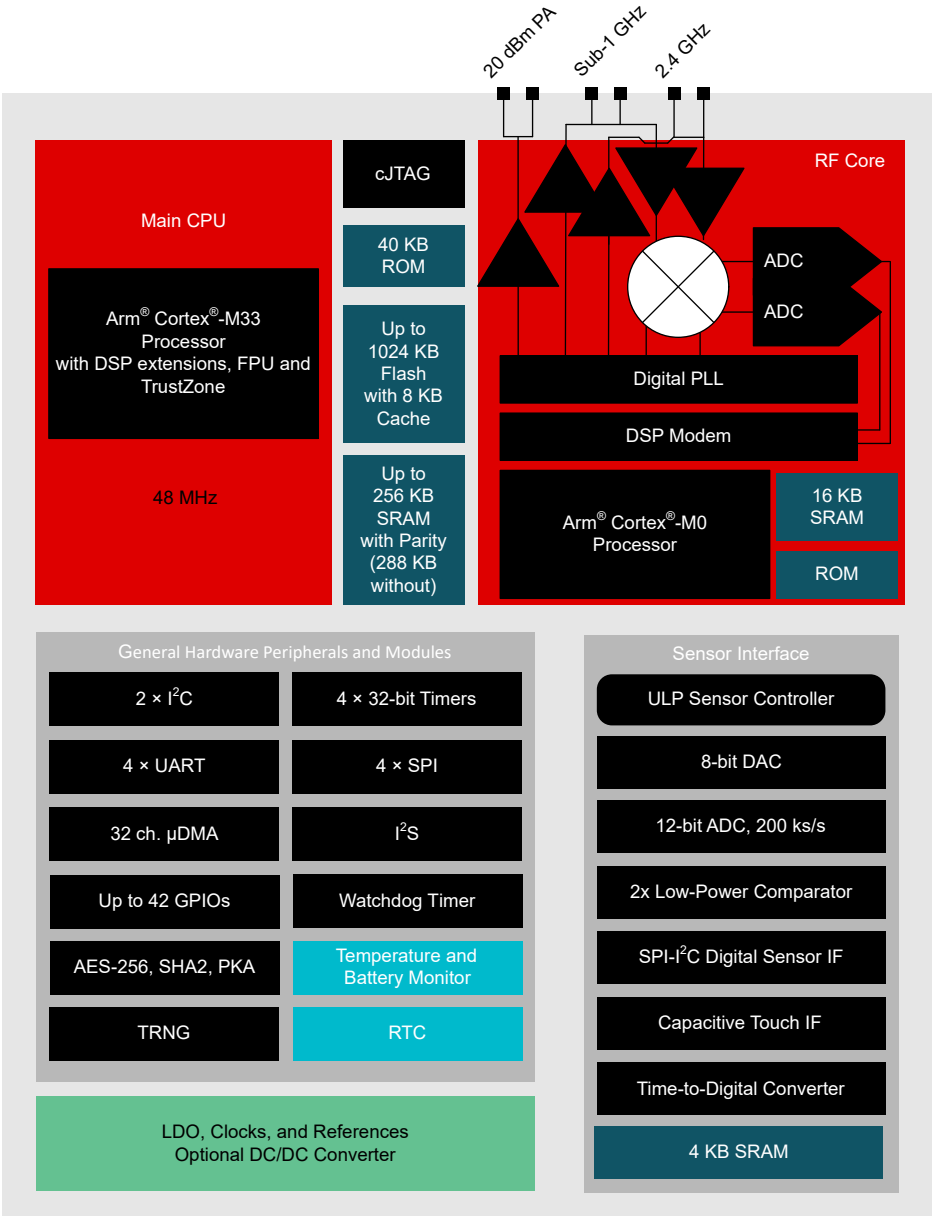


Figure 1-1. CC13x4x10 and CC26x4x10 Block Diagram

- Advanced serial integration
  - Universal Asynchronous Receivers/Transmitters (UARTs)
  - Inter-Integrated Circuit (I^2C) modules
  - Serial Peripheral Interface modules (SPIs)
  - Inter-IC Sound module (I^2S)
- System integration
  - Direct Memory Access (µDMA) controller
  - Four 32-bit timers (up to eight 16-bit) with pulse width modulation (PWM) capability and synchronization
  - 32 kHz Real-Time Clock (RTC)
  - Watchdog timer
  - On-chip temperature and supply voltage sensing
  - GPIO with normal or high-drive capabilities
  - GPIOs with analog capability for ADC and comparator
  - Fully flexible digital pin muxing allows pins to be used as GPIO or any peripheral function

- IEEE 1149.7 compliant 2-pin cJTAG with legacy 1149.1 JTAG support
- 7-mm × 7-mm VQFN and 8-mm x 8-mm VQFN package

For applications requiring extreme conservation of power, the CC13x4x10 and CC26x4x10 device platform features a power-management system to efficiently power down the devices to a low-power state during extended periods of inactivity. A power-up and power-down sequencer, a 32-bit sleep timer (an RTC) with interrupt, and ultra-low-leakage (ULL) RAM with retention in all power modes positions the MCU perfectly for battery applications.

In addition, the CC13x4x10 and CC26x4x10 device platform offers the advantages of the widely available development tools of Arm®, SoC infrastructure IP applications, and a large user community. Additionally, the microcontroller uses Arm® Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost.

TI offers a complete development environment to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network.

1.3 Functional Overview

The following subsections provide an overview of the features of the CC13x4x10 and CC26x4x10 device platform.

1.3.1 Arm®Cortex-M33 with FPU

The following subsections provide an overview of the Arm®Cortex-M33 processor core and instruction set, the integrated system timer (SysTick), and the NVIC.

1.3.1.1 Processor Core

The CC13x4x10 and CC26x4x10 device platform contains an Arm® Cortex®-M33 system CPU with TrustZone®, which runs the application and the higher layers of radio protocol stacks. The system CPU is the foundation of a high-performance, low-cost platform that meets the system requirements of minimal memory implementation, and low-power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

Features include the following:

- Armv8-M architecture with TrustZone® security extension optimized for small-footprint embedded applications
- Arm® Thumb®-2 mixed 16- and 32-bit instruction set delivers the high performance expected of a 32-bit Arm® core in a compact memory size
- 8 regions of Non-secure memory protected regions
- 8 regions of Secure memory protected regions
- 4 regions of Security Attribute Unit (SAU)
- Single-cycle multiply instruction and hardware divide
- Digital-signal-processing (DSP) extension
- IEEE 754-compliant single-precision Floating Point Unit (FPU)
- Fast code execution permits increased sleep mode time
- Deterministic, high-performance interrupt handling for time-critical applications
- Full debug with data matching for watchpoint generation
  - Data Watchpoint and Trace Unit (DWT)
  - JTAG Debug Access Port (DAP)
  - Flash Patch and Breakpoint Unit (FPB)
- Trace support reduces the number of pins required for debugging and tracing
  - Instrumentation Trace Macrocell Unit (ITM)
  - Trace Port Interface Unit (TPIU) with asynchronous serial wire output (SWO)
- Optimized for single-cycle Flash memory access
- Tightly connected to 8-KB 4-way random replacement cache for minimal active power consumption and wait states
- Ultra-low-power consumption with integrated sleep modes
- 48-MHz operation

1.3.1.2 System Timer (SysTick)

The Arm® Cortex®-M33 processor includes an integrated system timer (SysTick). SysTick provides a simple, 24-bit, clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways; for example:

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine
- A high-speed alarm timer using system clock
- A variable rate alarm or signal timer where the duration is range-dependent on the reference clock used and the dynamic range of the counter
- A simple counter used to measure time to completion and time used
- An internal clock-source control based on missing or meeting durations

1.3.1.3 Nested Vector Interrupt Controller (NVIC)

The CC13x4x10 and CC26x4x10 device controller includes the Arm® NVIC. The NVIC and Arm® Cortex®-M33 prioritize and handle all exceptions in handler mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the interrupt service routine (ISR). The interrupt vector is fetched in parallel to state saving, thus enabling efficient interrupt entry. The processor supports tail-chaining, that is, back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on seven exceptions (system handlers) and can set device interrupts.

Features of the NVIC are as follows:

- Deterministic, fast interrupt processing
  - Always 12 cycles, or just 6 cycles with tail-chaining
- External non-maskable interrupt (NMI) signal available for immediate execution of NMI handler for safety-critical applications
- Dynamically reprioritizable interrupts
- Exceptional interrupt handling through hardware implementation of required register manipulations

1.3.1.4 System Control Block (SCB)

The System Control Block (SCB) provides system implementation information and system control (configuration, control, and reporting of system exceptions).

1.3.2 On-Chip Memory

The following subsections describe the on-chip memory modules.

1.3.2.1 SRAM

The CC13x4x10 and CC26x4x10 device platform provides 256 KB with parity, or 288 KB without, of low-leakage, on-chip SRAM with optional retention in all power modes. Additionally, the 8 KB Flash cache RAM can be reconfigured to operate as normal system RAM.

Data can be transferred to and from the SRAM using the micro DMA (µDMA) controller.

1.3.2.2 Flash Memory

The Flash block provides an in-circuit, programmable, nonvolatile program memory for the device. The 1 MB Flash memory is organized as a set of 2 KB sectors that can be individually erased. Read-while-write is supported, meaning that half of the Flash is readable while writing or erasing the other half. Erasing a block causes the entire contents of the block to be reset to all 1s. These pages can be individually protected. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. In addition to holding program code and constants, the nonvolatile memory allows the application to save data that must be preserved so that it is available after restarting the device. Using this feature lets the user use saved network-specific data to avoid the need for a full start-up and network find-and-join process.

1.3.2.3 ROM

The ROM is preprogrammed with a boot sequence and a serial bootloader (SPI or UART).

1.3.3 Radio

The CC2674x10 devices provide a highly integrated low-power 2.4 GHz radio transceiver with support for multiple modulations and packet formats. The CC1314R10 device provides similar functionality optimized for the Sub-1 GHz bands. The CC1354x10 devices are a true dual-band radio with separate RF paths for Sub-1 GHz and 2.4 GHz operation. The radio subsystem provides an interface between the MCU and the radio, which makes it possible to issue commands, read status, also automate and sequence radio events.

1.3.4 Security Core

The security subsystem of the CC13x4x10 and CC26x4x10 device platform features an Advanced Encryption Standard (AES) module with 256-bit key support, local key storage and DMA capability. It also includes a Hash engine (SHA-2) and a Public Key Acceleration (PKA) engine.

Features of the security subsystem are as follows:

- ECB, CBC, CBC-MAC, CTR, CCM, and GCM modes of operation
  - Up to 118-Mbps throughput
- Key storage memory
- Low latency
- SHA-224, SHA-256, SHA-384, and SHA-512
- Hardware accelerators for elliptic curve calculations

1.3.5 Runtime Security

The device implements TrustZone® runtime security. The purpose and scope of the TrustZone implementation it to enable users to implement a Secure enclave for managing cryptography and cryptographic processes. Please refer to the TrustZone® technology for Armv8-M Architecture documentation for the definition of the terms Secure, Non-secure callable, or Non-secure. For the purpose of this document, the meanings of these terms is assumed to be known.

1.3.6 General-Purpose Timers

General-purpose timers can be used to count or time external events that drive the timer-input pins. Each 16- or 32-bit GPTM block provides two 16-bit timers or counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer.

The general-purpose timer module (GPTM) contains four 16- or 32-bit GPTM blocks with the following functional options:

- 16- or 32-bit operating modes:
  - 16- or 32-bit programmable one-shot timer
  - 16- or 32-bit programmable periodic timer
  - 16-bit general-purpose timer with an 8-bit prescaler
  - 16-bit input-edge count- or time-capture modes with an 8-bit prescaler
  - 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM signal
- Count up or down
- Four 32-bit counters or up to eight 16-bit counters
- Up to eight capture/compare pins
- Up to four PWM pins (one PWM pin per 32-bit timer)
- Daisy-chaining of timer modules allows a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- User-enabled stalling when the microcontroller asserts CPU halt flag during debug
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the ISR
- Efficient transfers using the µDMA controller

1.3.6.1 Watchdog Timer

The watchdog timer is used to regain control when the system fails because of a software error or an external device fails to respond properly. The watchdog timer can generate an interrupt or a reset when a predefined time-out value is reached.

1.3.6.2 Always-On Domain

The AON domain contains circuitry that is always enabled, except for the shutdown mode (where the digital supply is off). This domain includes the following:

- **Real-Time Clock (RTC)**: The RTC can be used to wake the CC13x4x10 and CC26x4x10 device platform from any state where it is active.

  The RTC contains three match registers and one compare register. With software support, the RTC can be used for clock and calendar operation. The RTC is clocked from the 32 kHz RC oscillator or the 32 kHz crystal oscillator.
- **Battery Monitor and Temperature Sensor (BATMON)**: The battery monitor and temperature sensors are accessible by software. BATMON provides continuous monitoring of battery state as well as coarse temperature.
- **Always On Watchdog Timer (AON_WDT)**: AON_WDT is used to regain control when the system has failed due to a software error or to the failure of an external device to respond in the expected way specifically during standby

1.3.7 Direct Memory Access

The CC13x4x10 and CC26x4x10 device platform includes a DMA controller, known as µDMA. The µDMA controller provides a way to offload data transfer tasks from the Arm® Cortex®-M33 processor, allowing more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. Channels in the µDMA are dedicated for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory, because the peripheral is ready to transfer more data.

1.3.8 System Control and Clock

System control determines the overall operation of the CC13x4x10 and CC26x4x10 device platform. System control provides information about the devices, controls power-saving features, controls the clocking of the devices and individual peripherals, and handles reset detection and reporting.

- Power control
  - On-chip fixed DC/DC converter and low drop-out (LDO) voltage regulators
  - Handles the power-up sequencing, power-down sequencing, and control for the core digital-logic and analog circuits
  - Low-power options for the microcontroller
  - Low-power options for on-chip modules
    - Software controls shutdown of individual peripherals and memory
    - RAM and configuration registers are retained in all power modes
  - Control-pin option for control of external DC/DC regulator
  - Configurable wake up from sleep timer or any GPIO interrupt
  - Voltage supervision circuitry
- Multiple clock sources for microcontroller system clock
  - High-frequency clock
    - RC oscillator (RCOSC_HF): on-chip 48-MHz RC oscillator.
    - External oscillator (XOSC_HF): external 48-MHz crystal oscillator connected across the X48M_P input and X48M_N output pins
    - Radio operation requires the external oscillator
  - Low-frequency clock
    - RC oscillator (RCOSC_LF): on-chip 32-kHz RC oscillator
    - External oscillator (XOSC_LF): external 32.768-kHz crystal oscillator connected across the X32K_Q1 input and X32K_Q2 output pins
    - External signal: an external 32.768-kHz clock signal can be supplied by using one of the digital input/output (DIO) pins as clock input
    - For accurate RTC operation or synchronous network timing
    - Used during power-saving modes and for RTC
  - CPU and periphery clock division options

1.3.9 Serial Communication Peripherals

The CC13x4x10 and CC26x4x10 device platform supports both asynchronous and synchronous serial communication including:

- Four UART modules
- Two I^2C modules
- I^2S module
- Four SPI modules

The following subsections provide more detail on each of the communication functions.

1.3.9.1 UART

A UART is an integrated circuit used for RS-232C serial communications. A UART contains a transmitter (parallel-to-serial converter) and a receiver (serial-to-parallel converter); each is clocked separately.

The CC13x4x10 and CC26x4x10 device platform includes a fully programmable UART. The UART can generate individually masked interrupts from the receive (RX), transmit (TX), modem flow control, and error conditions. The module generates one combined interrupt when any of the interrupts are asserted and are unmasked.

The UARTs has the following features:

- Programmable baud-rate generator allows speeds up to 3 Mbps
- Separate 32 × 8 TX FIFOs and 32 × 16 RX FIFOs reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation that provides conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation and detection
  - One or two stop-bit generation
- Full modem-handshake support
- Programmable hardware flow control
- Standard FIFO-level interrupts
- Efficient transfers using the µDMA controller:
  - Separate channels for TX and RX
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

1.3.9.2 I^2C

The I^2C bus provides bidirectional data transfer through a 2-wire design (a serial data line SDA and a serial clock line SCL). The I^2C bus interfaces to external I^2C devices such as serial memory (RAM and ROM), networking devices, LCDs, tone generators, and so on. The I^2C bus may also be used for system testing and diagnostic purposes in product development and manufacturing.

Each device on the I^2C bus can be designated as a master or a slave. Each I^2C module supports both sending and receiving data (as either a master or a slave) and can operate simultaneously (as both a master and a slave). Both the I^2C master and slave can generate interrupts.

The CC13x4x10 and CC26x4x10 device platform includes an I^2C module with the following features:

- Devices on the I^2C bus can be designated as either a master or a slave:
  - Supports both transmitting and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I^2C modes:
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive

- Two transmission speeds:
  - Standard (100 kbps)
  - Fast (400 kbps)
- Clock low time-out interrupt
- Master and slave interrupt generation:
  - Master generates interrupts when a TX or RX operation completes (or aborts due to an error)
  - Slave generates interrupts when data is transferred or requested by a master or when a START or STOP condition is detected
  - Master with arbitration and clock synchronization, multi-master support, and 7-bit addressing mode

1.3.9.3 I^2S

An I^2S module enables the CC13x4x10 and CC26x4x10 device platform to communicate with external devices like codecs, DAC, ADCs, or DSPs. The devices only support audio streaming formats like I^2S, RJF, LJF, and DSP; the devices do not support configuration of external devices. The CC13x4x10 and CC26x4x10 device platform supports both external and internally generated bit clock and word clock (BCLK and WCLK).

1.3.9.4 SPI

An SPI module is a 3-wire or 4-wire bidirectional communication interface that converts data between parallel and serial. The SPI performs serial-to-parallel conversion on data received from a peripheral device and performs parallel-to-serial conversion on data transmitted to a peripheral device. The SPI can be configured as either a master or slave device. As a slave device, the SPI can be configured to disable its output, which allows coupling of a master device with multiple slave devices. The TX and RX paths are buffered with separate internal FIFOs.

The SPI also includes a programmable bit rate clock divider and prescaler to generate the output serial clock derived from the input clock of the SPI. Bit rates are generated based on the input clock, and the maximum bit rate is determined by the connected peripheral.

The CC13x4x10 and CC26x4x10 device platform includes four SPI modules with the following features:

- Programmable interface operation for Motorola SPI (3-wire and 4-wire), MICROWIRE, or TI synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate TX and RX FIFOs, each 32 bits wide and 8 locations deep
- Programmable data-frame size from 4 bits to 32 bits
- Internal loopback test mode for diagnostic and debug testing
- Standard FIFO-based interrupts and EoT interrupt
- Efficient transfers using the µDMA controller:
  - Separate channels for TX and RX
  - Receive single request asserted when data is in the FIFO; burst request is asserted when FIFO contains a configurable number of entries
  - Transmit single request asserted when there is space in the FIFO; burst request is asserted when FIFO contains a configurable number of entries

1.3.10 Programmable I/Os

I/O pins offer flexibility for a variety of connections. The CC13x4x10 and CC26x4x10 device platform supports highly configurable I/O pins that can be multiplexed to any digital peripheral through the I/O Controller.

Note

Analog functionality, Sensor Controller connections, and high-drive strength is limited to certain pins. See Chapter 15 for details.

- Up to 46 GPIOs, depending on configuration
- Up to five 8 mA drive strength pins
- Fully flexible digital pin muxing allows use as GPIO or any of several peripheral functions
- Programmable control for GPIO interrupts:
  - Interrupt generation masking per pin
  - Edge-triggered on rising or falling
- Bit masking in read and write operations through address lines
- Can initiate a µDMA transfer
- Pin state can be retained during all sleep modes
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for DIO configuration:
  - Weak pullup or pulldown resistors
  - Digital input enables

1.3.11 Sensor Controller

The sensor controller contains circuitry that can be selectively enabled in the power-down mode. The peripherals in this domain may be controlled by the sensor controller, which is a proprietary power-optimized CPU (sensor controller engine), or directly from the System CPU. The sensor controller engine CPU can read and monitor sensors or perform other tasks autonomously, thereby reducing power consumption and offloading the System CPU.

The sensor controller is set up using a PC-based configuration tool, and typical use cases may be (but not limited to) the following:

- Analog sensors using integrated ADC
- Digital sensors using GPIO with bit-banged I^2C and SPI
- Capacitive sensing
- Waveform generation
- Keyboard scan
- Quadrature decoder for polling rotation sensors
- Oscillator calibration

The peripherals in the sensor interface include the following:

- Analog comparator

  The ultra-low-power analog comparator can wake the CC13x4x10 and CC26x4x10 device platform from any active state. A configurable internal reference can be used with the comparator. The output of the comparator can also trigger an interrupt or trigger the ADC.
- Capacitive sensing

  Capacitive sensing is not a stand-alone module in the CC13x4x10 and CC26x4x10 device platform; rather, the functionality is achieved through the use of a constant current source, a time to digital converter, and a comparator. The analog comparator in this block can also be used as a higher-accuracy alternative to the ultra-low-power comparator. The sensor controller takes care of baseline tracking, hysteresis, filtering, and other related functions.

- ADC

  The ADC is a 12-bit, 200 ksamples/s ADC with 8 inputs and a built-in voltage reference. The ADC can be triggered by many different sources including timers, I/O pins, software, the analog comparator, and the RTC.

  An ADC is a peripheral that converts a continuous analog voltage to a discrete digital number. The ADC module features 12-bit conversion resolution and supports eight input channels plus an internal division of the battery voltage and a temperature sensor.
- Low-power UART, SPI, and I^2C digital sensor interface

  The analog modules can be connected to up to eight different I/Os.

1.3.12 Random Number Generator

The random number generator generates true random numbers for backoff calculations or security keys.

1.3.13 cJTAG and JTAG

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a test access port (TAP) and boundary scan architecture for digital integrated circuits. The JTAG port also provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards (PCBs) and obtain manufacturing information on the components. The JTAG port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging. The compact JTAG (cJTAG) interface has the following features:

- IEEE 1149.1-1990 compliant TAP controller
- IEEE 1149.7 cJTAG interface
- ICEPick JTAG router
- A 4-bit IR chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE and PRELOAD, EXTEST and INTEST
- Arm® additional instructions: APACC, DPACC, and ABORT

1.3.14 Power Supply System

1.3.14.1 Supply System

There are several voltage levels in use on the CC13x4x10 and CC26x4x10 device platform. Figure 1-2 shows an overview of the supply system.

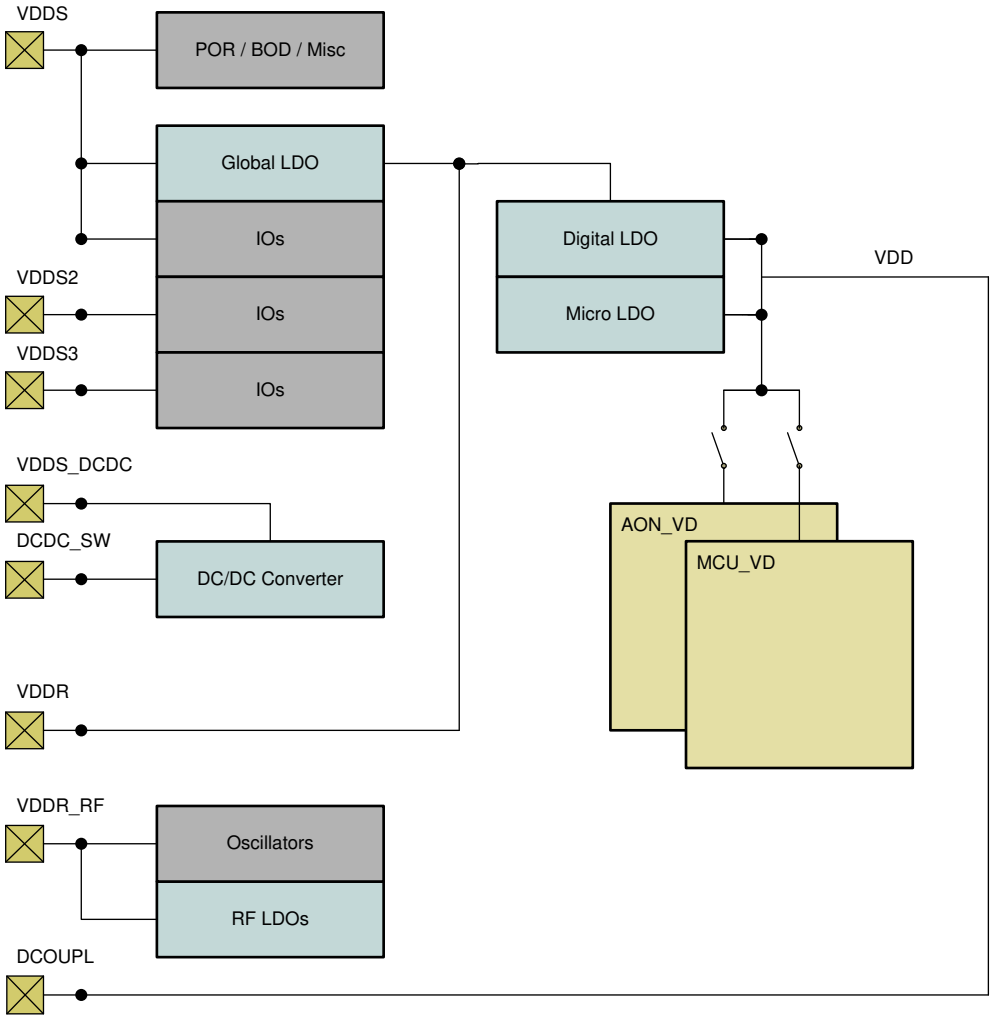


Figure 1-2. CC13x4x10 and CC26x4x10 Supply System

1.3.14.1.1 VDDS

The battery voltage on the CC13x4x10 and CC26x4x10 device platform is called VDDS (supply). This supply has the highest potential in the system and typically is the only one provided by the user.

Note

VDDS2 and VDDS3 must always be at the same potential as VDDS.

1.3.14.1.2 VDDR

The two VDDR (regulated) pins are normally powered from one of the internal regulators. For lowest power, TI recommends using the internal DC/DC regulator (see Section 1.3.14.2 for further details on this configuration).

Using the Global LDO is also an option. In this case the two VDDR pins must be tied together. In this case, VDDR should have a 22 µF decoupling capacitor, whereas VDDR_RF should have the decoupling recommended in the various reference designs. In this setup, VDDS_DCDC should be tied to VDDS and DCDC_SW should be left floating.

1.3.14.1.3 Digital Core Supply

The digital core of the CC13x4x10 and CC26x4x10 device platform is supplied by a 1.28 V regulator connected to VDDR. The output of this regulator requires an external decoupling capacitor for proper operation; this capacitor must be connected to the DCOUPL pin.

Note

The DCOUPL pin cannot be used to supply external circuitry.

When the system is in power down, a small low-power regulator (micro LDO) with limited current capacity supplies the digital domain to ensure enabled modules still have power.

1.3.14.1.4 Other Internal Supplies

Several other modules in the device (such as the frequency synthesizer, RF power amplifier, and so forth) have separate internal regulators running at either 1.4 V (analog modules) or 1.28 V (digital modules). These regulators are powered up or down automatically by firmware when needed.

1.3.14.2 DC/DC Converter

The on-chip buck-mode DC/DC converter provides a simple way to reduce the power consumption of the device. The DC/DC converter is integrated into the supply system and handles bias and clocks automatically through the system controller.

The DC/DC converter is controlled through the AON_PMCTL:PWRCTL register.

To enable the DC/DC converter when the system is active, the AON_PMCTL:PWRCTL.DCDC_ACTIVE bit must be set. The DC/DC converter is also used periodically when the device is in Standby mode to maintain voltage on the VDDR domain.

The output voltage of the DC/DC regulator is typically trimmed to 1.68 V, but there are use cases where other voltage levels are used. The voltage levels are controlled automatically by the device and cannot be changed by the user.

Note

The DC/DC regulator output cannot be used to supply external circuitry.

This page intentionally left blank.

Chapter 2

Arm® Cortex®-M33 Processor with FPU

The Arm® Cortex®-M33 core brings high-performance 32- bit computing to cost-sensitive embedded microcontroller applications, such as factory automation and control, industrial control power devices, building and home automation, and stepper motor control.

Note

This chapter provides information on the Arm® Cortex®-M33 processor, and content to this chapter is taken from the Arm® documentation listed in Section Related Documentation.

2.1 Arm® Cortex®-M33 Processor Introduction 58
2.2 Block Diagram 58
2.3 Overview 59
2.4 Programming Model 62
2.5 Arm® Cortex®-M33 Registers 69

2.1 Arm® Cortex®-M33 Processor Introduction

The Cortex-M33 processor is a high performance, energy efficient processor. It is intended for microcontroller and embedded applications that require an efficient mix of control capability and signal processing instructions. The processor is based on the Armv8-M architecture and is primarily for use in environments where security is an important consideration. The processor uses a Harvard architecture, characterized by separate buses for instructions and data. It provides an in-order pipeline to execute Thumb® and Thumb-2 instructions, including saturating arithmetic operations.

The following features are included:

- Arm® TrustZone® technology, using the Armv8-M Security Extension supporting Secure and Non-secure states
- Floating-point Extension
- Digital Signal Processing (DSP) Extension
- Memory Protection Unit (MPU) Extension
- Security Attribution Unit (SAU) Extension
- Instruction Trace Macrocell (ITM) Extension
- Wakeup Interrupt Controller (WIC) Extension
- Nested Vectored Interrupt Controller (NVIC)
- Trace Port Interface Unit (TPIU) with a Trace Port Analyzer (TPA) including Serial Wire Output (SWO) mode
- Data Watchpoint and Trace (DWT), and Breakpoint Unit (BPU) accessible over a JTAG debug port
- ROM table allowing debuggers to determine which components are implemented in the Cortex-M33 processor
- Dedicated hardware division
- Ultra-low power consumption with integrated sleep modes

2.2 Block Diagram

Figure 2-1 shows the Core Processor Unit (CPU) block diagram.

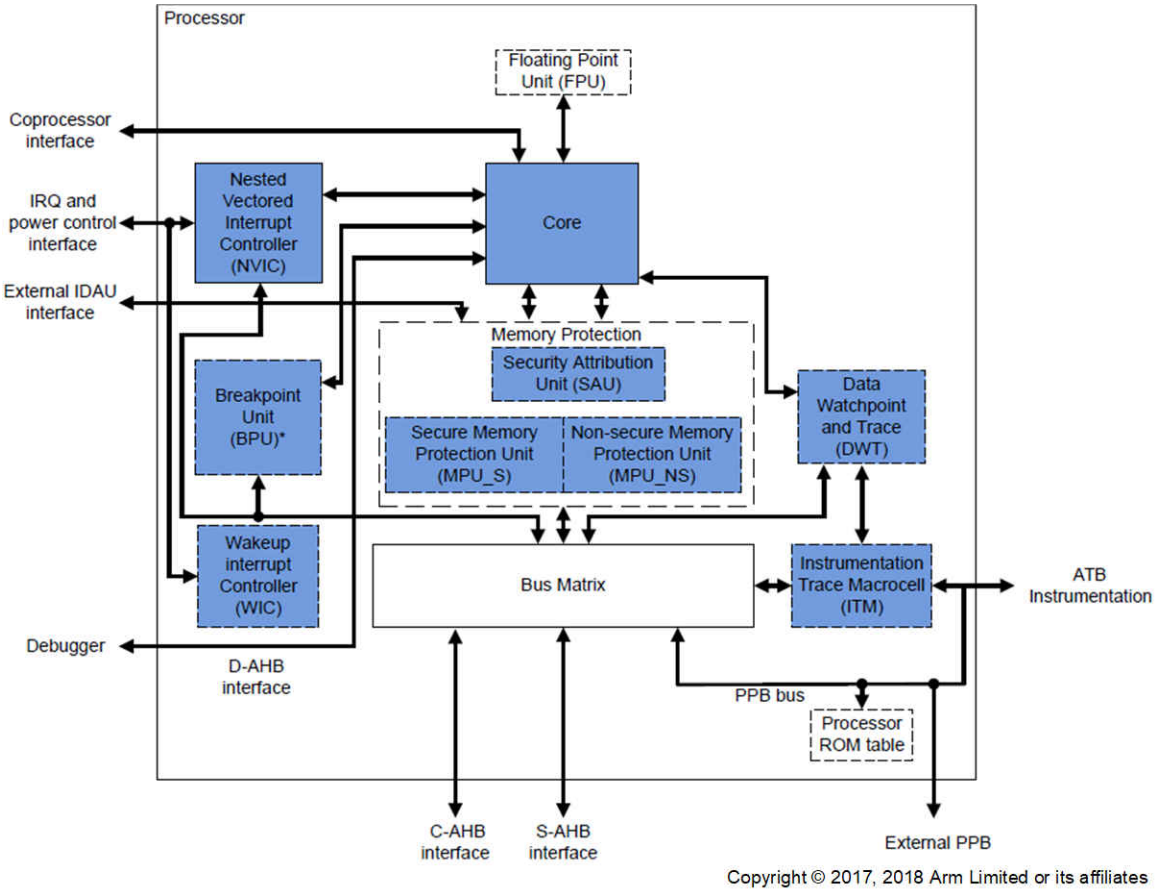


Figure 2-1. Block Diagram

2.3 Overview

2.3.1 Integrated Configurable Debug

The Arm® Cortex®-M33 processor debug functionality includes processor halt, single-step, processor core register access, Vector Catch, unlimited software breakpoints, and full system memory access. The processor also includes support for hardware breakpoints and watchpoints:

- A breakpoint unit supporting eight instruction comparators
- A watchpoint unit supporting four data watchpoint comparators

The Cortex®-M33 processor supports system level debug permissions to control access from a debugger to resources and memory. The authentication can be used to allow a debugger full access to Non-secure code and data without exposing any Secure information. All debug registers are accessible by the D-AHB interface. D-AHB interface accesses are only in little-endian format.

The debug is done through JTAG and cJTAG. Figure 2-2 shows a diagram of the JTAG connection with the processor.

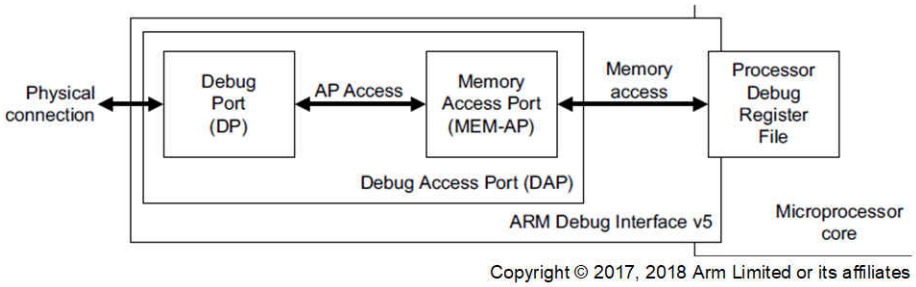


Figure 2-2. Simple Arm Debug Interface MEM-AP Implementation

See the *Armv8-M Architecture Reference Manual* for more information.

2.3.2 Trace Port Interface Unit

Figure 2-3 shows the trace port interface unit (TPIU) block diagram (main blocks of the TPIU and the clock domains). The TPIU acts as a bridge between the Arm® Cortex® -M33 trace data from the ITM, and an off-chip trace port analyzer.

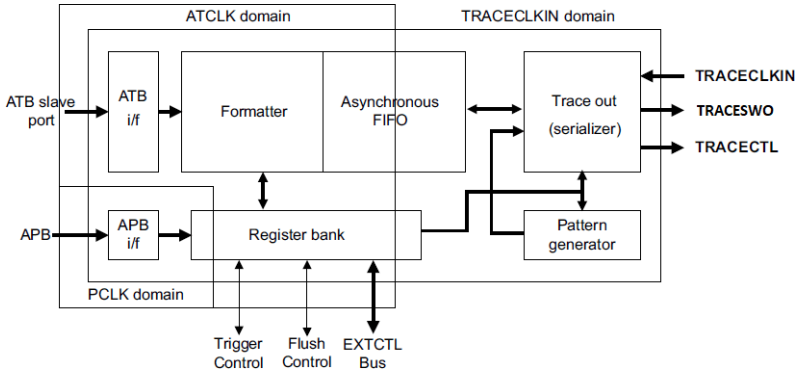


Figure 2-3. TPIU Block Diagram

2.3.3 Arm® Cortex®-M33 System Peripheral Details

The system components of the Arm® Cortex®-M33 is described in the following sections.

2.3.3.1 Floating Point Unit (FPU)

The Arm® Cortex®-M33 FPU is an implementation of the single precision variant of the Armv8-M Floating point extension, FPv5 architecture. It provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

The FPU supports all single-precision data-processing instructions and data types described in the *Armv8-M Architecture Reference Manual*.

2.3.3.2 Memory Protection Unit (MPU)

The MPU improves system reliability by defining the memory attributes for different memory regions. There are two MPUs, one Secure and one Non-secure. Each MPU can define memory attributes independently. There are 8 Secure and 8 Non-secure memory regions.

2.3.3.3 System Timer (SysTick)

The processor has two 24-bit system timers, a Non-secure SysTick timer and a Secure SysTick timer. When enabled, each timer counts down from the reload value to zero, reloads (wraps to) the value in the CPU_SYSTICK:RVR on the next clock cycle, then decrements on subsequent clock cycles. When the processor is halted for debugging, the counter does not decrement.

2.3.3.4 Nested Vectored Interrupt Controller (NVIC)

An embedded interrupt controller (INTC) that supports low-latency interrupt processing and has a programmable priority level ranging from 0-255. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority. There is also an external non-maskable interrupt.

2.3.3.5 System Control Block (SCB)

The System Control Block (SCB) is an address region in the System Control Space (SCS) that provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

2.3.3.6 System Control Space (SCS)

The System Control Space (SCS) is the programmer's model interface to the processor. The SCS provides system implementation information and system control. The processor provides debug through registers in the SCS. The SCS includes the following regions:

- Implementation Control Block (ICB)
- System Timer (SYSTICK)
- Nested Vectored Interrupt Controller (NVIC)
- System Control Block (SCB)
- Memory Protection Unit (MPU)
- Debug Control Block (DCB)
- Software Interrupt Generator (SIG)
- Floating Point Unit (FPU)

2.3.3.7 Security Attribution Unit (SAU)

The Security Attribution Unit (SAU) determines the security of an address accessed by the processor. For instructions, the SAU returns the security attribute (Secure or Non-secure) and identifies whether the instruction address is in a Non-secure callable region.

For data, the SAU returns the security attribute (Secure or Non-secure). When a memory access is performed, the security of the address is verified by the SAU. Any address that matches multiple SAU regions is marked with the most secure attribute of the matching regions. There are 4 regions for which security can be defined.

The SAU is used with the IDAU and other security configuration settings to fully specify addresses as Secure, Non-secure, or Non-secure callable. See Section 2.4.7 for more information.

2.4 Programming Model

The Cortex®-M33 programmer's model is an implementation of the Armv8-M Main Extension architecture. For a complete description of the programmer's model, refer to the *Armv8-M Architecture Reference Manual*, which also contains the Armv8-M Thumb® instructions.

2.4.1 Modes of Operation and Execution

The Cortex®-M33 processor supports Secure and Non-secure security states, Thread and Handler operating modes, and can run in either Thumb® or Debug operating states. In addition, the processor can limit or exclude access to some resources by executing code in privileged or unprivileged mode. See the *Armv8-M Architecture Reference Manual* for more information about the modes of operation and execution.

2.4.1.1 Security States

The programmers model includes two orthogonal security states, Secure state and Non-secure state. The processor always resets into Secure state. Each security state includes a set of independent operating modes and supports both privileged and unprivileged user access. Registers in the System Control Space are banked across Secure and Non-secure state, with the Non-secure register view available at an aliased address to Secure state.

2.4.1.2 Operating Modes

For each security state, the processor can operate in Thread or Handler mode. The conditions which cause the processor to enter Thread or Handler mode are as follows:

- The processor enters Thread mode on reset, or as a result of an exception return to Thread mode. Privileged and Unprivileged code can run in Thread mode.
- The processor enters Handler mode as a result of an exception. All code is privileged in Handler mode.

The processor can change security state on taking an exception, for example when a Secure exception is taken from Non-secure state, the Thread mode enters the Secure state Handler mode. The processor can also call Secure functions from Non-secure state and Non-secure functions from Secure state. The Security Extension includes requirements for these calls to prevent Secure data from being accessed in Non-secure state.

2.4.1.3 Operating States

The processor can operate in Thumb or Debug state:

- Thumb state is the state of normal execution running 16-bit and 32-bit halfword-aligned Thumb instructions.
- Debug state is the state when the processor is halted in debug.

2.4.1.4 Privileged Access and Unprivileged User Access

Code can execute as privileged or unprivileged. Unprivileged execution limits or excludes access to some resources appropriate to the current security state. Privileged execution has access to all resources available to the security state. Handler mode is always privileged. Thread mode can be privileged or unprivileged.

2.4.2 Instruction Set Summary

The processor implements the following instruction from Armv8-M:

- All base instructions
- All instructions in the Main Extension
- All instructions in the Security Extension
- All instructions in the DSP Extension
- All single-precision instructions and double precision load and store instructions in the Floating-point Extension

For more information about Armv8-M instructions, see the *Armv8-M Architecture Reference Manual*.

2.4.3 Memory Model

The processor contains a bus matrix that arbitrates instruction fetches and memory accesses from the processor core between the external memory system and the internal System Control Space (SCS) and debug components. Priority is given to the processor to ensure that any debug accesses are as nonintrusive as possible. The system memory map is Armv8-M Main Extension compliant, and is common both to the debugger and processor accesses. The default memory map provides user and privileged access to all regions except for the Private Peripheral Bus (PPB). The PPB space is privileged access only.

The security level associated with an address is determined by either the internal Secure Attribution Unit (SAU) or an external Implementation Defined Attribution Unit (IDAU) in the system (see Section 2.4.7.1 for more details). Some internal peripherals have memory-mapped registers in the PPB region which are banked between Secure and Non-secure state. When the processor is in Secure state, software can access both the Secure and Non-secure versions of these registers. The Non-secure versions are accessed using an aliased address. See the *Armv8-M Architecture Reference Manual* and Chapter 3 for more information about the memory model.

2.4.3.1 Private Peripheral Bus

The Private Peripheral Bus (PPB) memory region provides access to internal and external processor resources. The internal PPB provides access to:

- The System Control Space (SCS), including the Memory Protection Unit (MPU), Secure Attribution Unit (SAU), and the Nested Vectored Interrupt Controller (NVIC)
- The Data Watchpoint and Trace (DWT)
- The Breakpoint Unit (BPU), if included
- The ROM table

The external PPB (EPPB) provides access to implementation-specific external areas of the PPB memory map.

2.4.3.2 Unaligned Accesses

The Cortex®-M33 processor supports unaligned accesses. They are converted into two or more aligned AHB transactions on the C-AHB or S-AHB master ports on the processor. Unaligned support is only available for load/store singles (LDR, LDRH, STR, STRH, TBH) to addresses in normal memory. Load/store double and load/store multiple instructions already support word aligned accesses, but do not permit other unaligned accesses, and generate a fault if this is attempted. Unaligned accesses in device memory are not permitted and generate a fault. Unaligned accesses that cross memory map boundaries are architecturally unpredictable.

Note

If CPU_SCB:CCR.UNALIGN_TRP for the current security state is set, any unaligned accesses generate a fault.

2.4.4 Exclusive Monitor

The Cortex®-M33 processor implements a local exclusive monitor. The local monitor within the processor has been constructed so that it does not hold any physical address, but instead treats any store-exclusive access as matching the address of the previous load-exclusive. This means that the implemented exclusives reservation granule is the entire memory address range.

For more information about semaphores and the local exclusive monitor, see the *Armv8-M Architecture Reference Manual*.

2.4.5 Processor Core Registers Summary

Table 2-1 shows the processor core register set summary. Each of these registers is 32 bits wide. Some of the registers are banked. The Secure view of these registers is available when the Cortex-M33 processor is in Secure state and the Non-secure view when Cortex-M33 processor is in Non-secure state.

Table 2-1. Processor Core Register Set Summary

Name	Description
R0 - R12	R0-R12 are general-purpose registers for data operations
MSP (R13) PSP (R13)	The Stack Pointer (SP) is register R13. In Thread mode, the CONTROL register indicates which stack pointer to use, Main Stack Pointer (MSP) or Process Stack Pointer (PSP). There are two MSP registers in the Cortex-M33 processor: • MSP_NS for the Non-secure state • MSP_S for the Secure state. There are two PSP registers in the Cortex-M33 processor: • PSP_NS for the Non-secure state • PSP_S for the Secure state
MSPLIM PSPLIM	The stack limit registers limit the extent to which the MSP and PSP registers can descend respectively. There are two MSPLIM registers in the Cortex-M33 processor: • MSPLIM_NS for the Non-secure state • MSPLIM_S for the Secure state There are two PSPLIM registers in the Cortex-M33 processor: • PSPLIM_NS for the Non-secure state • PSPLIM_S for the Secure state
LR (R14)	The Link Register (LR) is register R14 and stores the return information for subroutines, function calls, and exceptions.
PC (R15)	The Program Counter (PC) is register R15 and contains the current program address
PSR	The Program Status Register (PSR) combines: • Application Program Status Register (APSR) • Interrupt Program Status Register (IPSR) • Execution Program Status Register (EPSR) These registers provide different views of the PSR
PRIMASK	The PRIMASK register prevents activation of exceptions with configurable priority. There are two PRIMASK registers in the Cortex-M33 processor: • PRIMASK_NS for the Non-secure state • PRIMASK_S for the Secure state
BASEPRI	The BASEPRI register defines the minimum priority for exception processing. There are two BASEPRI registers in the Cortex-M33 processor: • BASEPRI_NS for the Non-secure state • BASEPRI_S for the Secure state
FAULTMASK	The FAULTMASK register prevents activation of all exceptions except for NON-MASKABLE INTERRUPT (NMI) and optionally Secure HardFault. There are two FAULTMASK registers in the Cortex-M33 processor: • FAULTMASK_NS for the Non-secure state • FAULTMASK_S for the Secure state

Table 2-1. Processor Core Register Set Summary (continued)

Name	Description
CONTROL	The CONTROL register controls the stack used, and optionally the privilege level, when the processor is in Thread mode. There are two CONTROL registers in the Cortex-M33 processor: • CONTROL_NS for the Non-secure state • CONTROL_S for the Secure state

Note

See the *Armv8-M Architecture Reference Manual* for information about the processor core registers and their addresses, access types, and reset values.

2.4.6 Exceptions

Exceptions are handled and prioritized by the processor and the NVIC. In addition to architecturally defined behavior, the processor's advanced exception and interrupt handling allows implementation defined behavior to benefit from the reduced interrupt latency.

2.4.6.1 Exception Handling and Prioritization

The processor core and the Nested Vectored Interrupt Controller (NVIC) together prioritize and handle all exceptions.

When handling exceptions:

- All exceptions are handled in Handler mode.
- Processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).
- The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining that enables back-to-back interrupts without the overhead of state saving and restoration.

Software can configure the priorities of these interrupts.

Exceptions can be specified as either Secure or Non-secure. When an exception is taken, the processor switches to the associated security state. The priority of Secure and Non-secure exceptions can be programmed independently. It is possible to deprioritize Non-secure configurable exceptions using the CPU_SCB.AIRCR.PRIS bit field to enable Secure interrupts to take priority (see Section 2.5.5).

When taking and returning from an exception, the register state is always stored using the stack pointer associated with the background security state. When taking a Non-secure exception from Secure state, all the register state is stacked and then registers are cleared to prevent Secure data being available to the Non-secure handler. The vector base address (CPU_SCB:VTOR), located in the System Control Block, is banked between Secure and Non-secure state (see Section 2.5.5). VTOR_S contains the Secure vector base address, and VTOR_NS contains the Non-secure vector base address. These registers can be programmed by software, and also initialized at reset by the system.

Note

Vector table entries are compatible with interworking between Arm and Thumb instructions. This causes bit[0] of the vector value to load into the Execution Program Status Register (EPSR) T-bit on exception entry. All populated vectors in the vector table entries must have bit[0] set. Creating a table entry with bit[0] clear generates an INVSTATE fault on the first instruction of the handler corresponding to this vector.

2.4.7 Runtime Security

TrustZone runtime security consists of:

- Watermark registers that determine which parts of flash and SRAM are marked Secure, Non-secure, or Non-secure callable
- An address range (0x58000000 - 0x5FFFFFFF) and (0x78000000 - 0x7FFFFFFF) that is marked as Secure
- Relocation of critical periphery and some singular registers into this secure range, see Section 3.2.1 for details
- Security Attribute Unit (SAU), with 4 regions
- Secure Memory Protection Unit (MPU) with 8 regions

2.4.7.1 IDAU Watermark Registers

The watermark registers determine which parts of SRAM and Flash are considered Secure, Non-secure callable, or Non-secure. From base address and counting up, these registers define three regions which are first Secure, then Non-secure callable, then Non-secure. These settings are then used to apply the attributes to the background memory map for the Cortex M33 through the IDAU and are used in the bus matrix to block access to Secure memory from Non-secure bus initiators (for example µDMA or I^2S).

The following watermark registers are described in Section 7.8.1.

- PRCM:NVMNSCADDR
- PRCM:NVMNSADDR
- PRCM:SRAMNSADDR
- PRCM:SRAMNSCADDR
- PRCM:BUSSECCFG
- PRCM:CPULOCK

2.4.7.2 Secure Memory Range for Registers

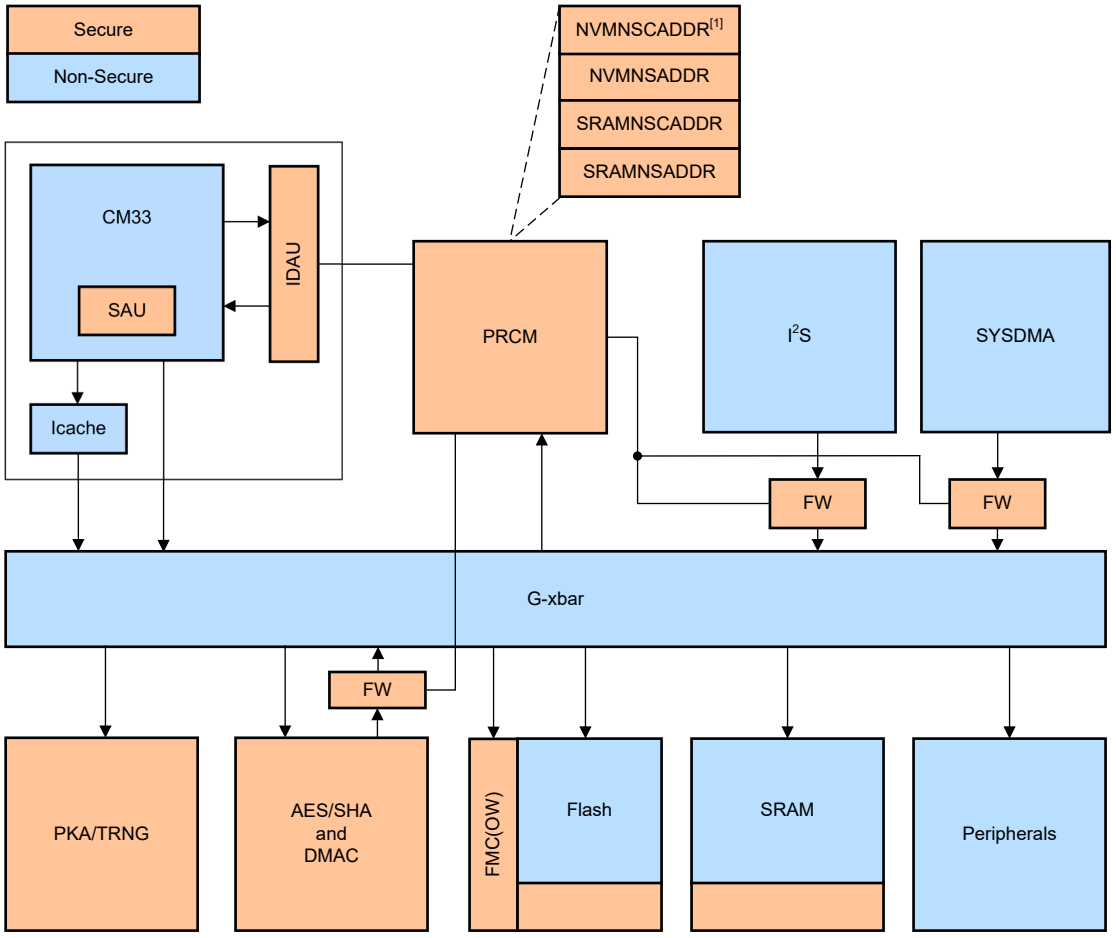
Some memory mapped registers are relocated to a statically defined Secure range to prevent Non-secure code and bus initiators from affecting these. These are located in a Secure region because they can either possibly be used to elevate privilege or they constitute assets the solution is intended to protect.

In general this moves cryptographic accelerators and the TRNG to the Secure range as well as flash and SRAM control registers and a selection of system management registers.

See Chapter 3 for more details.

2.4.7.3 Bus Topology

Figure 2-4 shows the bus topology.



[1] At boot, the following is copied from CCFG to PRCM:

CCFG:TRUSTZONE_FLASH_CFG.NSCADDR_BOUNDARY to PRCM:NVMNSCADDR
CCFG:TRUSTZONE_FLASH_CFG.NSADDR_BOUNDARY to PRCM:NVMNSADDR
CCFG:TRUSTZONE_SRAM_CFG.NSCADDR_BOUNDARY to PRCM:SRAMNSCADDR
CCFG:TRUSTZONE_SRAM_CFG.NSADDR_BOUNDARY to PRCM:SRAMNSADDR

Figure 2-4. Bus Topology

The CCFG block contains the watermark registers that control the IDAU as well as other initiators on the system. The firewalls on I^{2}S, SYSDMA and AES are controlled by the PRCM:BUSSECCFG setting, as is access to registers marked Secure.

2.4.7.4 Intended Use

The intended use is to implement a Secure-side execution environment that manages security sensitive system tasks and cryptography. System tasks include writing to Flash (can affect the security execution environment), managing power (can effect execution flow and hence indirectly affect Secure code) as well as some other system settings that might affect the security execution environment.

The watermark registers can be configured either by the pre-allocated CCFG settings or by the initialization of the program that runs in Secure execution environment. The watermark registers statically configure the IDAU to determine Secure regions and Non-secure regions in Flash and SRAM memory. These Secure memory regions (Flash and SRAM) along with Secure periphery, isolated from code running as Non-secure, can then be used for cryptographic operations and key management.

A lock mechanism is provided so that the settings can be locked into place until the next reset.

The user can use the Secure Attribution Unit (SAU) to further configure other memory attributes to extend the functionality.

However, SAU attributes are only enforced for the Cortex-M33. The SAU attributes are not enforced for other bus initiators in the system, including the µDMA, I^2S, and AES and Hash Cryptoprocessor. Therefore, use of the SAU is not recommended.

2.5 Arm® Cortex®-M33 Registers

2.5.1 CPU_ITM Registers

Table 2-2 lists the memory-mapped registers for the CPU_ITM registers. All register offset addresses not listed in Table 2-2 should be considered as reserved locations and the register contents should not be modified.

Table 2-2. CPU_ITM Registers

Offset	Acronym	Register Name	Section
0h	STIM0	Provides the interface for generating Instrumentation packets	Section 2.5.1.1
4h	STIM1	Provides the interface for generating Instrumentation packets	Section 2.5.1.2
8h	STIM2	Provides the interface for generating Instrumentation packets	Section 2.5.1.3
Ch	STIM3	Provides the interface for generating Instrumentation packets	Section 2.5.1.4
10h	STIM4	Provides the interface for generating Instrumentation packets	Section 2.5.1.5
14h	STIM5	Provides the interface for generating Instrumentation packets	Section 2.5.1.6
18h	STIM6	Provides the interface for generating Instrumentation packets	Section 2.5.1.7
1Ch	STIM7	Provides the interface for generating Instrumentation packets	Section 2.5.1.8
20h	STIM8	Provides the interface for generating Instrumentation packets	Section 2.5.1.9
24h	STIM9	Provides the interface for generating Instrumentation packets	Section 2.5.1.10
28h	STIM10	Provides the interface for generating Instrumentation packets	Section 2.5.1.11
2Ch	STIM11	Provides the interface for generating Instrumentation packets	Section 2.5.1.12
30h	STIM12	Provides the interface for generating Instrumentation packets	Section 2.5.1.13
34h	STIM13	Provides the interface for generating Instrumentation packets	Section 2.5.1.14
38h	STIM14	Provides the interface for generating Instrumentation packets	Section 2.5.1.15
3Ch	STIM15	Provides the interface for generating Instrumentation packets	Section 2.5.1.16
40h	STIM16	Provides the interface for generating Instrumentation packets	Section 2.5.1.17
44h	STIM17	Provides the interface for generating Instrumentation packets	Section 2.5.1.18
48h	STIM18	Provides the interface for generating Instrumentation packets	Section 2.5.1.19
4Ch	STIM19	Provides the interface for generating Instrumentation packets	Section 2.5.1.20
50h	STIM20	Provides the interface for generating Instrumentation packets	Section 2.5.1.21
54h	STIM21	Provides the interface for generating Instrumentation packets	Section 2.5.1.22
58h	STIM22	Provides the interface for generating Instrumentation packets	Section 2.5.1.23
5Ch	STIM23	Provides the interface for generating Instrumentation packets	Section 2.5.1.24

Table 2-2. CPU_ITM Registers (continued)

Offset	Acronym	Register Name	Section
60h	STIM24	Provides the interface for generating Instrumentation packets	Section 2.5.1.25
64h	STIM25	Provides the interface for generating Instrumentation packets	Section 2.5.1.26
68h	STIM26	Provides the interface for generating Instrumentation packets	Section 2.5.1.27
6Ch	STIM27	Provides the interface for generating Instrumentation packets	Section 2.5.1.28
70h	STIM28	Provides the interface for generating Instrumentation packets	Section 2.5.1.29
74h	STIM29	Provides the interface for generating Instrumentation packets	Section 2.5.1.30
78h	STIM30	Provides the interface for generating Instrumentation packets	Section 2.5.1.31
7Ch	STIM31	Provides the interface for generating Instrumentation packets	Section 2.5.1.32
E00h	TER0	Provide an individual enable bit for each ITM_STIM register	Section 2.5.1.33
E40h	TPR	Controls which stimulus ports can be accessed by unprivileged code	Section 2.5.1.34
E80h	TCR	Configures and controls transfers through the ITM interface	Section 2.5.1.35
EF0h	INT_ATREADY	Integration Mode: Read ATB Ready	Section 2.5.1.36
EF8h	INT_ATVALID	Integration Mode: Write ATB Valid	Section 2.5.1.37
F00h	ITCTRL	Integration Mode Control Register	Section 2.5.1.38
FBCh	DEVARCH	Provides CoreSight discovery information for the ITM	Section 2.5.1.39
FCCh	DEVTYPE	Provides CoreSight discovery information for the ITM	Section 2.5.1.40
FD0h	PIDR4	Provides CoreSight discovery information for the ITM	Section 2.5.1.41
FD4h	PIDR5	Provides CoreSight discovery information for the ITM	Section 2.5.1.42
FD8h	PIDR6	Provides CoreSight discovery information for the ITM	Section 2.5.1.43
FDCh	PIDR7	Provides CoreSight discovery information for the ITM	Section 2.5.1.44
FE0h	PIDR0	Provides CoreSight discovery information for the ITM	Section 2.5.1.45
FE4h	PIDR1	Provides CoreSight discovery information for the ITM	Section 2.5.1.46
FE8h	PIDR2	Provides CoreSight discovery information for the ITM	Section 2.5.1.47
FECh	PIDR3	Provides CoreSight discovery information for the ITM	Section 2.5.1.48
FF0h	CIDR0	Provides CoreSight discovery information for the ITM	Section 2.5.1.49
FF4h	CIDR1	Provides CoreSight discovery information for the ITM	Section 2.5.1.50
FF8h	CIDR2	Provides CoreSight discovery information for the ITM	Section 2.5.1.51
FFCh	CIDR3	Provides CoreSight discovery information for the ITM	Section 2.5.1.52

Complex bit access types are encoded to fit into small table cells. Table 2-3 shows the codes that are used for access types in this section.

Table 2-3. CPU_ITM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.1.1 STIM0 Register (Offset = 0h) [Reset = 00000000h]

STIM0 is shown in Table 2-4.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-4. STIM0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.2 STIM1 Register (Offset = 4h) [Reset = 00000000h]

STIM1 is shown in Table 2-5.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-5. STIM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.3 STIM2 Register (Offset = 8h) [Reset = 00000000h]

STIM2 is shown in Table 2-6.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-6. STIM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.4 STIM3 Register (Offset = Ch) [Reset = 00000000h]

STIM3 is shown in Table 2-7.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-7. STIM3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.5 STIM4 Register (Offset = 10h) [Reset = 00000000h]

STIM4 is shown in Table 2-8.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-8. STIM4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.6 STIM5 Register (Offset = 14h) [Reset = 00000000h]

STIM5 is shown in Table 2-9.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-9. STIM5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.7 STIM6 Register (Offset = 18h) [Reset = 00000000h]

STIM6 is shown in Table 2-10.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-10. STIM6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.8 STIM7 Register (Offset = 1Ch) [Reset = 00000000h]

STIM7 is shown in Table 2-11.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-11. STIM7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.9 STIM8 Register (Offset = 20h) [Reset = 00000000h]

STIM8 is shown in Table 2-12.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-12. STIM8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.10 STIM9 Register (Offset = 24h) [Reset = 00000000h]

STIM9 is shown in Table 2-13.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-13. STIM9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.11 STIM10 Register (Offset = 28h) [Reset = 00000000h]

STIM10 is shown in Table 2-14.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-14. STIM10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.12 STIM11 Register (Offset = 2Ch) [Reset = 00000000h]

STIM11 is shown in Table 2-15.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-15. STIM11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.13 STIM12 Register (Offset = 30h) [Reset = 00000000h]

STIM12 is shown in Table 2-16.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-16. STIM12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.14 STIM13 Register (Offset = 34h) [Reset = 00000000h]

STIM13 is shown in Table 2-17.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-17. STIM13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.15 STIM14 Register (Offset = 38h) [Reset = 00000000h]

STIM14 is shown in Table 2-18.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-18. STIM14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.16 STIM15 Register (Offset = 3Ch) [Reset = 00000000h]

STIM15 is shown in Table 2-19.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-19. STIM15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.17 STIM16 Register (Offset = 40h) [Reset = 00000000h]

STIM16 is shown in Table 2-20.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-20. STIM16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.18 STIM17 Register (Offset = 44h) [Reset = 00000000h]

STIM17 is shown in Table 2-21.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-21. STIM17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.19 STIM18 Register (Offset = 48h) [Reset = 00000000h]

STIM18 is shown in Table 2-22.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-22. STIM18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.20 STIM19 Register (Offset = 4Ch) [Reset = 00000000h]

STIM19 is shown in Table 2-23.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-23. STIM19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.21 STIM20 Register (Offset = 50h) [Reset = 00000000h]

STIM20 is shown in Table 2-24.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-24. STIM20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.22 STIM21 Register (Offset = 54h) [Reset = 00000000h]

STIM21 is shown in Table 2-25.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-25. STIM21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.23 STIM22 Register (Offset = 58h) [Reset = 00000000h]

STIM22 is shown in Table 2-26.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-26. STIM22 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.24 STIM23 Register (Offset = 5Ch) [Reset = 00000000h]

STIM23 is shown in Table 2-27.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-27. STIM23 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.25 STIM24 Register (Offset = 60h) [Reset = 00000000h]

STIM24 is shown in Table 2-28.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-28. STIM24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.26 STIM25 Register (Offset = 64h) [Reset = 00000000h]

STIM25 is shown in Table 2-29.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-29. STIM25 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.27 STIM26 Register (Offset = 68h) [Reset = 00000000h]

STIM26 is shown in Table 2-30.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-30. STIM26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.28 STIM27 Register (Offset = 6Ch) [Reset = 00000000h]

STIM27 is shown in Table 2-31.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-31. STIM27 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.29 STIM28 Register (Offset = 70h) [Reset = 00000000h]

STIM28 is shown in Table 2-32.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-32. STIM28 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.30 STIM29 Register (Offset = 74h) [Reset = 00000000h]

STIM29 is shown in Table 2-33.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-33. STIM29 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.31 STIM30 Register (Offset = 78h) [Reset = 00000000h]

STIM30 is shown in Table 2-34.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-34. STIM30 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.32 STIM31 Register (Offset = 7Ch) [Reset = 00000000h]

STIM31 is shown in Table 2-35.

Return to the Summary Table.

Provides the interface for generating Instrumentation packets

Table 2-35. STIM31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DISABLED	R	0h	Indicates whether the Stimulus Port is enabled or disabled
0	FIFOREADY	R	0h	Indicates whether the Stimulus Port can accept data

2.5.1.33 TER0 Register (Offset = E00h) [Reset = 00000000h]

TER0 is shown in Table 2-36.

Return to the Summary Table.

Provide an individual enable bit for each ITM_STIM register

Table 2-36. TER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	STIMENA	R/W	0h	For STIMENA[m] in ITM_TER*n, controls whether ITM_STIM(32*n + m) is enabled

2.5.1.34 TPR Register (Offset = E40h) [Reset = 00000000h]

TPR is shown in Table 2-37.

Return to the Summary Table.

Controls which stimulus ports can be accessed by unprivileged code

Table 2-37. TPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PRIVMASK	R/W	0h	For PRIVMASK[m], defines the access permissions of ITM_STIM Stimulus Ports 8m to 8m+7 inclusive

2.5.1.35 TCR Register (Offset = E80h) [Reset = 00000000h]

TCR is shown in Table 2-38.

Return to the Summary Table.

Configures and controls transfers through the ITM interface

Table 2-38. TCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23	BUSY	R	0h	Indicates whether the ITM is currently processing events
22-16	TraceBusID	R/W	0h	Identifier for multi-source trace stream formatting. If multi-source trace is in use, the debugger must write a unique non-zero trace ID value to this field
15-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-10	GTSFREQ	R/W	0h	Defines how often the ITM generates a global timestamp, based on the global timestamp clock frequency, or disables generation of global timestamps
9-8	TSPrescale	R/W	0h	Local timestamp prescaler, used with the trace packet reference clock
7-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5	STALLENA	R/W	0h	Stall the PE to guarantee delivery of Data Trace packets.
4	SWOENA	R/W	0h	Enables asynchronous clocking of the timestamp counter
3	TXENA	R/W	0h	Enables forwarding of hardware event packet from the DWT unit to the ITM for output to the TPIU
2	SYNCENA	R/W	0h	Enables Synchronization packet transmission for a synchronous TPIU
1	TSENA	R/W	0h	Enables Local timestamp generation
0	ITMENA	R/W	0h	Enables the ITM

2.5.1.36 INT_ATREADY Register (Offset = EF0h) [Reset = 00000000h]

INT_ATREADY is shown in Table 2-39.

Return to the Summary Table.

Integration Mode: Read ATB Ready

Table 2-39. INT_ATREADY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	AFVALID	R	0h	A read of this bit returns the value of AFVALID
0	ATREADY	R	0h	A read of this bit returns the value of ATREADY

2.5.1.37 INT_ATVALID Register (Offset = EF8h) [Reset = 00000000h]

INT_ATVALID is shown in Table 2-40.

Return to the Summary Table.

Integration Mode: Write ATB Valid

Table 2-40. INT_ATVALID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	AFREADY	W	0h	A write to this bit gives the value of AFREADY
0	ATREADY	W	0h	A write to this bit gives the value of ATVALID

2.5.1.38 ITCTRL Register (Offset = F00h) [Reset = 00000000h]

ITCTRL is shown in Table 2-41.

Return to the Summary Table.

Integration Mode Control Register

Table 2-41. ITCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
0	IME	R/W	0h	Integration mode enable bit - The possible values are: 0 - The trace unit is not in integration mode. 1 - The trace unit is in integration mode. This mode enables: A debug agent to perform topology detection. SoC test software to perform integration testing.

2.5.1.39 DEVARCH Register (Offset = FBCh) [Reset = 00000000h]

DEVARCH is shown in Table 2-42.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-42. DEVARCH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	ARCHITECT	R	0h	Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.
20	PRESENT	R	0h	Defines that the DEVARCH register is present
19-16	REVISION	R	0h	Defines the architecture revision of the component
15-12	ARCHVER	R	0h	Defines the architecture version of the component
11-0	ARCHPART	R	0h	Defines the architecture of the component

2.5.1.40 DEVTYPE Register (Offset = FCCh) [Reset = 00000000h]

DEVTYPE is shown in Table 2-43.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-43. DEVTYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	SUB	R	0h	Component sub-type
3-0	MAJOR	R	0h	Component major type

2.5.1.41 PIDR4 Register (Offset = FD0h) [Reset = 00000000h]

PIDR4 is shown in Table 2-44.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-44. PIDR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	SIZE	R	0h	See CoreSight Architecture Specification
3-0	DES_2	R	0h	See CoreSight Architecture Specification

2.5.1.42 PIDR5 Register (Offset = FD4h) [Reset = 00000000h]

PIDR5 is shown in Table 2-45.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-45. PIDR5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.1.43 PIDR6 Register (Offset = FD8h) [Reset = 00000000h]

PIDR6 is shown in Table 2-46.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-46. PIDR6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.1.44 PIDR7 Register (Offset = FDCh) [Reset = 00000000h]

PIDR7 is shown in Table 2-47.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-47. PIDR7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.1.45 PIDR0 Register (Offset = FE0h) [Reset = 00000000h]

PIDR0 is shown in Table 2-48.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-48. PIDR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PART_0	R	0h	See CoreSight Architecture Specification

2.5.1.46 PIDR1 Register (Offset = FE4h) [Reset = 00000000h]

PIDR1 is shown in Table 2-49.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-49. PIDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	DES_0	R	0h	See CoreSight Architecture Specification
3-0	PART_1	R	0h	See CoreSight Architecture Specification

2.5.1.47 PIDR2 Register (Offset = FE8h) [Reset = 00000000h]

PIDR2 is shown in Table 2-50.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-50. PIDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	REVISION	R	0h	See CoreSight Architecture Specification
3	JEDEC	R	0h	See CoreSight Architecture Specification
2-0	DES_1	R	0h	See CoreSight Architecture Specification

2.5.1.48 PIDR3 Register (Offset = FECh) [Reset = 00000000h]

PIDR3 is shown in Table 2-51.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-51. PIDR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	REVAND	R	0h	See CoreSight Architecture Specification
3-0	CMOD	R	0h	See CoreSight Architecture Specification

2.5.1.49 CIDR0 Register (Offset = FF0h) [Reset = 00000000h]

CIDR0 is shown in Table 2-52.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-52. CIDR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_0	R	0h	See CoreSight Architecture Specification

2.5.1.50 CIDR1 Register (Offset = FF4h) [Reset = 00000000h]

CIDR1 is shown in Table 2-53.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-53. CIDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	CLASS	R	0h	See CoreSight Architecture Specification
3-0	PRMBL_1	R	0h	See CoreSight Architecture Specification

2.5.1.51 CIDR2 Register (Offset = FF8h) [Reset = 00000000h]

CIDR2 is shown in Table 2-54.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-54. CIDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_2	R	0h	See CoreSight Architecture Specification

2.5.1.52 CIDR3 Register (Offset = FFCh) [Reset = 00000000h]

CIDR3 is shown in Table 2-55.

Return to the Summary Table.

Provides CoreSight discovery information for the ITM

Table 2-55. CIDR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_3	R	0h	See CoreSight Architecture Specification

2.5.2 CPU_DWT Registers

Table 2-56 lists the memory-mapped registers for the CPU_DWT registers. All register offset addresses not listed in Table 2-56 should be considered as reserved locations and the register contents should not be modified.

Table 2-56. CPU_DWT Registers

Offset	Acronym	Register Name	Section
0h	CTRL	Provides configuration and status information for the DWT unit, and used to control features of the unit	Section 2.5.2.1
4h	CYCCNT	Shows or sets the value of the processor cycle counter, CYCCNT	Section 2.5.2.2
8h	CPICNT	CPI Count Register	Section 2.5.2.3
Ch	EXCCNT	Counts the total cycles spent in exception processing	Section 2.5.2.4
10h	SLEEPCNT	Sleep Count Register	Section 2.5.2.5
14h	LSUCNT	Increments on the additional cycles required to execute all load or store instructions	Section 2.5.2.6
18h	FOLDCNT	Increments on the additional cycles required to execute all load or store instructions	Section 2.5.2.7
1Ch	PCSR	Program Counter Sample Register	Section 2.5.2.8
28h	FUNCTION0	Controls the operation of watchpoint comparator 0	Section 2.5.2.9
38h	FUNCTION1	Controls the operation of watchpoint comparator 1	Section 2.5.2.10
48h	FUNCTION2	Controls the operation of watchpoint comparator 2	Section 2.5.2.11
58h	FUNCTION3	Controls the operation of watchpoint comparator 3	Section 2.5.2.12
FBCh	DEVARCH	Provides CoreSight discovery information for the DWT	Section 2.5.2.13
FCCh	DEVTYPE	Provides CoreSight discovery information for the DWT	Section 2.5.2.14
FD0h	PIDR4	Provides CoreSight discovery information for the DWT	Section 2.5.2.15
FD4h	PIDR5	Provides CoreSight discovery information for the DWT	Section 2.5.2.16
FD8h	PIDR6	Provides CoreSight discovery information for the DWT	Section 2.5.2.17
FDCh	PIDR7	Provides CoreSight discovery information for the DWT	Section 2.5.2.18
FE0h	PIDR0	Provides CoreSight discovery information for the DWT	Section 2.5.2.19
FE4h	PIDR1	Provides CoreSight discovery information for the DWT	Section 2.5.2.20
FE8h	PIDR2	Provides CoreSight discovery information for the DWT	Section 2.5.2.21
FECh	PIDR3	Provides CoreSight discovery information for the DWT	Section 2.5.2.22
FF0h	CIDR0	Provides CoreSight discovery information for the DWT	Section 2.5.2.23
FF4h	CIDR1	Provides CoreSight discovery information for the DWT	Section 2.5.2.24
FF8h	CIDR2	Provides CoreSight discovery information for the DWT	Section 2.5.2.25
FFCh	CIDR3	Provides CoreSight discovery information for the DWT	Section 2.5.2.26

Complex bit access types are encoded to fit into small table cells. Table 2-57 shows the codes that are used for access types in this section.

Table 2-57. CPU_DWT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.2.1 CTRL Register (Offset = 0h) [Reset = 00000000h]

CTRL is shown in Table 2-58.

Return to the Summary Table.

Provides configuration and status information for the DWT unit, and used to control features of the unit

Table 2-58. CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	NUMCOMP	R	0h	Number of DWT comparators implemented
27	NOTRCPKT	R	0h	Indicates whether the implementation does not support trace
26	NOEXTTRIG	R	0h	Reserved, RAZ
25	NOCYCCNT	R	0h	Indicates whether the implementation does not include a cycle counter
24	NOPRFCNT	R	0h	Indicates whether the implementation does not include the profiling counters
23	CYCDISS	R	0h	Controls whether the cycle counter is disabled in Secure state
22	CYCEVTENA	R	0h	Enables Event Counter packet generation on POSTCNT underflow
21	FOLDEVTENA	R	0h	Enables DWT_FOLDCNT counter
20	LSUEVTENA	R	0h	Enables DWT_LSUCNT counter
19	SLEEPEVTENA	R	0h	Enable DWT_SLEEPCNT counter
18	EXCEVTENA	R	0h	Enables DWT_EXCCNT counter
17	CPIEVTENA	R	0h	Enables DWT_CPICNT counter
16	EXTTRCENA	R	0h	Enables generation of Exception Trace packets
15-13	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
12	PCSAMPLENA	R	0h	Enables use of POSTCNT counter as a timer for Periodic PC Sample packet generation
11-10	SYNCTAP	R	0h	Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the Synchronization packet rate
9	CYCTAP	R	0h	Selects the position of the POSTCNT tap on the CYCCNT counter
8-5	POSTINIT	R	0h	Initial value for the POSTCNT counter
4-1	POSTPRESET	R	0h	Reload value for the POSTCNT counter
0	CYCCNTENA	R	0h	Enables CYCCNT

2.5.2.2 CYCCNT Register (Offset = 4h) [Reset = 00000000h]

CYCCNT is shown in Table 2-59.

Return to the Summary Table.

Shows or sets the value of the processor cycle counter, CYCCNT

Table 2-59. CYCCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CYCCNT	R/W	0h	Increments one on each processor clock cycle when DWT_CTRL.CYCCNTENA == 1 and DEMCR.TRCENA == 1. On overflow, CYCCNT wraps to zero

2.5.2.3 CPICNT Register (Offset = 8h) [Reset = 00000000h]

CPICNT is shown in Table 2-60.

Return to the Summary Table.

CPI Count Register

Table 2-60. CPICNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	CPICNT	R/W	0h	Counts one on each cycle when all of the following are true: - DWT_CTRL.CPIEVTENA == 1 and DEMCR.TRCENA == 1. - No instruction is executed. - No load-store operation is in progress, see DWT_LSUCNT. - No exception-entry or exception-exit operation is in progress, see DWT_EXCCNT. - The PE is not in a power saving mode, see DWT_SLEEPCNT. - Either SecureNoninvasiveDebugAllowed() == TRUE, or the PE is in Non-secure state and NoninvasiveDebugAllowed() == TRUE.

2.5.2.4 EXCCNT Register (Offset = Ch) [Reset = 00000000h]

EXCCNT is shown in Table 2-61.

Return to the Summary Table.

Counts the total cycles spent in exception processing

Table 2-61. EXCCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	EXCCNT	R/W	0h	Counts one on each cycle when all of the following are true: - DWT_CTRL.EXCEVTENA == 1 and DEMCR.TRCENA == 1. - No instruction is executed, see DWT_CPICNT. - An exception-entry or exception-exit related operation is in progress. - Either SecureNoninvasiveDebugAllowed() == TRUE, or NS-Req for the operation is set to Non-secure and NoninvasiveDebugAllowed() == TRUE.

2.5.2.5 SLEEPCNT Register (Offset = 10h) [Reset = 00000000h]

SLEEPCNT is shown in Table 2-62.

Return to the Summary Table.

Sleep Count Register

Table 2-62. SLEEPCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	SLEEPCNT	R/W	0h	Counts one on each cycle when all of the following are true: - DWT_CTRL.SLEEPEVTENA == 1 and DEMCR.TRCENA == 1. - No instruction is executed, see DWT_CPICNT. - No load-store operation is in progress, see DWT_LSUCNT. - No exception-entry or exception-exit operation is in progress, see DWT_EXCCNT. - The PE is in a power saving mode. - Either SecureNoninvasiveDebugAllowed() == TRUE, or the PE is in Non-secure state and NoninvasiveDebugAllowed() == TRUE.

2.5.2.6 LSUCNT Register (Offset = 14h) [Reset = 00000000h]

LSUCNT is shown in Table 2-63.

Return to the Summary Table.

Increments on the additional cycles required to execute all load or store instructions

Table 2-63. LSUCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	LSUCNT	R/W	0h	Counts one on each cycle when all of the following are true: - DWT_CTRL.LSUEVTENA == 1 and DEMCR.TRCENA == 1. - No instruction is executed, see DWT_CPICNT. - No exception-entry or exception-exit operation is in progress, see DWT_EXCCNT. - A load-store operation is in progress. - Either SecureNoninvasiveDebugAllowed() == TRUE, or NS-Req for the operation is set to Non-secure and NoninvasiveDebugAllowed() == TRUE.

2.5.2.7 FOLDCNT Register (Offset = 18h) [Reset = 00000000h]

FOLDCNT is shown in Table 2-64.

Return to the Summary Table.

Increments on the additional cycles required to execute all load or store instructions

Table 2-64. FOLDCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	FOLDCNT	R/W	0h	Counts on each cycle when all of the following are true: - DWT_CTRL.FOLDEVTENA == 1 and DEMCR.TRCENA == 1. - At least two instructions are executed, see DWT_CPICNT. - Either SecureNoninvasiveDebugAllowed() == TRUE, or the PE is in Non-secure state and NoninvasiveDebugAllowed() == TRUE. The counter is incremented by the number of instructions executed, minus one

2.5.2.8 PCSR Register (Offset = 1Ch) [Reset = 00000000h]

PCSR is shown in Table 2-65.

Return to the Summary Table.

Program Counter Sample Register

Table 2-65. PCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	EIASAMPLE	R	0h	The possible values of this field are: 0xFFFFFFFF One of the following is true: - The PE is halted in Debug state. - The Security Extension is implemented, the sampled instruction was executed in Secure state, and SecureNoninvasiveDebugAllowed() == FALSE. - NoninvasiveDebugAllowed() == FALSE. - DEMCR.TRCENA == 0. - The address of a recently-executed instruction is not available. Not 0xFFFFFFFF Instruction address of a recently executed instruction. Bit [0] of the sample instruction address is 0.

2.5.2.9 FUNCTION0 Register (Offset = 28h) [Reset = 00000000h]

FUNCTION0 is shown in Table 2-66.

Return to the Summary Table.

Controls the operation of watchpoint comparator 0

Table 2-66. FUNCTION0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	ID	R	0h	Identifies the capabilities for MATCH for comparator *n
26-25	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
24	MATCHED	R	0h	Set to 1 when the comparator matches
23-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-10	DATAVSIZE	R/W	0h	Defines the size of the object being watched for by Data Value and Data Address comparators
9-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5-4	ACTION	R/W	0h	Defines the action on a match. This field is ignored and the comparator generates no actions if it is disabled by MATCH
3-0	MATCH	R/W	0h	Controls the type of match generated by this comparator

2.5.2.10 FUNCTION1 Register (Offset = 38h) [Reset = 00000000h]

FUNCTION1 is shown in Table 2-67.

Return to the Summary Table.

Controls the operation of watchpoint comparator 1

Table 2-67. FUNCTION1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	ID	R	0h	Identifies the capabilities for MATCH for comparator *n
26-25	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
24	MATCHED	R	0h	Set to 1 when the comparator matches
23-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-10	DATAVSIZE	R/W	0h	Defines the size of the object being watched for by Data Value and Data Address comparators
9-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5-4	ACTION	R/W	0h	Defines the action on a match. This field is ignored and the comparator generates no actions if it is disabled by MATCH
3-0	MATCH	R/W	0h	Controls the type of match generated by this comparator

2.5.2.11 FUNCTION2 Register (Offset = 48h) [Reset = 00000000h]

FUNCTION2 is shown in Table 2-68.

Return to the Summary Table.

Controls the operation of watchpoint comparator 2

Table 2-68. FUNCTION2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	ID	R	0h	Identifies the capabilities for MATCH for comparator *n
26-25	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
24	MATCHED	R	0h	Set to 1 when the comparator matches
23-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-10	DATAVSIZE	R/W	0h	Defines the size of the object being watched for by Data Value and Data Address comparators
9-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5-4	ACTION	R/W	0h	Defines the action on a match. This field is ignored and the comparator generates no actions if it is disabled by MATCH
3-0	MATCH	R/W	0h	Controls the type of match generated by this comparator

2.5.2.12 FUNCTION3 Register (Offset = 58h) [Reset = 00000000h]

FUNCTION3 is shown in Table 2-69.

Return to the Summary Table.

Controls the operation of watchpoint comparator 3

Table 2-69. FUNCTION3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	ID	R	0h	Identifies the capabilities for MATCH for comparator *n
26-25	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
24	MATCHED	R	0h	Set to 1 when the comparator matches
23-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-10	DATAVSIZE	R/W	0h	Defines the size of the object being watched for by Data Value and Data Address comparators
9-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5-4	ACTION	R/W	0h	Defines the action on a match. This field is ignored and the comparator generates no actions if it is disabled by MATCH
3-0	MATCH	R/W	0h	Controls the type of match generated by this comparator

2.5.2.13 DEVARCH Register (Offset = FBCh) [Reset = 00000000h]

DEVARCH is shown in Table 2-70.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-70. DEVARCH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-21	ARCHITECT	R	0h	Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code.
20	PRESENT	R	0h	Defines that the DEVARCH register is present
19-16	REVISION	R	0h	Defines the architecture revision of the component
15-12	ARCHVER	R	0h	Defines the architecture version of the component
11-0	ARCHPART	R	0h	Defines the architecture of the component

2.5.2.14 DEVTYPE Register (Offset = FCCh) [Reset = 00000000h]

DEVTYPE is shown in Table 2-71.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-71. DEVTYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	SUB	R	0h	Component sub-type
3-0	MAJOR	R	0h	Component major type

2.5.2.15 PIDR4 Register (Offset = FD0h) [Reset = 00000000h]

PIDR4 is shown in Table 2-72.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-72. PIDR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	SIZE	R	0h	See CoreSight Architecture Specification
3-0	DES_2	R	0h	See CoreSight Architecture Specification

2.5.2.16 PIDR5 Register (Offset = FD4h) [Reset = 00000000h]

PIDR5 is shown in Table 2-73.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-73. PIDR5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.2.17 PIDR6 Register (Offset = FD8h) [Reset = 00000000h]

PIDR6 is shown in Table 2-74.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-74. PIDR6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.2.18 PIDR7 Register (Offset = FDCh) [Reset = 00000000h]

PIDR7 is shown in Table 2-75.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-75. PIDR7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.2.19 PIDR0 Register (Offset = FE0h) [Reset = 00000000h]

PIDR0 is shown in Table 2-76.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-76. PIDR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PART_0	R	0h	See CoreSight Architecture Specification

2.5.2.20 PIDR1 Register (Offset = FE4h) [Reset = 00000000h]

PIDR1 is shown in Table 2-77.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-77. PIDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	DES_0	R	0h	See CoreSight Architecture Specification
3-0	PART_1	R	0h	See CoreSight Architecture Specification

2.5.2.21 PIDR2 Register (Offset = FE8h) [Reset = 00000000h]

PIDR2 is shown in Table 2-78.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-78. PIDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	REVISION	R	0h	See CoreSight Architecture Specification
3	JEDEC	R	0h	See CoreSight Architecture Specification
2-0	DES_1	R	0h	See CoreSight Architecture Specification

2.5.2.22 PIDR3 Register (Offset = FECh) [Reset = 00000000h]

PIDR3 is shown in Table 2-79.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-79. PIDR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	REVAND	R	0h	See CoreSight Architecture Specification
3-0	CMOD	R	0h	See CoreSight Architecture Specification

2.5.2.23 CIDR0 Register (Offset = FF0h) [Reset = 00000000h]

CIDR0 is shown in Table 2-80.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-80. CIDR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_0	R	0h	See CoreSight Architecture Specification

2.5.2.24 CIDR1 Register (Offset = FF4h) [Reset = 00000000h]

CIDR1 is shown in Table 2-81.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-81. CIDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	CLASS	R	0h	See CoreSight Architecture Specification
3-0	PRMBL_1	R	0h	See CoreSight Architecture Specification

2.5.2.25 CIDR2 Register (Offset = FF8h) [Reset = 00000000h]

CIDR2 is shown in Table 2-82.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-82. CIDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_2	R	0h	See CoreSight Architecture Specification

2.5.2.26 CIDR3 Register (Offset = FFCh) [Reset = 00000000h]

CIDR3 is shown in Table 2-83.

Return to the Summary Table.

Provides CoreSight discovery information for the DWT

Table 2-83. CIDR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	PRMBL_3	R	0h	See CoreSight Architecture Specification

2.5.3 CPU_SYSTICK Registers

Table 2-84 lists the memory-mapped registers for the CPU_SYSTICK registers. All register offset addresses not listed in Table 2-84 should be considered as reserved locations and the register contents should not be modified.

Table 2-84. CPU_SYSTICK Registers

Offset	Acronym	Register Name	Section
0h	CSR	Controls the SysTick timer and provides status data `FTSSS	Section 2.5.3.1
4h	RVR	Provides access SysTick timer counter reload value `FTSSS	Section 2.5.3.2
8h	CVR	Reads or clears the SysTick timer current counter value `FTSSS	Section 2.5.3.3
Ch	CALIB	Reads the SysTick timer calibration value and parameters `FTSSS	Section 2.5.3.4

Complex bit access types are encoded to fit into small table cells. Table 2-85 shows the codes that are used for access types in this section.

Table 2-85. CPU_SYSTICK Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.3.1 CSR Register (Offset = 0h) [Reset = 00000000h]

CSR is shown in Table 2-86.

Return to the Summary Table.

Controls the SysTick timer and provides status data `FTSSS

Table 2-86. CSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
16	COUNTFLAG	R/W	0h	Indicates whether the counter has counted to zero since the last read of this register
15-3	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
2	CLKSOURCE	R/W	0h	Indicates the SysTick clock source
1	TICKINT	R/W	0h	Indicates whether counting to 0 causes the status of the SysTick exception to change to pending
0	ENABLE	R/W	0h	Indicates the enabled status of the SysTick counter

2.5.3.2 RVR Register (Offset = 4h) [Reset = 00000000h]

RVR is shown in Table 2-87.

Return to the Summary Table.

Provides access SysTick timer counter reload value `FTSSS

Table 2-87. RVR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23-0	RELOAD	R/W	0h	The value to load into the SYST_CVR `FTSSS when the counter reaches 0

2.5.3.3 CVR Register (Offset = 8h) [Reset = 00000000h]

CVR is shown in Table 2-88.

Return to the Summary Table.

Reads or clears the SysTick timer current counter value `FTSSS

Table 2-88. CVR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23-0	CURRENT	W	0h	Writing any value clears the SysTick timer counter `FTSSS to zero

2.5.3.4 CALIB Register (Offset = Ch) [Reset = 00000000h]

CALIB is shown in Table 2-89.

Return to the Summary Table.

Reads the SysTick timer calibration value and parameters `FTSSS

Table 2-89. CALIB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NOREF	R	0h	Indicates whether the IMPLEMENTATION DEFINED reference clock is implemented
30	SKEW	R	0h	Indicates whether the 10ms calibration value is exact
29-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23-0	TENMS	R	0h	Optionally, holds a reload value to be used for 10ms (100Hz) timing, subject to system clock skew errors. If this field is zero, the calibration value is not known

2.5.4 CPU_NVIC Registers

Table 2-90 lists the memory-mapped registers for the CPU_NVIC registers. All register offset addresses not listed in Table 2-90 should be considered as reserved locations and the register contents should not be modified.

Table 2-90. CPU_NVIC Registers

Offset	Acronym	Register Name	Section
0h	ISER0	Enables or reads the enabled state of each group of 32 interrupts	Section 2.5.4.1
4h	ISER1	Enables or reads the enabled state of each group of 32 interrupts	Section 2.5.4.2
80h	ICER0	Clears or reads the enabled state of each group of 32 interrupts	Section 2.5.4.3
84h	ICER1	Clears or reads the enabled state of each group of 32 interrupts	Section 2.5.4.4
100h	ISPR0	Enables or reads the pending state of each group of 32 interrupts	Section 2.5.4.5
104h	ISPR1	Enables or reads the pending state of each group of 32 interrupts	Section 2.5.4.6
180h	ICPR0	Clears or reads the pending state of each group of 32 interrupts	Section 2.5.4.7
184h	ICPR1	Clears or reads the pending state of each group of 32 interrupts	Section 2.5.4.8
200h	IABR0	For each group of 32 interrupts, shows the active state of each interrupt	Section 2.5.4.9
204h	IABR1	For each group of 32 interrupts, shows the active state of each interrupt	Section 2.5.4.10
280h	ITNS0	For each group of 32 interrupts, determines whether each interrupt targets Non-secure or Secure state	Section 2.5.4.11
284h	ITNS1	For each group of 32 interrupts, determines whether each interrupt targets Non-secure or Secure state	Section 2.5.4.12
300h	IPR0	Sets or reads interrupt priorities	Section 2.5.4.13
304h	IPR1	Sets or reads interrupt priorities	Section 2.5.4.14
308h	IPR2	Sets or reads interrupt priorities	Section 2.5.4.15
30Ch	IPR3	Sets or reads interrupt priorities	Section 2.5.4.16
310h	IPR4	Sets or reads interrupt priorities	Section 2.5.4.17
314h	IPR5	Sets or reads interrupt priorities	Section 2.5.4.18
318h	IPR6	Sets or reads interrupt priorities	Section 2.5.4.19
31Ch	IPR7	Sets or reads interrupt priorities	Section 2.5.4.20
320h	IPR8	Sets or reads interrupt priorities	Section 2.5.4.21
324h	IPR9	Sets or reads interrupt priorities	Section 2.5.4.22
328h	IPR10	Sets or reads interrupt priorities	Section 2.5.4.23
32Ch	IPR11	Sets or reads interrupt priorities	Section 2.5.4.24

Complex bit access types are encoded to fit into small table cells. Table 2-91 shows the codes that are used for access types in this section.

Table 2-91. CPU_NVIC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.4.1 ISER0 Register (Offset = 0h) [Reset = 00000000h]

ISER0 is shown in Table 2-92.

Return to the Summary Table.

Enables or reads the enabled state of each group of 32 interrupts

Table 2-92. ISER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R	0h	For SETENA[m] in NVIC_ISER*n, indicates whether interrupt 32*n + m is enabled

2.5.4.2 ISER1 Register (Offset = 4h) [Reset = 00000000h]

ISER1 is shown in Table 2-93.

Return to the Summary Table.

Enables or reads the enabled state of each group of 32 interrupts

Table 2-93. ISER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	SETENA	R	0h	For SETENA[m] in NVIC_ISER*n, indicates whether interrupt 32*n + m is enabled

2.5.4.3 ICER0 Register (Offset = 80h) [Reset = 00000000h]

ICER0 is shown in Table 2-94.

Return to the Summary Table.

Clears or reads the enabled state of each group of 32 interrupts

Table 2-94. ICER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRENA	R	0h	For CLRENA[m] in NVIC_ICER*n, indicates whether interrupt 32*n + m is enabled

2.5.4.4 ICER1 Register (Offset = 84h) [Reset = 00000000h]

ICER1 is shown in Table 2-95.

Return to the Summary Table.

Clears or reads the enabled state of each group of 32 interrupts

Table 2-95. ICER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	CLRENA	R	0h	For CLRENA[m] in NVIC_ICER*n, indicates whether interrupt 32*n + m is enabled

2.5.4.5 ISPR0 Register (Offset = 100h) [Reset = 00000000h]

ISPR0 is shown in Table 2-96.

Return to the Summary Table.

Enables or reads the pending state of each group of 32 interrupts

Table 2-96. ISPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETPEND	R	0h	For SETPEND[m] in NVIC_ISPR*n, indicates whether interrupt 32*n + m is pending

2.5.4.6 ISPR1 Register (Offset = 104h) [Reset = 00000000h]

ISPR1 is shown in Table 2-97.

Return to the Summary Table.

Enables or reads the pending state of each group of 32 interrupts

Table 2-97. ISPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	SETPEND	R	0h	For SETPEND[m] in NVIC_ISPR*n, indicates whether interrupt 32*n + m is pending

2.5.4.7 ICPR0 Register (Offset = 180h) [Reset = 00000000h]

ICPR0 is shown in Table 2-98.

Return to the Summary Table.

Clears or reads the pending state of each group of 32 interrupts

Table 2-98. ICPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R	0h	For CLRPEND[m] in NVIC_ICPR*n, indicates whether interrupt 32*n + m is pending

2.5.4.8 ICPR1 Register (Offset = 184h) [Reset = 00000000h]

ICPR1 is shown in Table 2-99.

Return to the Summary Table.

Clears or reads the pending state of each group of 32 interrupts

Table 2-99. ICPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	CLRPEND	R	0h	For CLRPEND[m] in NVIC_ICPR*n, indicates whether interrupt 32*n + m is pending

2.5.4.9 IABR0 Register (Offset = 200h) [Reset = 00000000h]

IABR0 is shown in Table 2-100.

Return to the Summary Table.

For each group of 32 interrupts, shows the active state of each interrupt

Table 2-100. IABR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ACTIVE	R	0h	For ACTIVE[m] in NVIC_IABR*n, indicates the active state for interrupt 32*n+m

2.5.4.10 IABR1 Register (Offset = 204h) [Reset = 00000000h]

IABR1 is shown in Table 2-101.

Return to the Summary Table.

For each group of 32 interrupts, shows the active state of each interrupt

Table 2-101. IABR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	ACTIVE	R	0h	For ACTIVE[m] in NVIC_IABR*n, indicates the active state for interrupt 32*n+m

2.5.4.11 ITNS0 Register (Offset = 280h) [Reset = 00000000h]

ITNS0 is shown in Table 2-102.

Return to the Summary Table.

For each group of 32 interrupts, determines whether each interrupt targets Non-secure or Secure state

Table 2-102. ITNS0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ITNS	R/W	0h	For ITNS[m] in NVIC_ITNS*n, the target Security state for interrupt 32*n+m

2.5.4.12 ITNS1 Register (Offset = 284h) [Reset = 00000000h]

ITNS1 is shown in Table 2-103.

Return to the Summary Table.

For each group of 32 interrupts, determines whether each interrupt targets Non-secure or Secure state

Table 2-103. ITNS1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-0	ITNS	R/W	0h	For ITNS[m] in NVIC_ITNS*n, the target Security state for interrupt 32*n+m

2.5.4.13 IPR0 Register (Offset = 300h) [Reset = 00000000h]

IPR0 is shown in Table 2-104.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-104. IPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*0, the priority of interrupt number 4*0+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*0, the priority of interrupt number 4*0+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*0, the priority of interrupt number 4*0+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*0, the priority of interrupt number 4*0+0, or is RES0 if the PE does not implement this interrupt

2.5.4.14 IPR1 Register (Offset = 304h) [Reset = 00000000h]

IPR1 is shown in Table 2-105.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-105. IPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*1, the priority of interrupt number 4*1+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*1, the priority of interrupt number 4*1+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*1, the priority of interrupt number 4*1+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*1, the priority of interrupt number 4*1+0, or is RES0 if the PE does not implement this interrupt

2.5.4.15 IPR2 Register (Offset = 308h) [Reset = 00000000h]

IPR2 is shown in Table 2-106.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-106. IPR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*2, the priority of interrupt number 4*2+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*2, the priority of interrupt number 4*2+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*2, the priority of interrupt number 4*2+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*2, the priority of interrupt number 4*2+0, or is RES0 if the PE does not implement this interrupt

2.5.4.16 IPR3 Register (Offset = 30Ch) [Reset = 00000000h]

IPR3 is shown in Table 2-107.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-107. IPR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*3, the priority of interrupt number 4*3+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*3, the priority of interrupt number 4*3+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*3, the priority of interrupt number 4*3+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*3, the priority of interrupt number 4*3+0, or is RES0 if the PE does not implement this interrupt

2.5.4.17 IPR4 Register (Offset = 310h) [Reset = 00000000h]

IPR4 is shown in Table 2-108.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-108. IPR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*4, the priority of interrupt number 4*4+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*4, the priority of interrupt number 4*4+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*4, the priority of interrupt number 4*4+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*4, the priority of interrupt number 4*4+0, or is RES0 if the PE does not implement this interrupt

2.5.4.18 IPR5 Register (Offset = 314h) [Reset = 00000000h]

IPR5 is shown in Table 2-109.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-109. IPR5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*5, the priority of interrupt number 4*5+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*5, the priority of interrupt number 4*5+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*5, the priority of interrupt number 4*5+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*5, the priority of interrupt number 4*5+0, or is RES0 if the PE does not implement this interrupt

2.5.4.19 IPR6 Register (Offset = 318h) [Reset = 00000000h]

IPR6 is shown in Table 2-110.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-110. IPR6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*6, the priority of interrupt number 4*6+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*6, the priority of interrupt number 4*6+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*6, the priority of interrupt number 4*6+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*6, the priority of interrupt number 4*6+0, or is RES0 if the PE does not implement this interrupt

2.5.4.20 IPR7 Register (Offset = 31Ch) [Reset = 00000000h]

IPR7 is shown in Table 2-111.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-111. IPR7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*7, the priority of interrupt number 4*7+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*7, the priority of interrupt number 4*7+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*7, the priority of interrupt number 4*7+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*7, the priority of interrupt number 4*7+0, or is RES0 if the PE does not implement this interrupt

2.5.4.21 IPR8 Register (Offset = 320h) [Reset = 00000000h]

IPR8 is shown in Table 2-112.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-112. IPR8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*8, the priority of interrupt number 4*8+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*8, the priority of interrupt number 4*8+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*8, the priority of interrupt number 4*8+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*8, the priority of interrupt number 4*8+0, or is RES0 if the PE does not implement this interrupt

2.5.4.22 IPR9 Register (Offset = 324h) [Reset = 00000000h]

IPR9 is shown in Table 2-113.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-113. IPR9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*9, the priority of interrupt number 4*9+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*9, the priority of interrupt number 4*9+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*9, the priority of interrupt number 4*9+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*9, the priority of interrupt number 4*9+0, or is RES0 if the PE does not implement this interrupt

2.5.4.23 IPR10 Register (Offset = 328h) [Reset = 00000000h]

IPR10 is shown in Table 2-114.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-114. IPR10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*10, the priority of interrupt number 4*10+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*10, the priority of interrupt number 4*10+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*10, the priority of interrupt number 4*10+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*10, the priority of interrupt number 4*10+0, or is RES0 if the PE does not implement this interrupt

2.5.4.24 IPR11 Register (Offset = 32Ch) [Reset = 00000000h]

IPR11 is shown in Table 2-115.

Return to the Summary Table.

Sets or reads interrupt priorities

Table 2-115. IPR11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_N3	R/W	0h	For register NVIC_IPR*11, the priority of interrupt number 4*11+3, or is RES0 if the PE does not implement this interrupt
23-16	PRI_N2	R/W	0h	For register NVIC_IPR*11, the priority of interrupt number 4*11+2, or is RES0 if the PE does not implement this interrupt
15-8	PRI_N1	R/W	0h	For register NVIC_IPR*11, the priority of interrupt number 4*11+1, or is RES0 if the PE does not implement this interrupt
7-0	PRI_N0	R/W	0h	For register NVIC_IPR*11, the priority of interrupt number 4*11+0, or is RES0 if the PE does not implement this interrupt

2.5.5 CPU_SCS Registers

Table 2-116 lists the memory-mapped registers for the CPU_SCS registers. All register offset addresses not listed in Table 2-116 should be considered as reserved locations and the register contents should not be modified.

Table 2-116. CPU_SCS Registers

Offset	Acronym	Register Name	Section
0h	CPUID	CPUID Base	Section 2.5.5.1
4h	ICSR	Interrupt Control State	Section 2.5.5.2
8h	VTOR	Vector Table Offset	Section 2.5.5.3
Ch	AIRCR	Application Interrupt/Reset Control	Section 2.5.5.4
10h	SCR	System Control	Section 2.5.5.5
14h	CCR	Configuration Control	Section 2.5.5.6
18h	SHPR1	System Handlers 4-7 Priority	Section 2.5.5.7
1Ch	SHPR2	System Handlers 8-11 Priority	Section 2.5.5.8
20h	SHPR3	System Handlers 12-15 Priority	Section 2.5.5.9
24h	SHCSR	System Handler Control and State	Section 2.5.5.10
28h	CFSR	Configurable Fault Status	Section 2.5.5.11
2Ch	HFSR	Hard Fault Status	Section 2.5.5.12
30h	DFSR	Debug Fault Status	Section 2.5.5.13
34h	MMFAR	Mem Manage Fault Address	Section 2.5.5.14
38h	BFAR	Bus Fault Address	Section 2.5.5.15
3Ch	AFSR	Auxiliary Fault Status	Section 2.5.5.16
40h	ID_PFR0	Processor Feature 0	Section 2.5.5.17
44h	ID_PFR1	Processor Feature 1	Section 2.5.5.18
48h	ID_DFR0	Debug Feature 0	Section 2.5.5.19
4Ch	ID_AFR0	Auxiliary Feature 0	Section 2.5.5.20
50h	ID_MMFR0	Memory Model Feature 0	Section 2.5.5.21
54h	ID_MMFR1	Memory Model Feature 1	Section 2.5.5.22
58h	ID_MMFR2	Memory Model Feature 2	Section 2.5.5.23
5Ch	ID_MMFR3	Memory Model Feature 3	Section 2.5.5.24
60h	ID_ISAR0	ISA Feature 0	Section 2.5.5.25
64h	ID_ISAR1	ISA Feature 1	Section 2.5.5.26
68h	ID_ISAR2	ISA Feature 2	Section 2.5.5.27
6Ch	ID_ISAR3	ISA Feature 3	Section 2.5.5.28
70h	ID_ISAR4	ISA Feature 4	Section 2.5.5.29

Complex bit access types are encoded to fit into small table cells. Table 2-117 shows the codes that are used for access types in this section.

Table 2-117. CPU_SCS Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

2.5.5.1 CPUID Register (Offset = 0h) [Reset = 410FD214h]

CPUID is shown in Table 2-118.

Return to the Summary Table.

CPUID Base
This register determines the ID number of the processor core, the version number of the processor core and the implementation details of the processor core.

Table 2-118. CPUID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	IMPLEMENTER	R	41h	Implementor code.
23-20	VARIANT	R	0h	Implementation defined variant number.
19-16	CONSTANT	R	Fh	Reads as 0xF
15-4	PARTNO	R	D21h	Number of processor within family.
3-0	REVISION	R	4h	Implementation defined revision number.

2.5.5.2 ICSR Register (Offset = 4h) [Reset = 00000000h]

ICSR is shown in Table 2-119.

Return to the Summary Table.

Interrupt Control State
This register is used to set a pending Non-Maskable Interrupt (NMI), set or clear a pending SVC, set or clear a pending SysTick, check for pending exceptions, check the vector number of the highest priority pended exception, and check the vector number of the active exception.

Table 2-119. ICSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NMIPENDSET	R/W	0h	Set pending NMI bit. Setting this bit pends and activates an NMI. Because NMI is the highest-priority interrupt, it takes effect as soon as it registers. 0: No action 1: Set pending NMI
30	PENDNMICLR	R/W	0h	Pend NMI clear. Allows the NMI exception pending state to be cleared. 0x0 No effect. 0x1 Clear pending status.
29	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
28	PENDSVSET	R/W	0h	Set pending pendSV bit. 0: No action 1: Set pending PendSV
27	PENDSVCLR	W	X	Clear pending pendSV bit 0: No action 1: Clear pending pendSV
26	PENDSTSET	R/W	0h	Set a pending SysTick bit. 0: No action 1: Set pending SysTick
25	PENDSTCLR	W	X	Clear pending SysTick bit 0: No action 1: Clear pending SysTick
24	STTNS	R	0h	SysTick Targets Non-secure. Controls whether in a single SysTick implementation, the SysTick is Secure or Non-secure. 0x0 SysTick is Secure. 0x1 SysTick is Non-secure.
23	ISRPREEMPT	R	0h	This field can only be used at debug time. It indicates that a pending interrupt is to be taken in the next running cycle. If DHCSR.C_MASKINTS= 0, the interrupt is serviced. 0: A pending exception is not serviced. 1: A pending exception is serviced on exit from the debug halt state
22	ISRPENDING	R	0h	Interrupt pending flag. Excludes NMI and faults. 0x0: Interrupt not pending 0x1: Interrupt pending
21	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
20-12	VECTPENDING	R	0h	Pending ISR number field. This field contains the interrupt number of the highest priority pending ISR.
11	RETTOBASE	R	0h	Indicates whether there are preempted active exceptions: 0: There are preempted active exceptions to execute 1: There are no active exceptions, or the currently-executing exception is the only active exception.
10-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8-0	VECTACTIVE	R	0h	Active ISR number field. Reset clears this field.

2.5.5.3 VTOR Register (Offset = 8h) [Reset = 00000000h]

VTOR is shown in Table 2-120.

Return to the Summary Table.

Vector Table Offset
This register is used to relocated the vector table base address. The vector table base offset determines the offset from the bottom of the memory map. The two most significant bits and the seven least significant bits of the vector table base offset must be 0. The portion of vector table base offset that is allowed to change is TBLOFF.

Table 2-120. VTOR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	TBLOFF	R/W	0h	Bits 31 down to 7 of the vector table base offset.
6-0	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.4 AIRCR Register (Offset = Ch) [Reset = FA050000h]

AIRCR is shown in Table 2-121.

Return to the Summary Table.

Application Interrupt/Reset Control
This register is used to determine data endianness, clear all active state information for debug or to recover from a hard failure, execute a system reset, alter the priority grouping position (binary point).

Table 2-121. AIRCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	VECTKEY	R/W	FA05h	Register key. Writing to this register (AIRCR) requires 0x05FA in VECTKEY. Otherwise the write value is ignored. Read always returns 0xFA05.
15	ENDIANESS	R	0h	Data endianness bit 0h = Little endian 1h = Big endian
14	PRIS	R	0h	Prioritize Secure exceptions. The value of this bit defines whether Secure exception priority boosting is enabled.
13	BFHFNMINS	R/W	0h	BusFault, HardFault, and NMI Non-secure enable. The value of this bit defines whether BusFault and NMI exceptions are Non-secure, and whether exceptions target the Non-secure HardFault exception 0x0 BusFault, HardFault, and NMI are Secure. 0x1 BusFault and NMI are Non-secure and exceptions can target Non-secure HardFault.
12-11	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
10-8	PRIGROUP	R/W	0h	Interrupt priority grouping field. This field is a binary point position indicator for creating subpriorities for exceptions that share the same pre-emption level. It divides the PRI_n field in the Interrupt Priority Registers (NVIC_IPR0, NVIC_IPR1,..., and NVIC_IPR8) into a pre-emption level and a subpriority level. The binary point is a left-of value. This means that the PRIGROUP value represents a point starting at the left of the Least Significant Bit (LSB). The lowest value might not be 0 depending on the number of bits allocated for priorities, and implementation choices.
7-4	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	SYSRESETREQS	R/W	0h	System reset request Secure only. The value of this bit defines whether the SYSRESETREQ bit is functional for Non-secure use
2	SYSRESETREQ	W	0h	Requests a warm reset. Setting this bit does not prevent Halting Debug from running.
1	VECTCLRACTIVE	W	0h	Clears all active state information for active NMI, fault, and interrupts. It is the responsibility of the application to reinitialize the stack. This bit is for returning to a known state during debug. The bit self-clears. IPSR is not cleared by this operation. So, if used by an application, it must only be used at the base level of activation, or within a system handler whose active bit can be set.
0	RESERVED	W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.5 SCR Register (Offset = 10h) [Reset = 00000000h]

SCR is shown in Table 2-122.

Return to the Summary Table.

System Control
This register is used for power-management functions, i.e., signaling to the system when the processor can enter a low power state, controlling how the processor enters and exits low power states.

Table 2-122. SCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4	SEVONPEND	R/W	0h	Send Event on Pending bit: 0: Only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded 1: Enabled events and all interrupts, including disabled interrupts, can wakeup the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction.
3	SLEEPDEEPS	R/W	0h	Sleep deep secure. This field controls whether the SLEEPDEEP bit is only accessible from the Secure state
2	SLEEPDEEP	R/W	0h	Controls whether the processor uses sleep or deep sleep as its low power mode 0h = Sleep 1h = Deep sleep
1	SLEEPONEXIT	R/W	0h	Sleep on exit when returning from Handler mode to Thread mode. Enables interrupt driven applications to avoid returning to empty main application. 0: Do not sleep when returning to thread mode 1: Sleep on ISR exit
0	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.6 CCR Register (Offset = 14h) [Reset = 00000001h]

CCR is shown in Table 2-123.

Return to the Summary Table.

Configuration Control
This register is used to enable NMI, HardFault and FAULTMASK to ignore bus fault, trap divide by zero and unaligned accesses, enable user access to the Software Trigger Interrupt Register (STIR), control entry to Thread Mode.

Table 2-123. CCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	BFHFNMIGN	R/W	0h	Enables handlers with priority -1 or -2 to ignore data BusFaults caused by load and store instructions. This applies to the HardFault, NMI, and FAULTMASK escalated handlers: 0: Data BusFaults caused by load and store instructions cause a lock-up 1: Data BusFaults caused by load and store instructions are ignored. Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect problems.
7-5	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4	DIV_0_TRP	R/W	0h	Enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0: 0: Do not trap divide by 0. In this mode, a divide by zero returns a quotient of 0. 1: Trap divide by 0. The relevant Usage Fault Status Register bit is CFSR.DIVBYZERO.
3	UNALIGN_TRP	R/W	0h	Enables unaligned access traps: 0: Do not trap unaligned halfword and word accesses 1: Trap unaligned halfword and word accesses. The relevant Usage Fault Status Register bit is CFSR.UNALIGNED. If this bit is set to 1, an unaligned access generates a UsageFault. Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the value in UNALIGN_TRP.
2	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	USERSETMPEND	R/W	0h	Enables unprivileged software access to STIR: 0: User code is not allowed to write to the Software Trigger Interrupt register (STIR). 1: User code can write the Software Trigger Interrupt register (STIR) to trigger (pend) a Main exception, which is associated with the Main stack pointer.
0	RESERVED	R	1h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.7 SHPR1 Register (Offset = 18h) [Reset = 00000000h]

SHPR1 is shown in Table 2-124.

Return to the Summary Table.

System Handlers 4-7 Priority
This register is used to prioritize the following system handlers: Memory manage, Bus fault, and Usage fault. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

Table 2-124. SHPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	PRI_6	R/W	0h	Priority of system handler 6. UsageFault
15-8	PRI_5	R/W	0h	Priority of system handler 5: BusFault
7-0	PRI_4	R/W	0h	Priority of system handler 4: MemManage

2.5.5.8 SHPR2 Register (Offset = 1Ch) [Reset = 00000000h]

SHPR2 is shown in Table 2-125.

Return to the Summary Table.

System Handlers 8-11 Priority
This register is used to prioritize the SVC handler. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

Table 2-125. SHPR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_11	R/W	0h	Priority of system handler 11. SVCall
23-0	RESERVED	R	0h	Reserved

2.5.5.9 SHPR3 Register (Offset = 20h) [Reset = 00000000h]

SHPR3 is shown in Table 2-126.

Return to the Summary Table.

System Handlers 12-15 Priority
This register is used to prioritize the following system handlers: SysTick, PendSV and Debug Monitor. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

Table 2-126. SHPR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_15	R/W	0h	Priority of system handler 15. SysTick exception
23-16	PRI_14	R/W	0h	Priority of system handler 14. Pend SV
15-8	RESERVED	R	0h	Reserved
7-0	PRI_12	R/W	0h	Priority of system handler 12. Debug Monitor

2.5.5.10 SHCSR Register (Offset = 24h) [Reset = 00000000h]

SHCSR is shown in Table 2-127.

Return to the Summary Table.

System Handler Control and State
This register is used to enable or disable the system handlers, determine the pending status of bus fault, mem manage fault, and SVC, determine the active status of the system handlers. If a fault condition occurs while its fault handler is disabled, the fault escalates to a Hard Fault.

Table 2-127. SHCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
21	HARDFAULTPENDED	R	0h	SecureFault exception pended state 0h = Exception is not active 1h = Exception is pending.
20	SECUREFAULTPENDED	R	0h	SecureFault exception pended state 0h = Exception is not active 1h = Exception is pending.
19	SECUREFAULTENA	R/W	0h	SecureFault exception enable. 0h = Exception disabled 1h = Exception enabled
18	USGFAULTENA	R/W	0h	Usage fault system handler enable 0h = Exception disabled 1h = Exception enabled
17	BUSFAULTENA	R/W	0h	Bus fault system handler enable 0h = Exception disabled 1h = Exception enabled
16	MEMFAULTENA	R/W	0h	MemManage fault system handler enable 0h = Exception disabled 1h = Exception enabled
15	SVCALLPENDED	R	0h	SVCall pending 0h = Exception is not active 1h = Exception is pending.
14	BUSFAULTPENDED	R	0h	BusFault pending 0h = Exception is not active 1h = Exception is pending.
13	MEMFAULTPENDED	R	0h	MemManage exception pending 0h = Exception is not active 1h = Exception is pending.
12	USGFAULTPENDED	R	0h	Usage fault pending 0h = Exception is not active 1h = Exception is pending.
11	SYSTICKACT	R	0h	SysTick active flag. 0x0: Not active 0x1: Active 0h = Exception is not active 1h = Exception is active
10	PENDSVACT	R	0h	PendSV active 0x0: Not active 0x1: Active
9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	MONITORACT	R	0h	Debug monitor active 0h = Exception is not active 1h = Exception is active

Table 2-127. SHCSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	SVCALLACT	R	0h	SVCall active 0h = Exception is not active 1h = Exception is active
6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5	NMIACT	R	0h	NMI exception active state 0h = Exception is not active 1h = Exception is active
4	SECUREFAULTACT	R	0h	SecureFault exception active state 0h = Exception is not active 1h = Exception is active
3	USGFAULTACT	R	0h	UsageFault exception active 0h = Exception is not active 1h = Exception is active
2	HARDFAULTACT	R	0h	HardFault exception active state. Indicates and allows limited modification of the active state of the HardFault exception for the selected Security state 0h = Exception is not active 1h = Exception is active
1	BUSFAULTACT	R	0h	BusFault exception active 0h = Exception is not active 1h = Exception is active
0	MEMFAULTACT	R	0h	MemManage exception active 0h = Exception is not active 1h = Exception is active

2.5.5.11 CFSR Register (Offset = 28h) [Reset = 00000000h]

CFSR is shown in Table 2-128.

Return to the Summary Table.

Configurable Fault Status
This register is used to obtain information about local faults. These registers include three subsections: The first byte is Memory Manage Fault Status Register (MMFSR). The second byte is Bus Fault Status Register (BFSR). The higher half-word is Usage Fault Status Register (UFSR). The flags in these registers indicate the causes of local faults. Multiple flags can be set if more than one fault occurs. These register are read/write-clear. This means that they can be read normally, but writing a 1 to any bit clears that bit.
The CFSR is byte accessible. CFSR or its subregisters can be accessed as follows:
The following accesses are possible to the CFSR register:
- access the complete register with a word access to 0xE000ED28.
- access the MMFSR with a byte access to 0xE000ED28
- access the MMFSR and BFSR with a halfword access to 0xE000ED28
- access the BFSR with a byte access to 0xE000ED29
- access the UFSR with a halfword access to 0xE000ED2A.

Table 2-128. CFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
25	DIVBYZERO	R/W	0h	When CCR.DIV_0_TRP (see Configuration Control Register on page 8-26) is enabled and an SDIV or UDIV instruction is used with a divisor of 0, this fault occurs The instruction is executed and the return PC points to it. If CCR.DIV_0_TRP is not set, then the divide returns a quotient of 0.
24	UNALIGNED	R/W	0h	When CCR.UNALIGN_TRP is enabled, and there is an attempt to make an unaligned memory access, then this fault occurs. Unaligned LDM/STM/LDRD/STRD instructions always fault irrespective of the setting of CCR.UNALIGN_TRP.
23-20	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
19	NOCP	R/W	0h	Attempt to use a coprocessor instruction. The processor does not support coprocessor instructions.
18	INVPC	R/W	0h	Attempt to load EXC_RETURN into PC illegally. Invalid instruction, invalid context, invalid value. The return PC points to the instruction that tried to set the PC.
17	INVSTATE	R/W	0h	Indicates an attempt to execute in an invalid EPSR state (e.g. after a BX type instruction has changed state). This includes state change after entry to or return from exception, as well as from inter-working instructions. Return PC points to faulting instruction, with the invalid state.
16	UNDEFINSTR	R/W	0h	This bit is set when the processor attempts to execute an undefined instruction. This is an instruction that the processor cannot decode. The return PC points to the undefined instruction.
15	BFARVALID	R/W	0h	This bit is set if the Bus Fault Address Register (BFAR) contains a valid address. This is true after a bus fault where the address is known. Other faults can clear this bit, such as a Mem Manage fault occurring later. If a Bus fault occurs that is escalated to a Hard Fault because of priority, the Hard Fault handler must clear this bit. This prevents problems if returning to a stacked active Bus fault handler whose BFAR value has been overwritten.
14-13	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
12	STKERR	R/W	0h	Stacking from exception has caused one or more bus faults. The SP is still adjusted and the values in the context area on the stack might be incorrect. BFAR is not written.

Table 2-128. CFSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	UNSTKERR	R/W	0h	Unstack from exception return has caused one or more bus faults. This is chained to the handler, so that the original return stack is still present. SP is not adjusted from failing return and new save is not performed. BFAR is not written.
10	IMPRECISERR	R/W	0h	Imprecise data bus error. It is a BusFault, but the Return PC is not related to the causing instruction. This is not a synchronous fault. So, if detected when the priority of the current activation is higher than the Bus Fault, it only pends. Bus fault activates when returning to a lower priority activation. If a precise fault occurs before returning to a lower priority exception, the handler detects both IMPRECISERR set and one of the precise fault status bits set at the same time. BFAR is not written.
9	PRECISERR	R/W	0h	Precise data bus error return.
8	IBUSERR	R/W	0h	Instruction bus error flag. This flag is set by a prefetch error. The fault stops on the instruction, so if the error occurs under a branch shadow, no fault occurs. BFAR is not written.
7	MMARVALID	R/W	0h	Memory Manage Address Register (MMFAR) address valid flag. A later-arriving fault, such as a bus fault, can clear a memory manage fault.. If a MemManage fault occurs that is escalated to a Hard Fault because of priority, the Hard Fault handler must clear this bit. This prevents problems on return to a stacked active MemManage handler whose MMFAR value has been overwritten.
6-5	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4	MSTKERR	R/W	0h	Stacking from exception has caused one or more access violations. The SP is still adjusted and the values in the context area on the stack might be incorrect. MMFAR is not written.
3	MUNSTKERR	R/W	0h	Unstack from exception return has caused one or more access violations. This is chained to the handler, so that the original return stack is still present. SP is not adjusted from failing return and new save is not performed. MMFAR is not written.
2	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	DACCVIOL	R/W	0h	Data access violation flag. Attempting to load or store at a location that does not permit the operation sets this flag. The return PC points to the faulting instruction. This error loads MMFAR with the address of the attempted access.
0	IACCVIOL	R/W	0h	Instruction access violation flag. Attempting to fetch an instruction from a location that does not permit execution sets this flag. This occurs on any access to an XN region, even when the MPU is disabled or not present. The return PC points to the faulting instruction. MMFAR is not written.

2.5.5.12 HFSR Register (Offset = 2Ch) [Reset = 00000000h]

HFSR is shown in Table 2-129.

Return to the Summary Table.

Hard Fault Status
This register is used to obtain information about events that activate the Hard Fault handler. This register is a write-clear register. This means that writing a 1 to a bit clears that bit.

Table 2-129. HFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DEBUGEVT	R/W1C	0h	This bit is set if there is a fault related to debug. This is only possible when halting debug is not enabled. For monitor enabled debug, it only happens for BKPT when the current priority is higher than the monitor. When both halting and monitor debug are disabled, it only happens for debug events that are not ignored (minimally, BKPT). The Debug Fault Status Register is updated.
30	FORCED	R/W1C	0h	Hard Fault activated because a Configurable Fault was received and cannot activate because of priority or because the Configurable Fault is disabled. The Hard Fault handler then has to read the other fault status registers to determine cause.
29-2	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	VECTTBL	R/W1C	0h	This bit is set if there is a fault because of vector table read on exception processing (Bus Fault). This case is always a Hard Fault. The return PC points to the pre-empted instruction.
0	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.13 DFSR Register (Offset = 30h) [Reset = 00000000h]

DFSR is shown in Table 2-130.

Return to the Summary Table.

Debug Fault Status
This register is used to monitor external debug requests, vector catches, data watchpoint match, BKPT instruction execution, halt requests. Multiple flags in the Debug Fault Status Register can be set when multiple fault conditions occur. The register is read/write clear. This means that it can be read normally. Writing a 1 to a bit clears that bit. Note that these bits are not set unless the event is caught. This means that it causes a stop of some sort. If halting debug is enabled, these events stop the processor into debug. If debug is disabled and the debug monitor is enabled, then this becomes a debug monitor handler call, if priority permits. If debug and the monitor are both disabled, some of these events are Hard Faults, and some are ignored.

Table 2-130. DFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4	EXTERNAL	R/W	0h	External debug request flag. The processor stops on next instruction boundary. 0x0: External debug request signal not asserted 0x1: External debug request signal asserted
3	VCATCH	R/W	0h	Vector catch flag. When this flag is set, a flag in one of the local fault status registers is also set to indicate the type of fault. 0x0: No vector catch occurred 0x1: Vector catch occurred
2	DWTTRAP	R/W	0h	Data Watchpoint and Trace (DWT) flag. The processor stops at the current instruction or at the next instruction. 0x0: No DWT match 0x1: DWT match
1	BKPT	R/W	0h	BKPT flag. The BKPT flag is set by a BKPT instruction in flash patch code, and also by normal code. Return PC points to breakpoint containing instruction. 0x0: No BKPT instruction execution 0x1: BKPT instruction execution
0	HALTED	R/W	0h	Halt request flag. The processor is halted on the next instruction. 0x0: No halt request 0x1: Halt requested by NVIC, including step

2.5.5.14 MMFAR Register (Offset = 34h) [Reset = 00000000h]

MMFAR is shown in Table 2-131.

Return to the Summary Table.

Mem Manage Fault Address
This register is used to read the address of the location that caused a Memory Manage Fault.

Table 2-131. MMFAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	X	Mem Manage fault address field. This field is the data address of a faulted load or store attempt. When an unaligned access faults, the address is the actual address that faulted. Because an access can be split into multiple parts, each aligned, this address can be any offset in the range of the requested size. Flags CFSR.IACCVIOL, CFSR.DACCVIOL ,CFSR.MUNSTKERR and CFSR.MSTKERR in combination with CFSR.MMARVALIDindicate the cause of the fault.

2.5.5.15 BFAR Register (Offset = 38h) [Reset = 00000000h]

BFAR is shown in Table 2-132.

Return to the Summary Table.

Bus Fault Address
This register is used to read the address of the location that generated a Bus Fault.

Table 2-132. BFAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	X	Bus fault address field. This field is the data address of a faulted load or store attempt. When an unaligned access faults, the address is the address requested by the instruction, even if that is not the address that faulted. Flags CFSR.IBUSERR, CFSR.PRECISERR, CFSR.IMPRECISERR, CFSR.UNSTKERR and CFSR.STKERR in combination with CFSR.BFARVALID indicate the cause of the fault.

2.5.5.16 AFSR Register (Offset = 3Ch) [Reset = 00000000h]

AFSR is shown in Table 2-133.

Return to the Summary Table.

Auxiliary Fault Status
This register is used to determine additional system fault information to software. Single-cycle high level on an auxiliary faults is latched as one. The bit can only be cleared by writing a one to the corresponding bit. Auxiliary fault inputs to the CPU are tied to 0.

Table 2-133. AFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IMPDEF	R/W	0h	Implementation defined. The bits map directly onto the signal assignment to the auxiliary fault inputs. Tied to 0

2.5.5.17 ID_PFR0 Register (Offset = 40h) [Reset = 00000030h]

ID_PFR0 is shown in Table 2-134.

Return to the Summary Table.

Processor Feature 0

Table 2-134. ID_PFR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	STATE1	R	3h	State1 (T-bit == 1) 0x0: N/A 0x1: N/A 0x2: Thumb-2 encoding with the 16-bit basic instructions plus 32-bit Buncond/BL but no other 32-bit basic instructions (Note non-basic 32-bit instructions can be added using the appropriate instruction attribute, but other 32-bit basic instructions cannot.) 0x3: Thumb-2 encoding with all Thumb-2 basic instructions
3-0	STATE0	R	0h	State0 (T-bit == 0) 0x0: No ARM encoding 0x1: N/A

2.5.5.18 ID_PFR1 Register (Offset = 44h) [Reset = 00000210h]

ID_PFR1 is shown in Table 2-135.

Return to the Summary Table.

Processor Feature 1

Table 2-135. ID_PFR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-8	MICROCONTROLLER_ PROGRAMMERS_MODEL	R	2h	Microcontroller programmer's model 0x0: Not supported 0x2: Two-stack support
7-4	SECURITY	R	1h	Security. Identifies whether the Security Extension is implemented
3-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.19 ID_DFR0 Register (Offset = 48h) [Reset = 00100000h]

ID_DFR0 is shown in Table 2-136.

Return to the Summary Table.

Debug Feature 0
This register provides a high level view of the debug system. Further details are provided in the debug infrastructure itself.

Table 2-136. ID_DFR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23-20	MICROCONTROLLER_ DEBUG_MODEL	R	1h	Microcontroller Debug Model - memory mapped 0x0: Not supported 0x1: Microcontroller debug v1 (ITMv1 and DWTv1)
19-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.20 ID_AFR0 Register (Offset = 4Ch) [Reset = 00000000h]

ID_AFR0 is shown in Table 2-137.

Return to the Summary Table.

Auxiliary Feature 0
This register provides some freedom for implementation defined features to be registered. Not used in Cortex-M.

Table 2-137. ID_AFR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.21 ID_MMFR0 Register (Offset = 50h) [Reset = 00100030h]

ID_MMFR0 is shown in Table 2-138.

Return to the Summary Table.

Memory Model Feature 0
General information on the memory model and memory management support.

Table 2-138. ID_MMFR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	00100030h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.22 ID_MMFR1 Register (Offset = 54h) [Reset = 00000000h]

ID_MMFR1 is shown in Table 2-139.

Return to the Summary Table.

Memory Model Feature 1
General information on the memory model and memory management support.

Table 2-139. ID_MMFR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.23 ID_MMFR2 Register (Offset = 58h) [Reset = 01000000h]

ID_MMFR2 is shown in Table 2-140.

Return to the Summary Table.

Memory Model Feature 2
General information on the memory model and memory management support.

Table 2-140. ID_MMFR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
27-24	WAIT_FOR_INTERRUPT_ STALLING	R	1h	wait for interrupt stalling 0x0: Not supported 0x1: Wait for interrupt supported
23-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.24 ID_MMFR3 Register (Offset = 5Ch) [Reset = 00000000h]

ID_MMFR3 is shown in Table 2-141.

Return to the Summary Table.

Memory Model Feature 3
General information on the memory model and memory management support.

Table 2-141. ID_MMFR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.25 ID_ISAR0 Register (Offset = 60h) [Reset = 01101110h]

ID_ISAR0 is shown in Table 2-142.

Return to the Summary Table.

ISA Feature 0
Information on the instruction set attributes register

Table 2-142. ID_ISAR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	01101110h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.26 ID_ISAR1 Register (Offset = 64h) [Reset = 02112000h]

ID_ISAR1 is shown in Table 2-143.

Return to the Summary Table.

ISA Feature 1
Information on the instruction set attributes register

Table 2-143. ID_ISAR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	02112000h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.27 ID_ISAR2 Register (Offset = 68h) [Reset = 21232231h]

ID_ISAR2 is shown in Table 2-144.

Return to the Summary Table.

ISA Feature 2
Information on the instruction set attributes register

Table 2-144. ID_ISAR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	21232231h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.28 ID_ISAR3 Register (Offset = 6Ch) [Reset = 01111131h]

ID_ISAR3 is shown in Table 2-145.

Return to the Summary Table.

ISA Feature 3
Information on the instruction set attributes register

Table 2-145. ID_ISAR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	01111131h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.5.29 ID_ISAR4 Register (Offset = 70h) [Reset = 01310132h]

ID_ISAR4 is shown in Table 2-146.

Return to the Summary Table.

ISA Feature 4
Information on the instruction set attributes register

Table 2-146. ID_ISAR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	01310132h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.6 CPU_MPU Registers

Table 2-147 lists the memory-mapped registers for the CPU_MPU registers. All register offset addresses not listed in Table 2-147 should be considered as reserved locations and the register contents should not be modified.

Table 2-147. CPU_MPU Registers

Offset	Acronym	Register Name	Section
0h	TYPE	The MPU Type Register indicates how many regions the MPU supports	Section 2.5.6.1
4h	CTRL	Enables the MPU and, when the MPU is enabled, controls whether the default memory map is enabled as a background region for privileged accesses, and whether the MPU is enabled for HardFaults, NMIs, and exception handlers when FAULTMASK is set to 1	Section 2.5.6.2
8h	RNR	Selects the region currently accessed by MPU_RBAR and MPU_RLAR	Section 2.5.6.3
Ch	RBAR	Provides indirect read and write access to the base address of the currently selected MPU region	Section 2.5.6.4
10h	RLAR	Provides indirect read and write access to the limit address of the currently selected MPU region	Section 2.5.6.5
14h	RBAR_A1	Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]:(1[1:0])	Section 2.5.6.6
18h	RLAR_A1	Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(1[1:0])	Section 2.5.6.7
1Ch	RBAR_A2	Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]:(2[1:0])	Section 2.5.6.8
20h	RLAR_A2	Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(2[1:0])	Section 2.5.6.9
24h	RBAR_A3	Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]:(3[1:0])	Section 2.5.6.10
28h	RLAR_A3	Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(3[1:0])	Section 2.5.6.11
30h	MAIR0	Along with MPU_MAIR1, provides the memory attribute encodings corresponding to the AttrIndex values	Section 2.5.6.12
3Ch	MAIR1	Along with MPU_MAIR0, provides the memory attribute encodings corresponding to the AttrIndex values	Section 2.5.6.13

Complex bit access types are encoded to fit into small table cells. Table 2-148 shows the codes that are used for access types in this section.

Table 2-148. CPU_MPU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.6.1 TYPE Register (Offset = 0h) [Reset = 00000800h]

TYPE is shown in Table 2-149.

Return to the Summary Table.

The MPU Type Register indicates how many regions the MPU supports

Table 2-149. TYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
15-8	DREGION	R	8h	Number of regions supported by the MPU
7-1	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
0	SEPARATE	R	0h	Indicates support for separate instructions and data address regions

2.5.6.2 CTRL Register (Offset = 4h) [Reset = 00000000h]

CTRL is shown in Table 2-150.

Return to the Summary Table.

Enables the MPU and, when the MPU is enabled, controls whether the default memory map is enabled as a background region for privileged accesses, and whether the MPU is enabled for HardFaults, NMIs, and exception handlers when FAULTMASK is set to 1

Table 2-150. CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
2	PRIVDEFENA	R/W	0h	Controls whether the default memory map is enabled for privileged software
1	HFNMIENA	R/W	0h	Controls whether handlers executing with priority less than 0 access memory with the MPU enabled or disabled. This applies to HardFaults, NMIs, and exception handlers when FAULTMASK is set to 1
0	ENABLE	R/W	0h	Enables the MPU

2.5.6.3 RNR Register (Offset = 8h) [Reset = 00000000h]

RNR is shown in Table 2-151.

Return to the Summary Table.

Selects the region currently accessed by MPU_RBAR and MPU_RLAR

Table 2-151. RNR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	REGION	R/W	0h	Indicates the memory region accessed by MPU_RBAR and MPU_RLAR

2.5.6.4 RBAR Register (Offset = Ch) [Reset = 00000000h]

RBAR is shown in Table 2-152.

Return to the Summary Table.

Provides indirect read and write access to the base address of the currently selected MPU region

Table 2-152. RBAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	BASE	R/W	0h	Contains bits [31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero extended to provide the base address to be checked against
4-3	SH	R/W	0h	Defines the Shareability domain of this region for Normal memory
2-1	AP	R/W	0h	Defines the access permissions for this region
0	XN	R/W	0h	Defines whether code can be executed from this region

2.5.6.5 RLAR Register (Offset = 10h) [Reset = 00000000h]

RLAR is shown in Table 2-153.

Return to the Summary Table.

Provides indirect read and write access to the limit address of the currently selected MPU region

Table 2-153. RLAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	LIMIT	R/W	0h	Contains bits [31:5] of the upper inclusive limit of the selected MPU memory region. This value is postfixed with 0x1F to provide the limit address to be checked against
4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3-1	ATTRINDX	R/W	0h	Associates a set of attributes in the MPU_MAIR0 and MPU_MAIR1 fields
0	EN	R/W	0h	Region enable

2.5.6.6 RBAR_A1 Register (Offset = 14h) [Reset = 00000000h]

RBAR_A1 is shown in Table 2-154.

Return to the Summary Table.

Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]: (1[1:0])

Table 2-154. RBAR_A1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	BASE	R/W	0h	Contains bits [31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero extended to provide the base address to be checked against
4-3	SH	R/W	0h	Defines the Shareability domain of this region for Normal memory
2-1	AP	R/W	0h	Defines the access permissions for this region
0	XN	R/W	0h	Defines whether code can be executed from this region

2.5.6.7 RLAR_A1 Register (Offset = 18h) [Reset = 00000000h]

RLAR_A1 is shown in Table 2-155.

Return to the Summary Table.

Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(1[1:0])

Table 2-155. RLAR_A1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	LIMIT	R/W	0h	Contains bits [31:5] of the upper inclusive limit of the selected MPU memory region. This value is postfixed with 0x1F to provide the limit address to be checked against
4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3-1	ATTRINDX	R/W	0h	Associates a set of attributes in the MPU_MAIR0 and MPU_MAIR1 fields
0	EN	R/W	0h	Region enable

2.5.6.8 RBAR_A2 Register (Offset = 1Ch) [Reset = 00000000h]

RBAR_A2 is shown in Table 2-156.

Return to the Summary Table.

Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]: (2[1:0])

Table 2-156. RBAR_A2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	BASE	R/W	0h	Contains bits [31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero extended to provide the base address to be checked against
4-3	SH	R/W	0h	Defines the Shareability domain of this region for Normal memory
2-1	AP	R/W	0h	Defines the access permissions for this region
0	XN	R/W	0h	Defines whether code can be executed from this region

2.5.6.9 RLAR_A2 Register (Offset = 20h) [Reset = 00000000h]

RLAR_A2 is shown in Table 2-157.

Return to the Summary Table.

Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(2[1:0])

Table 2-157. RLAR_A2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	LIMIT	R/W	0h	Contains bits [31:5] of the upper inclusive limit of the selected MPU memory region. This value is postfixed with 0x1F to provide the limit address to be checked against
4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3-1	ATTRINDX	R/W	0h	Associates a set of attributes in the MPU_MAIR0 and MPU_MAIR1 fields
0	EN	R/W	0h	Region enable

2.5.6.10 RBAR_A3 Register (Offset = 24h) [Reset = 00000000h]

RBAR_A3 is shown in Table 2-158.

Return to the Summary Table.

Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]: (3[1:0])

Table 2-158. RBAR_A3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	BASE	R/W	0h	Contains bits [31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero extended to provide the base address to be checked against
4-3	SH	R/W	0h	Defines the Shareability domain of this region for Normal memory
2-1	AP	R/W	0h	Defines the access permissions for this region
0	XN	R/W	0h	Defines whether code can be executed from this region

2.5.6.11 RLAR_A3 Register (Offset = 28h) [Reset = 00000000h]

RLAR_A3 is shown in Table 2-159.

Return to the Summary Table.

Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(3[1:0])

Table 2-159. RLAR_A3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	LIMIT	R/W	0h	Contains bits [31:5] of the upper inclusive limit of the selected MPU memory region. This value is postfixed with 0x1F to provide the limit address to be checked against
4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3-1	ATTRINDX	R/W	0h	Associates a set of attributes in the MPU_MAIR0 and MPU_MAIR1 fields
0	EN	R/W	0h	Region enable

2.5.6.12 MAIR0 Register (Offset = 30h) [Reset = 00000000h]

MAIR0 is shown in Table 2-160.

Return to the Summary Table.

Along with MPU_MAIR1, provides the memory attribute encodings corresponding to the AttrIndex values

Table 2-160. MAIR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	ATTR3	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 3
23-16	ATTR2	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 2
15-8	ATTR1	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 1
7-0	ATTR0	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 0

2.5.6.13 MAIR1 Register (Offset = 3Ch) [Reset = 00000000h]

MAIR1 is shown in Table 2-161.

Return to the Summary Table.

Along with MPU_MAIR0, provides the memory attribute encodings corresponding to the AttrIndex values

Table 2-161. MAIR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	ATTR7	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 7
23-16	ATTR6	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 6
15-8	ATTR5	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 5
7-0	ATTR4	R/W	0h	Memory attribute encoding for MPU regions with an AttrIndex of 4

2.5.7 CPU_SAU Registers

Table 2-162 lists the memory-mapped registers for the CPU_SAU registers. All register offset addresses not listed in Table 2-162 should be considered as reserved locations and the register contents should not be modified.

Table 2-162. CPU_SAU Registers

Offset	Acronym	Register Name	Section
0h	CTRL	Allows enabling of the Security Attribution Unit	Section 2.5.7.1
4h	TYPE	Indicates the number of regions implemented by the Security Attribution Unit	Section 2.5.7.2
8h	RNR	Selects the region currently accessed by SAU_RBAR and SAU_RLAR	Section 2.5.7.3
Ch	RBAR	Provides indirect read and write access to the base address of the currently selected SAU region	Section 2.5.7.4
10h	RLAR	Provides indirect read and write access to the limit address of the currently selected SAU region	Section 2.5.7.5
14h	SFSR	Provides information about any security related faults	Section 2.5.7.6
18h	SFAR	Shows the address of the memory location that caused a Security violation	Section 2.5.7.7

Complex bit access types are encoded to fit into small table cells. Table 2-163 shows the codes that are used for access types in this section.

Table 2-163. CPU_SAU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.7.1 CTRL Register (Offset = 0h) [Reset = 00000000h]

CTRL is shown in Table 2-164.

Return to the Summary Table.

Allows enabling of the Security Attribution Unit

Table 2-164. CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	ALLNS	R/W	0h	When SAU_CTRL.ENABLE is 0 this bit controls if the memory is marked as Non-secure or Secure
0	ENABLE	R/W	0h	Enables the SAU

2.5.7.2 TYPE Register (Offset = 4h) [Reset = 00000000h]

TYPE is shown in Table 2-165.

Return to the Summary Table.

Indicates the number of regions implemented by the Security Attribution Unit

Table 2-165. TYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	SREGION	R	0h	The number of implemented SAU regions

2.5.7.3 RNR Register (Offset = 8h) [Reset = 00000000h]

RNR is shown in Table 2-166.

Return to the Summary Table.

Selects the region currently accessed by SAU_RBAR and SAU_RLAR

Table 2-166. RNR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	REGION	R/W	0h	Indicates the SAU region accessed by SAU_RBAR and SAU_RLAR

2.5.7.4 RBAR Register (Offset = Ch) [Reset = 00000000h]

RBAR is shown in Table 2-167.

Return to the Summary Table.

Provides indirect read and write access to the base address of the currently selected SAU region

Table 2-167. RBAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	BADDR	R/W	0h	Holds bits [31:5] of the base address for the selected SAU region
4-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.7.5 RLAR Register (Offset = 10h) [Reset = 00000000h]

RLAR is shown in Table 2-168.

Return to the Summary Table.

Provides indirect read and write access to the limit address of the currently selected SAU region

Table 2-168. RLAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	LADDR	R/W	0h	Holds bits [31:5] of the limit address for the selected SAU region
4-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	NSC	R/W	0h	Controls whether Non-secure state is permitted to execute an SG instruction from this region
0	ENABLE	R/W	0h	SAU region enable

2.5.7.6 SFSR Register (Offset = 14h) [Reset = 00000000h]

SFSR is shown in Table 2-169.

Return to the Summary Table.

Provides information about any security related faults

Table 2-169. SFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7	LSERR	R/W	0h	Sticky flag indicating that an error occurred during lazy state activation or deactivation
6	SFARVALID	R/W	0h	This bit is set when the SFAR register contains a valid value. As with similar fields, such as BFSR.BFARVALID and MMFSR.MMARVALID, this bit can be cleared by other exceptions, such as BusFault
5	LSPERR	R/W	0h	Stick flag indicating that an SAU or IDAU violation occurred during the lazy preservation of floating-point state
4	INVTRAN	R/W	0h	Sticky flag indicating that an exception was raised due to a branch that was not flagged as being domain crossing causing a transition from Secure to Non-secure memory
3	AUVIOL	R/W	0h	Sticky flag indicating that an attempt was made to access parts of the address space that are marked as Secure with NS-Req for the transaction set to Non-secure. This bit is not set if the violation occurred during lazy state preservation. See LSPERR
2	INVER	R/W	0h	This can be caused by EXC_RETURN.DCRS being set to 0 when returning from an exception in the Non-secure state, or by EXC_RETURN.ES being set to 1 when returning from an exception in the Non-secure state
1	INVIS	R/W	0h	This bit is set if the integrity signature in an exception stack frame is found to be invalid during the unstacking operation
0	INVEP	R/W	0h	This bit is set if a function call from the Non-secure state or exception targets a non-SG instruction in the Secure state. This bit is also set if the target address is a SG instruction, but there is no matching SAU/IDAU region with the NSC flag set

2.5.7.7 SFAR Register (Offset = 18h) [Reset = 00000000h]

SFAR is shown in Table 2-170.

Return to the Summary Table.

Shows the address of the memory location that caused a Security violation

Table 2-170. SFAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	0h	The address of an access that caused a attribution unit violation. This field is only valid when SFSR.SFARVALID is set. This allows the actual flip flops associated with this register to be shared with other fault address registers. If an implementation chooses to share the storage in this way, care must be taken to not leak Secure address information to the Non-secure state. One way of achieving this is to share the SFAR register with the MMFAR_S register, which is not accessible to the Non-secure state

2.5.8 CPU_DCB Registers

Table 2-171 lists the memory-mapped registers for the CPU_DCB registers. All register offset addresses not listed in Table 2-171 should be considered as reserved locations and the register contents should not be modified.

Table 2-171. CPU_DCB Registers

Offset	Acronym	Register Name	Section
10h	DHCSR	Controls halting debug	Section 2.5.8.1
14h	DCRSR	With the DCRDR, provides debug access to the general-purpose registers, special-purpose registers, and the FP extension registers. A write to the DCRSR specifies the register to transfer, whether the transfer is a read or write, and starts the transfer	Section 2.5.8.2
18h	DCRDR	With the DCRSR, provides debug access to the general-purpose registers, special-purpose registers, and the FP Extension registers. If the Main Extension is implemented, it can also be used for message passing between an external debugger and a debug agent running on the PE	Section 2.5.8.3
1Ch	DEMCR	Manages vector catch behavior and DebugMonitor handling when debugging	Section 2.5.8.4
24h	DAUTHCTRL	This register allows the external authentication interface to be	Section 2.5.8.5
28h	DSCSR	Provides control and status information for Secure debug	Section 2.5.8.6

Complex bit access types are encoded to fit into small table cells. Table 2-172 shows the codes that are used for access types in this section.

Table 2-172. CPU_DCB Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.8.1 DHCSR Register (Offset = 10h) [Reset = 00000000h]

DHCSR is shown in Table 2-173.

Return to the Summary Table.

Controls halting debug

Table 2-173. DHCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
26	S_RESTART_ST	R	0h	Indicates the PE has processed a request to clear DHCSR.C_HALT to 0. That is, either a write to DHCSR that clears DHCSR.C_HALT from 1 to 0, or an External Restart Request
25	S_RESET_ST	R	0h	Indicates whether the PE has been reset since the last read of the DHCSR
24	S_RETIRE_ST	R	0h	Set to 1 every time the PE retires one of more instructions
23-21	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
20	S_SDE	R	0h	Indicates whether Secure invasive debug is allowed
19	S_LOCKUP	R	0h	Indicates whether the PE is in Lockup state
18	S_SLEEP	R	0h	Indicates whether the PE is sleeping
17	S_HALT	R	0h	Indicates whether the PE is in Debug state
31-16	DBGKEY	W	0h	A debugger must write 0xA05F to this field to enable write access to the remaining bits, otherwise the PE ignores the write access
15-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5	C_SNAPSTALL	R	0h	Allow imprecise entry to Debug state
4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	C_MASKINTS	R/W	0h	When debug is enabled, the debugger can write to this bit to mask PendSV, SysTick and external configurable interrupts
2	C_STEP	R/W	0h	Enable single instruction step
1	C_HALT	R/W	0h	PE enter Debug state halt request
0	C_DEBUGEN	R/W	0h	Enable Halting debug

2.5.8.2 DCRSR Register (Offset = 14h) [Reset = 00000000h]

DCRSR is shown in Table 2-174.

Return to the Summary Table.

With the DCRDR, provides debug access to the general-purpose registers, special-purpose registers, and the FP extension registers. A write to the DCRSR specifies the register to transfer, whether the transfer is a read or write, and starts the transfer

Table 2-174. DCRSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
16	REGWnR	W	0h	Specifies the access type for the transfer
15-7	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
6-0	REGSEL	W	0h	Specifies the general-purpose register, special-purpose register, or FP register to transfer

2.5.8.3 DCRDR Register (Offset = 18h) [Reset = 00000000h]

DCRDR is shown in Table 2-175.

Return to the Summary Table.

With the DCRSR, provides debug access to the general-purpose registers, special-purpose registers, and the FP Extension registers. If the Main Extension is implemented, it can also be used for message passing between an external debugger and a debug agent running on the PE

Table 2-175. DCRDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DBGTMP	R/W	0h	Provides debug access for reading and writing the general-purpose registers, special-purpose registers, and Floating-point Extension registers

2.5.8.4 DEMCR Register (Offset = 1Ch) [Reset = 00000000h]

DEMCR is shown in Table 2-176.

Return to the Summary Table.

Manages vector catch behavior and DebugMonitor handling when debugging

Table 2-176. DEMCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
24	TRCENA	R/W	0h	Global enable for all DWT and ITM features
23-21	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
20	SDME	R	0h	Indicates whether the DebugMonitor targets the Secure or the Non-secure state and whether debug events are allowed in Secure state
19	MON_REQ	R	0h	DebugMonitor semaphore bit
18	MON_STEP	R	0h	Enable DebugMonitor stepping
17	MON_PEND	R	0h	Sets or clears the pending state of the DebugMonitor exception
16	MON_EN	R	0h	Enable the DebugMonitor exception
15-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11	VC_SFERR	R	0h	SecureFault exception halting debug vector catch enable
10	VC_HARDERR	R/W	0h	HardFault exception halting debug vector catch enable
9	VC_INTERR	R	0h	Enable halting debug vector catch for faults during exception entry and return
8	VC_BUSERR	R	0h	BusFault exception halting debug vector catch enable
7	VC_STATERR	R	0h	Enable halting debug trap on a UsageFault exception caused by a state information error, for example an Undefined Instruction exception
6	VC_CHKERR	R	0h	Enable halting debug trap on a UsageFault exception caused by a checking error, for example an alignment check error
5	VC_NOCPERR	R	0h	Enable halting debug trap on a UsageFault caused by an access to a coprocessor
4	VC_MMERR	R	0h	Enable halting debug trap on a MemManage exception
3-1	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
0	VC_CORERESET	R/W	0h	Enable Reset Vector Catch. This causes a warm reset to halt a running system

2.5.8.5 DAUTHCTRL Register (Offset = 24h) [Reset = 00000000h]

DAUTHCTRL is shown in Table 2-177.

Return to the Summary Table.

This register allows the external authentication interface to be overridden from software.

Table 2-177. DAUTHCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	INTSPNIDEN	R/W	0h	Internal Secure non-invasive debug enable. Overrides the external Secure non-invasive debug authentication interface
2	SPNIDENSEL	R/W	0h	Secure non-invasive debug enable select. Selects between DAUTHCTRL and the external authentication interface for control of Secure non-invasive debug
1	INTSPIDEN	R/W	0h	Internal Secure invasive debug enable. Overrides the external Secure invasive debug authentication Interfaces.
0	SPIDENSEL	R/W	0h	Secure invasive debug enable select. Selects between DAUTHCTRL and the external authentication interface for control of Secure invasive debug.

2.5.8.6 DSCSR Register (Offset = 28h) [Reset = 00000000h]

DSCSR is shown in Table 2-178.

Return to the Summary Table.

Provides control and status information for Secure debug

Table 2-178. DSCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
17	CDSKEY	R/W	0h	Writes to the CDS bit are ignored unless CDSKEY is concurrently written to zero
16	CDS	R/W	0h	This field indicates the current Security state of the processor
15-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	SBRSEL	R/W	0h	If SBRSELEN is 1 this bit selects whether the Non-secure or the Secure version of the memory-mapped Banked registers are accessible to the debugger
0	SBRSELEN	R/W	0h	Controls whether the SBRSEL field or the current Security state of the processor selects which version of the memory-mapped Banked registers are accessed to the debugger

2.5.9 CPU_SIG Registers

Table 2-179 lists the memory-mapped registers for the CPU_SIG registers. All register offset addresses not listed in Table 2-179 should be considered as reserved locations and the register contents should not be modified.

Table 2-179. CPU_SIG Registers

Offset	Acronym	Register Name	Section
0h	STIR	Provides a mechanism for software to generate an interrupt	Section 2.5.9.1

Complex bit access types are encoded to fit into small table cells. Table 2-180 shows the codes that are used for access types in this section.

Table 2-180. CPU_SIG Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.9.1 STIR Register (Offset = 0h) [Reset = 00000000h]

STIR is shown in Table 2-181.

Return to the Summary Table.

Provides a mechanism for software to generate an interrupt

Table 2-181. STIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8-0	INTID	W	0h	Indicates the interrupt to be pended. The value written is (ExceptionNumber - 16)

2.5.10 CPU_FPU Registers

Table 2-182 lists the memory-mapped registers for the CPU_FPU registers. All register offset addresses not listed in Table 2-182 should be considered as reserved locations and the register contents should not be modified.

Table 2-182. CPU_FPU Registers

Offset	Acronym	Register Name	Section
4h	FPCCR	Holds control data for the Floating-point extension	Section 2.5.10.1
8h	FPCAR	Holds the location of the unpopulated floating-point register space allocated on an exception stack frame	Section 2.5.10.2
Ch	FPDSCR	Holds the default values for the floating-point status control data that the PE assigns to the FPSCR when it creates a new floating-point context	Section 2.5.10.3
10h	MVFR0	Describes the features provided by the Floating-point Extension	Section 2.5.10.4
14h	MVFR1	Describes the features provided by the Floating-point Extension	Section 2.5.10.5
18h	MVFR2	Describes the features provided by the Floating-point Extension	Section 2.5.10.6

Complex bit access types are encoded to fit into small table cells. Table 2-183 shows the codes that are used for access types in this section.

Table 2-183. CPU_FPU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.10.1 FPCCR Register (Offset = 4h) [Reset = 00000000h]

FPCCR is shown in Table 2-184.

Return to the Summary Table.

Holds control data for the Floating-point extension

Table 2-184. FPCCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ASPEN	R/W	0h	When this bit is set to 1, execution of a floating-point instruction sets the CONTROL.FPCA bit to 1
30	LSPEN	R/W	0h	Enables lazy context save of floating-point state
29	LSPENS	R/W	0h	This bit controls whether the LSPEN bit is writeable from the Non-secure state
28	CLRONRET	R/W	0h	Clear floating-point caller saved registers on exception return
27	CLRONRETS	R/W	0h	This bit controls whether the CLRONRET bit is writeable from the Non-secure state
26	TS	R/W	0h	Treat floating-point registers as Secure enable
25-11	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
10	UFRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the UsageFault exception to pending
9	SPLIMVIOL	R/W	0h	This bit is banked between the Security states and indicates whether the floating-point context violates the stack pointer limit that was active when lazy state preservation was activated. SPLIMVIOL modifies the lazy floating-point state preservation behavior
8	MONRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the DebugMonitor exception to pending
7	SFRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the SecureFault exception to pending. This bit is only present in the Secure version of the register, and behaves as RAZ/WI when accessed from the Non-secure state
6	BFRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the BusFault exception to pending
5	MMRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the MemManage exception to pending
4	HFRDY	R/W	0h	Indicates whether the software executing when the PE allocated the floating-point stack frame was able to set the HardFault exception to pending
3	THREAD	R/W	0h	Indicates the PE mode when it allocated the floating-point stack frame
2	S	R/W	0h	Security status of the floating-point context. This bit is only present in the Secure version of the register, and behaves as RAZ/WI when accessed from the Non-secure state. This bit is updated whenever lazy state preservation is activated, or when a floating-point instruction is executed
1	USER	R/W	0h	Indicates the privilege level of the software executing when the PE allocated the floating-point stack frame
0	LSPACT	R/W	0h	Indicates whether lazy preservation of the floating-point state is active

2.5.10.2 FPCAR Register (Offset = 8h) [Reset = 00000000h]

FPCAR is shown in Table 2-185.

Return to the Summary Table.

Holds the location of the unpopulated floating-point register space allocated on an exception stack frame

Table 2-185. FPCAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	ADDRESS	R/W	0h	The location of the unpopulated floating-point register space allocated on an exception stack frame
2-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.10.3 FPDSCR Register (Offset = Ch) [Reset = 00000000h]

FPDSCR is shown in Table 2-186.

Return to the Summary Table.

Holds the default values for the floating-point status control data that the PE assigns to the FPSCR when it creates a new floating-point context

Table 2-186. FPDSCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
26	AHP	R/W	0h	Default value for FPSCR.AHP
25	DN	R/W	0h	Default value for FPSCR.DN
24	FZ	R/W	0h	Default value for FPSCR.FZ
23-22	RMode	R/W	0h	Default value for FPSCR.RMode
21-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.10.4 MVFR0 Register (Offset = 10h) [Reset = 00000000h]

MVFR0 is shown in Table 2-187.

Return to the Summary Table.

Describes the features provided by the Floating-point Extension

Table 2-187. MVFR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	FPRound	R	0h	Indicates the rounding modes supported by the FP Extension
27-24	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
23-20	FPSqrt	R	0h	Indicates the support for FP square root operations
19-16	FPDivide	R	0h	Indicates the support for FP divide operations
15-12	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
11-8	FPDP	R	0h	Indicates support for FP double-precision operations
7-4	FPSP	R	0h	Indicates support for FP single-precision operations
3-0	SIMDReg	R	0h	Indicates size of FP register file

2.5.10.5 MVFR1 Register (Offset = 14h) [Reset = 00000000h]

MVFR1 is shown in Table 2-188.

Return to the Summary Table.

Describes the features provided by the Floating-point Extension

Table 2-188. MVFR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	FMAC	R	0h	Indicates whether the FP Extension implements the fused multiply accumulate instructions
27-24	FPHP	R	0h	Indicates whether the FP Extension implements half-precision FP conversion instructions
23-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	FPDNaN	R	0h	Indicates whether the FP hardware implementation supports NaN propagation
3-0	FPFtZ	R	0h	Indicates whether subnormals are always flushed-to-zero

2.5.10.6 MVFR2 Register (Offset = 18h) [Reset = 00000000h]

MVFR2 is shown in Table 2-189.

Return to the Summary Table.

Describes the features provided by the Floating-point Extension

Table 2-189. MVFR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	FPMisc	R	0h	Indicates support for miscellaneous FP features
3-0	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.11 CPU_TPIU Registers

Table 2-190 lists the memory-mapped registers for the CPU_TPIU registers. All register offset addresses not listed in Table 2-190 should be considered as reserved locations and the register contents should not be modified.

Table 2-190. CPU_TPIU Registers

Offset	Acronym	Register Name	Section
0h	SSPSR	Supported Sync Port Sizes	Section 2.5.11.1
4h	CSPSR	Current Sync Port Size	Section 2.5.11.2
10h	ACPR	Async Clock Prescaler	Section 2.5.11.3
F0h	SPPR	Selected Pin Protocol	Section 2.5.11.4
300h	FFSR	Formatter and Flush Status	Section 2.5.11.5
304h	FFCR	Formatter and Flush Control	Section 2.5.11.6
308h	PSCR	Formatter Synchronization Counter	Section 2.5.11.7
FA0h	CLAIMMASK	Claim Tag Mask	Section 2.5.11.8
FA0h	CLAIMSET	Claim Tag Set	Section 2.5.11.9
FA4h	CLAIMTAG	Current Claim Tag	Section 2.5.11.10
FA4h	CLAIMCLR	Claim Tag Clear	Section 2.5.11.11
FC8h	DEVID	Device ID	Section 2.5.11.12
FCCh	DEVTYPE	The Device Type Identification	Section 2.5.11.13

Complex bit access types are encoded to fit into small table cells. Table 2-191 shows the codes that are used for access types in this section.

Table 2-191. CPU_TPIU Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

2.5.11.1 SSPSR Register (Offset = 0h) [Reset = 0000000Bh]

SSPSR is shown in Table 2-192.

Return to the Summary Table.

Supported Sync Port Sizes
This register represents a single port size that is supported on the device, that is, 4, 2 or 1. This is to ensure that tools do not attempt to select a port width that an attached TPA cannot capture.

Table 2-192. SSPSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	FOUR	R	1h	4-bit port size support 0x0: Not supported 0x1: Supported
2	THREE	R	0h	3-bit port size support 0x0: Not supported 0x1: Supported
1	TWO	R	1h	2-bit port size support 0x0: Not supported 0x1: Supported
0	ONE	R	1h	1-bit port size support 0x0: Not supported 0x1: Supported

2.5.11.2 CSPSR Register (Offset = 4h) [Reset = 00000001h]

CSPSR is shown in Table 2-193.

Return to the Summary Table.

Current Sync Port Size
This register has the same format as SSPSR but only one bit can be set, and all others must be zero. Writing values with more than one bit set, or setting a bit that is not indicated as supported can cause Unpredictable behavior. On reset this defaults to the smallest possible port size, 1 bit.

Table 2-193. CSPSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	FOUR	R/W	0h	4-bit port enable Writing values with more than one bit set in CSPSR, or setting a bit that is not indicated as supported in SSPSR can cause Unpredictable behavior.
2	THREE	R/W	0h	3-bit port enable Writing values with more than one bit set in CSPSR, or setting a bit that is not indicated as supported in SSPSR can cause Unpredictable behavior.
1	TWO	R/W	0h	2-bit port enable Writing values with more than one bit set in CSPSR, or setting a bit that is not indicated as supported in SSPSR can cause Unpredictable behavior.
0	ONE	R/W	1h	1-bit port enable Writing values with more than one bit set in CSPSR, or setting a bit that is not indicated as supported in SSPSR can cause Unpredictable behavior.

2.5.11.3 ACPR Register (Offset = 10h) [Reset = 00000000h]

ACPR is shown in Table 2-194.

Return to the Summary Table.

Async Clock Prescaler
This register scales the baud rate of the asynchronous output.

Table 2-194. ACPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
12-0	PRESCALER	R/W	0h	Divisor for input trace clock is (PRESCALER + 1).

2.5.11.4 SPPR Register (Offset = F0h) [Reset = 00000001h]

SPPR is shown in Table 2-195.

Return to the Summary Table.

Selected Pin Protocol
This register selects the protocol to be used for trace output.
Note: If this register is changed while trace data is being output, data corruption occurs.

Table 2-195. SPPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1-0	PROTOCOL	R/W	1h	Trace output protocol 0h = TracePort mode 1h = SerialWire Output (Manchester). This is the reset value. 2h = SerialWire Output (NRZ)

2.5.11.5 FFSR Register (Offset = 300h) [Reset = 00000008h]

FFSR is shown in Table 2-196.

Return to the Summary Table.

Formatter and Flush Status

Table 2-196. FFSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
3	FTNONSTOP	R	1h	0: Formatter can be stopped 1: Formatter cannot be stopped
2	TCPRESENT	R	0h	This field always reads as zero
1	FTSTOPPED	R	0h	This field always reads as zero
0	FLINPROG	R	0h	Read only. Flush in progress. Value can be: 0x0 When all the data received, before the flush is acknowledged, has been output on the trace port 0x1 When a flush is initiated.

2.5.11.6 FFCR Register (Offset = 304h) [Reset = 00000142h]

FFCR is shown in Table 2-197.

Return to the Summary Table.

Formatter and Flush Control
When one of the two single wire output (SWO) modes is selected, ENFCONT enables the formatter to be bypassed. If the formatter is bypassed, only the ITM/DWT trace source (ATDATA2) passes through. The TPIU accepts and discards data that is presented on the ETM port (ATDATA1). This function is intended to be used when it is necessary to connect a device containing an ETM to a trace capture device that is only able to capture Serial Wire Output (SWO) data. Enabling or disabling the formatter causes momentary data corruption.
Note: If the selected pin protocol register (SPPR.PROTOCOL) is set to 0x00 (TracePort mode), this register always reads 0x102, because the formatter is automatically enabled. If one of the serial wire modes is then selected, the register reverts to its previously programmed value.

Table 2-197. FFCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	TRIGIN	R/W	1h	Indicates that triggers are inserted when a trigger pin is asserted.
7	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
6	FONMAN	R/W	1h	Flush on manual. Value can be: 0x0 When the flush completes. Set to 0 on a reset of the TPIU. 0x1 Generates a flush.
5-2	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	ENFCONT	R/W	1h	Enable continuous formatting: 0: Continuous formatting disabled 1: Continuous formatting enabled
0	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.

2.5.11.7 PSCR Register (Offset = 308h) [Reset = 00000000h]

PSCR is shown in Table 2-198.

Return to the Summary Table.

Formatter Synchronization Counter

Table 2-198. PSCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4-0	PSCOUNT	R/W	0h	Periodic Synchronization Count. Determines the reload value of the Periodic Synchronization Counter. The reload value takes effect the next time the counter reaches zero. Reads from this register return the reload value programmed into this register 0b00000 Synchronization disabled. 0b00111 128 bytes. 0b01000 256 bytes. 0b11111 2^{31} bytes.

2.5.11.8 CLAIMMASK Register (Offset = FA0h) [Reset = 0000000Fh]

CLAIMMASK is shown in Table 2-199.

Return to the Summary Table.

Claim Tag Mask

Table 2-199. CLAIMMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLAIMMASK	R	Fh	This register forms one half of the Claim Tag value. When reading this register returns the number of bits that can be set (each bit is considered separately): 0: This claim tag bit is not implemented 1: This claim tag bit is implemented The behavior when writing to this register is described in CLAIMSET.

2.5.11.9 CLAIMSET Register (Offset = FA0h) [Reset = 0000000Fh]

CLAIMSET is shown in Table 2-200.

Return to the Summary Table.

Claim Tag Set

Table 2-200. CLAIMSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLAIMSET	W	Fh	This register forms one half of the Claim Tag value. Writing to this location allows individual bits to be set (each bit is considered separately): 0: No effect 1: Set this bit in the claim tag The behavior when reading from this location is described in CLAIMMASK.

2.5.11.10 CLAIMTAG Register (Offset = FA4h) [Reset = 00000000h]

CLAIMTAG is shown in Table 2-201.

Return to the Summary Table.

Current Claim Tag

Table 2-201. CLAIMTAG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLAIMTAG	R	0h	This register forms one half of the Claim Tag value. Reading this register returns the current Claim Tag value. Reading CLAIMMASK determines how many bits from this register must be used. The behavior when writing to this register is described in CLAIMCLR.

2.5.11.11 CLAIMCLR Register (Offset = FA4h) [Reset = 00000000h]

CLAIMCLR is shown in Table 2-202.

Return to the Summary Table.

Claim Tag Clear

Table 2-202. CLAIMCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLAIMCLR	W	0h	This register forms one half of the Claim Tag value. Writing to this location enables individual bits to be cleared (each bit is considered separately): 0: No effect 1: Clear this bit in the claim tag. The behavior when reading from this location is described in CLAIMTAG.

2.5.11.12 DEVID Register (Offset = FC8h) [Reset = 00000CA0h]

DEVID is shown in Table 2-203.

Return to the Summary Table.

Device ID

Table 2-203. DEVID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DEVID	R	CA0h	This field returns: 0xCA1 if there is an ETM present. 0xCA0 if there is no ETM present.

2.5.11.13 DEVTYPE Register (Offset = FCCh) [Reset = 00000011h]

DEVTYPE is shown in Table 2-204.

Return to the Summary Table.

The Device Type Identification

Table 2-204. DEVTYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	SUBTYPE	R	1h	Identifies the classification of the debug component
3-0	MAJORTYPE	R	1h	Indicates this device is a trace sink and specifically a TPIU

3.1 Introduction

The CC13x4x10 and CC26x4x10 device platform takes advantage of the TrustZone® technology for Armv8-M devices to provide a customer configurable memory map that partitions the memory map into Secure and Non-secure regions. See Table 3-1 for more details.

3.2 Memory Map (Secure and Non-secure)

Table 3-1. Memory Map (Secure and Non-secure)

Region	Start Address	End Address	User Config	ID[1]	IVALID[1]	S/NS	NSC	Additional Comments
Secure Flash	0x0	CCFG:TRUSTZONE_ FLASH_CFG.NSCADDR_ BOUNDARY - 1	CCFG	1	1	S	0	VTOR_S (Secure vector base address) is 0x0 by default
NSC Flash	CCFG:TRUSTZONE_ FLASH_CFG.NSCADDR_ BOUNDARY	CCFG:TRUSTZONE_ FLASH_CFG.NSADDR_ BOUNDARY- 1	CCFG	2	1	S	1	NSC: Non-secure callable Flash region. Note that bus masters in the SoC will see this region as Secure. The NSC property of the memory is only used by the CPU
Non-secure Flash	CCFG:TRUSTZONE_ FLASH_CFG.NSADDR_ BOUNDARY	End of Flash (device dependent)	CCFG	3	1	NS	0	Non-secure region
BROM/ GPRAM[2]	0x10000000	0x1FFFFFFF	Hardcoded	4	1	S	0	Secure region
Secure SRAM	0x20000000	CCFG:TRUSTZONE_ SRAM_CFG.NSCADDR_ BOUNDARY- 1	CCFG	5	1	S	0	Secure SRAM region
NSC SRAM	CCFG:TRUSTZONE_ SRAM_CFG.NSCADDR_ BOUNDARY	CCFG:TRUSTZONE_ SRAM_CFG.NSADDR_ BOUNDARY- 1	CCFG	6	1	S	1	NSC: Non-secure callable SRAM region. Note that bus masters in the SoC will see this region as Secure. The NSC property of the memory is only used by the CPU

Table 3-1. Memory Map (Secure and Non-secure) (continued)

Region	Start Address	End Address	User Config	ID[1]	IVALID[1]	S/NS	NSC	Additional Comments
Non-secure SRAM	CCFG:TRUSTZONE_SRAM_CFG.NSADDR_BOUNDARY	0x3FFFFFFF	CCFG	7	1	NS	0	Non-Secure SRAM
Non-secure Peripheral Region	0x40000000	0x57FFFFFF	Hardcoded	8	1	NS	0	
Secure Peripheral Region	0x58000000	0x5FFFFFFF	Hardcoded	9	1	S	0	
Non-secure Peripheral Region Alias (non-posted)	0x60000000	0x77FFFFFF	Hardcoded	10	1	NS	0	
Secure Peripheral Region Alias (non-posted)	0x78000000	0x7FFFFFFF	Hardcoded	11	1	S	0	
Non-secure (reserved)	0x80000000	0xDFFFFFFF	Hardcoded	12	1	NS	0	
Exempt (ARM Implementation)	0xE0000000	0xE0043FFF	Hardcoded	0	0	S	0	System Control Space exempted by ARM
Non-secure (reserved)	0xE0044000	0xE00FDFFF	Hardcoded	12	1	NS	0	
Exempt (Processor ROM Table)	0xE00FE000	0xE00FFFFF	Hardcoded	0	0	S	0	Excluded explicitly by IDAU
Internal (reserved)	0xE0100000	0xFFFFFFFF	Hardcoded	13	1	NS	0	

(1) The ID and IVALID fields are used by the Test Target (TT) instruction.
(2) GPRAM is only available when VIMS.CTL.MODE = 0. In Cache mode (VIMS.CTL.MODE = 1), the memory becomes cache and it caches both Secure and Non-secure regions.

Boundaries between S, NSC, and S regions must be configured such that the rules below are being followed:

The granularity required for the different watermark boundaries between S and NS are showed in Table 3-2.

Table 3-2. Granularity

CCFG Register	Granularity
TRUSTZONE_SRAM_CFG.NSCADDR_BOUNDARY	1 KB
TRUSTZONE_SRAM_CFG.NSADDR_BOUNDARY	1 KB
TRUSTZONE_FLASH_CFG.NSCADDR_BOUNDARY	1 KB
TRUSTZONE_FLASH_CFG.NSADDR_BOUNDARY	8 KB

Figure 3-1 shows the accessible memory per bus master for different runtime security configurations and Table 3-3 shows the logic of determining the security attribution for a given address (the core presents the address to the SAU and IDAU, which in turn signal to the core the security attributed of the address)

Non-secure (NS) addresses may be accessed by the Cortex®-M33 when in either the Secure or the Non-secure state. Further, NS addresses may be accessed by any of the masters in the system (DMA, I2S, Radio, or CRYPTO).

Secure (S) addresses may only be accessed by the Cortex®-M33 when it is in the Secure state, or by other Secure masters in the system. The only secure master in the system (other than the M33 in secure state) is the internal DMA of the AES and Hash Cryptoprocessor core (CRYPTO).

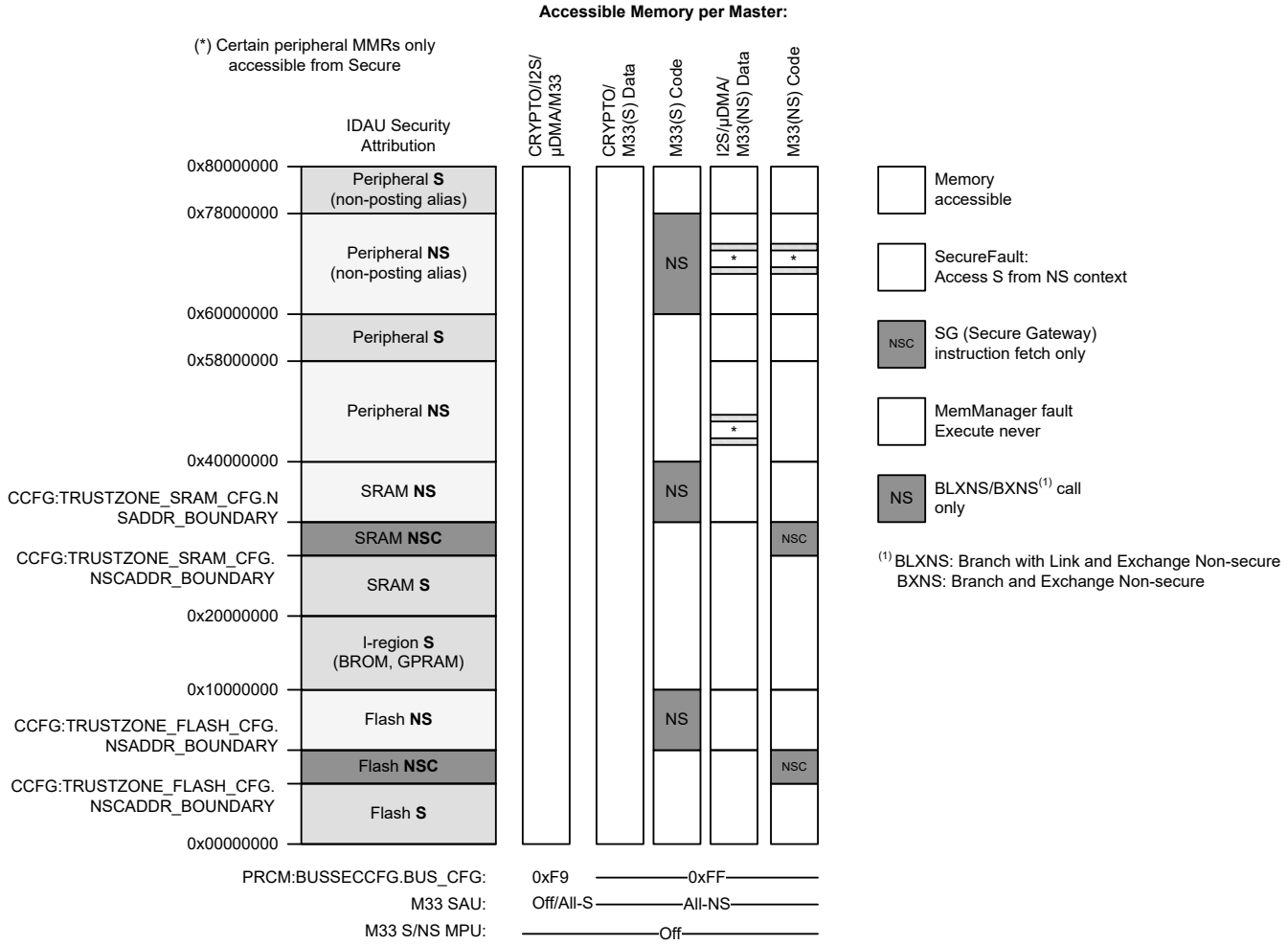


Figure 3-1. Accessible Memory per Bus Master for Different Runtime Security Configurations

Table 3-3. Determining the Security Attribution for a Given Address

IDAU Security Attribution	SAU Security Attribution	Final Security Attribution
NS	S	S
	NSC	NSC
	NS	NS
NSC	S	S
	NSC	NSC
	NS	NSC
S	S	S
	NS	S
	NSC	S

3.2.1 Bus Security

Only two values are supported for PRCM:BUSSECCFG.BUS_CFG; 0xFF and 0xF9. All other values are reserved, and can lead to unpredictable behavior. Table 3-4 through Table 3-6 show the system configuration resulting from PRCM:BUSSECCFG.BUS_CFG. This setting determines the base address of many apertures/ peripherals as well as the state of outbound address filtering for bus masters (if they can access all memory (Secure) or are restricted from accessing secure memory ranges (Non-secure)). Some peripherals only have parts of their memory map moved to Secure memory. This is for backwards compatibility.

Table 3-4. BUS_CFG Configuration

BUS_CFG	Firewall Enabled	CRYPTO Data Bus Master	CRYPTO Key Bus Master	CRYPTO Address Location	System Management Registers	Flash, PKA, TRNG, SRAM_MMR
0xFF	Yes	Secure	Secure	Secure(1)	Secure(1)	Secure(1)
0xF9	No	Non-secure	Non-secure	Non-secure(2)	Non-secure(2)	Non-secure(2)

(1) Registers located in 0x58000000 to 0x5FFFFFFF range
(2) Registers located in 0x40000000 to 0x4FFFFFFF range

The system management registers affected by the BUS_CFG configuration are all the registers in the apertures listed in Table 3-5 **except** the ones in the exemption column. Exempted registers are still accessible in the Non-secure offset but all others are secured.

Table 3-5. System Management Registers

Aperture	Exemption	Non-secure Offset	Secure Alias Offset
PRCM	RESETSECDMA	0x40082000	0x58082000
	RESETGPIO		
	RESETGPT		
	RESETI2C		
	RESETUART		
	RESETSPI		
	RESETI2S		
	RAMRETEN		
	NVMNSCADDR		
	NVMNSADDR		
	SRAMNSCADDR		
	SRAMNSADDR		
	BUSSECCFG		
	CPULOCK		
	MCUSRAMCFG		
	SECDMACLKGS		
	SECDMACLKGDS		
	SECDMACLKGR		
AON_PMCTL	AUXSCECLK	0x40090000	0x58090000
	RAMCFG		
	PWRSTAT		
	SHUTDOWN		
	RECHARGESTAT		
AON_RTC	CTL	0x40092000	0x58092000
	SEC		
	SUBSEC		
AUX_SYSIF	RTCSUBSECINC0	0x400C6000	0x580C6000
	RTCSUBSECINC1		
	RTCSUBSECINCCTL		
AUX_DDI0_OSC		0x400CA000	0x580CA000
AUX_SCE	NONSECDDIACC0	0x400E1000	0x580E1000
	NONSECDDIACC1		
	NONSECDDIACC2		
	NONSECDDIACC3		

Table 3-6 shows the apertures which have all registers mapped to the Secure address space.

Table 3-6. Apertures with all Registers Mapped to Secure Address Space

Aperture	Non-secure Offset	Secure Alias Offset	Size
CRYPTO	0x40024000	0x58024000	2 KB
PKA	0x40025000	0x58025000	4 KB
PKA_RAM	0x40026000	0x58026000	2 KB
PKA_INT	0x40027000	0x58027000	4 KB
TRNG	0x40028000	0x58028000	8 KB
FLASH(1)		0x58030000	8 KB

Table 3-6. Apertures with all Registers Mapped to Secure Address Space (continued)

Aperture	Non-secure Offset	Secure Alias Offset	Size
NVMNW[1]		0x58032000	4 KB
SRAM_MMR[1]		0x58035000	20 KB
AUX_DDI0_OSC	0x400CA000	0x580CA000	4 KB

(1) These apertures always reside in the Secure range.

3.3 Memory Map

Table 3-7. Memory Map

Aperture	Description	Base Address
FLASHMEM	Flash memory	0x00000000
SRAM	General purpose RAM	0x20000000
RFC_RAM	Command and Packet Engine RAM (CPERAM) in the RF core	0x21000000
RFC_ULLRAM	Command and Packet Engine Ultra Low Leakage RAM (CPERAM) in the RF core	0x21004000
SPI0	Serial Peripheral Interface (SPI) with master and slave capabilities	0x40000000
UART0	Universal Asynchronous Receiver/Transmitter (UART) interface	0x40001000
I2C0	Inter-Integrated Circuit (I2C) controller with master and slave capabilities	0x40002000
SPI1	Serial Peripheral Interface (SPI) with master and slave capabilities	0x40008000
SPI2	Serial Peripheral Interface (SPI) with master and slave capabilities	0x40009000
SPI3	Serial Peripheral Interface (SPI) with master and slave capabilities	0x4000A000
UART1	Universal Asynchronous Receiver/Transmitter (UART) interface	0x4000B000
UART2	Universal Asynchronous Receiver/Transmitter (UART) interface	0x4000C000
UART3	Universal Asynchronous Receiver/Transmitter (UART) interface	0x4000D000
GPT0	General Purpose Timer (GPT)	0x40010000
GPT1	General Purpose Timer (GPT)	0x40011000
GPT2	General Purpose Timer (GPT)	0x40012000
GPT3	General Purpose Timer (GPT)	0x40013000
UDMA0	Micro Direct Memory Access (µDMA) controller	0x40020000
I2S0	Inter-IC Sound (I2S) controller with I2S, LJF, RJF and DSP formats	0x40021000
GPIO	General Purpose Input/Output (GPIO) interface for controlling and reading IO status and IO event status	0x40022000
I2C1	Inter-Integrated Circuit (I2C) controller with master and slave capabilities	0x4002A000
VIMS	Versatile Instruction Memory System	0x40034000
RFC_PWR	RF core power management	0x40040000
RFC_DBELL	RF core doorbell	0x40041000
RFC_RAT	RF core radio timer	0x40043000

Table 3-7. Memory Map (continued)

Aperture	Description	Base Address
WDT	Watchdog timer	0x40080000
IOC	IO Controller (IOC)	0x40081000
EVENT	Event fabric	0x40083000
SMPH	Semaphore module	0x40084000
AON_EVENT	Always On (AON) event fabric	0x40093000
AON_IOC	Always On (AON) IO Controller (IOC)	0x40094000
AON_BATMON	Always On (AON) Battery And Temperature MONitor (BATMON)	0x40095000
AUX_SPIM	AUX Serial Peripheral Interface Master (SPIM)	0x400C1000
AUX_MAC	AUX Multiply-ACcumulate (MAC)	0x400C2000
AUX_TIMER2	AUX timer 2	0x400C3000
AUX_TDC	AUX Time to Digital Converter (TDC)	0x400C4000
AUX_EVCTL	AUX event controller	0x400C5000
AUX_TIMER01	AUX timer 0 and AUX timer 1, 16-bit timers	0x400C7000
AUX_SMPH	AUX semaphore	0x400C8000
AUX_ANAIF	AUX analog interface	0x400C9000
AUX_ADI4	Aux analog digital interface	0x400CB000
AUX_AIODIO0	AUX analog digital input output controller	0x400CC000
AUX_AIODIO1	AUX analog digital input output controller	0x400CD000
AUX_AIODIO2	AUX analog digital input output controller	0x400CE000
AUX_AIODIO3	AUX analog digital input output controller	0x400CF000
AUX_RAM	SRAM within the AUX domain, intended for AUX usage only	0x400E0000
CCFG	Customer configuration area	0x50000000
FCFG1	Factory configuration area	0x50000800
CRYPTO	Crypto core containing AES and SHA-2 accelerators	0x58024000
PKA	Public Key Accelerator (PKA)	0x58025000
PKA_RAM	SRAM within the PKA domain, intended for PKA usage only	0x58026000
PKA_INT	Integrated module which includes the PKA	0x58027000
TRNG	True Random Number Generator (TRNG)	0x58028000
FLASH	Flash sub-system registers, includes the No Wrapper (NW) flash API and EFUSE	0x58030000
NVMNW	Non-Volatile Memory No Wrapper (NVMNW) control	0x58032000
SRAM_MMR	General purpose RAM	0x58035000
PRCM	Power, Reset and Clock Management (PRCM)	0x58082000
AON_PMCTL	Always On (AON) power management controller	0x58090000
AON_RTC	Always On (AON) Real Time Clock (RTC)	0x58092000
AUX_SYSIF	AUX system interface	0x580C6000
AUX_DDI0_OSC	AUX Digital to Digital Interface (DDI)	0x580CA000
AUX_SCE	AUX Sensor Control Engine (SCE)	0x580E1000

Table 3-7. Memory Map (continued)

Aperture	Description	Base Address
CPU_ITM	Cortex-M's Instrumentation Trace Macrocell (ITM)	0xE0000000
CPU_DWT	Cortex-M's Data Watchpoint and Trace (DWT)	0xE0001000
CPU_FPB	Cortex-M's Flash Patch and Breakpoint (FPB)	0xE0002000
CPU_ICB	Cortex-M's Implementation Control Block (ICB)	0xE000E000
CPU_SYSTICK	Cortex-M's system timer (SYSTICK)	0xE000E010
CPU_NVIC	Cortex-M's Nested Vectored Interrupt Controller (NVIC)	0xE000E100
CPU_SCB	Cortex-M's System Control Block (SCB)	0xE000ED00
CPU_MPU	Cortex-M's Memory Protection Unit (MPU)	0xE000ED90
CPU_SAU	Cortex-M's Security Attribution Unit (SAU)	0xE000EDD0
CPU_DCB	Cortex-M's Debug Control Block	0xE000EDE0
CPU_SIG	Cortex-M's Software Interrupt Generator (SIG)	0xE000EF00
CPU_FPU	Cortex-M's Floating Point Unit (FPU)	0xE000EF30
CPU_TPIU	Cortex-M's Trace Port Interface Unit (TPIU)	0xE0040000

Chapter 4

Arm® Cortex®-M33 Peripherals

This chapter describes the Arm® Cortex®-M33 peripherals.

4.1 Arm® Cortex®-M33 Peripherals Introduction 276

4.1 Arm® Cortex®-M33 Peripherals Introduction

The Cortex®-M33 processor has a fixed default memory map that provides up to 4 GB of addressable memory. Figure 4-1 shows the Cortex®-M33 processor memory map.

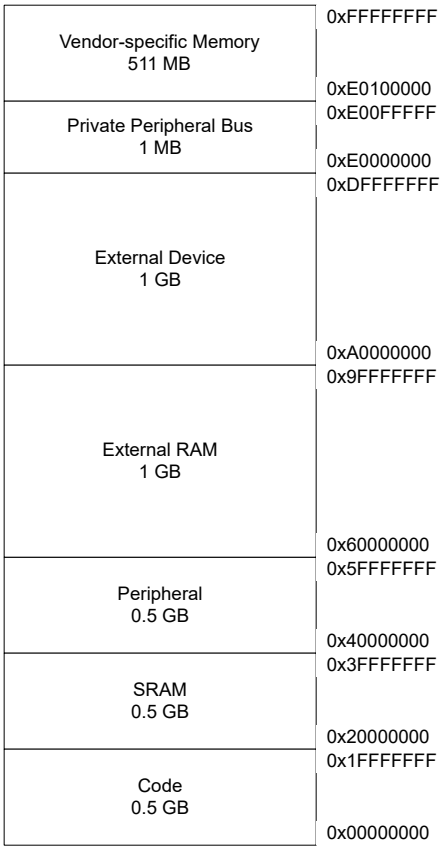


Figure 4-1. Arm® Cortex®-M33 Processor Memory Map

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral. The peripherals and their addresses are listed in Table 4-1.

Table 4-1. Core Peripheral Register Regions

Address	Core Peripheral	Link
0xE000E010 - 0xE000E01C	System Timer (SysTick)	Arm® Cortex®-M33 Device Generic User Guide
0xE000E100 - 0xE000E400	Nested Vectored Interrupt Controller (NVIC)	Arm® Cortex®-M33 Device Generic User Guide
0xE000ECFC - 0xE000ED8C	System Control Block (SCB)	Arm® Cortex®-M33 Device Generic User Guide
0xE0001000 - 0xE0001FFC	Data watchpoint and trace (DWT)	Arm® Cortex®-M33 Processor Technical Reference Manual
0xE0002000 - 0xE0002FFC	Flash Patch and Breakpoint (FPB)	Arm® Cortex®-M33 Processor Technical Reference Manual
0xE0000000 - 0xE0000FFC	Instrumentation Trace Macrocell (ITM)	Arm® Cortex®-M33 Processor Technical Reference Manual
0xE00FF000 - 0xE00FFFFC	ROM table	
0xE0040000 - 0xE0040FFC	Trace Port Interface Unit (TPIU)	Arm® Cortex®-M33 Processor Technical Reference Manual

Table 4-1. Core Peripheral Register Regions (continued)

Address	Core Peripheral	Link
0xE000ED90 - 0xE000EDC4	Memory Protection Unit (MPU)	Arm® Cortex®-M33 Device Generic User Guide
0xE000EDD0 - 0xE000EDE8	Security Attribution Unit (SAU)	Arm® Cortex®-M33 Device Generic User Guide

This page intentionally left blank.

Chapter 5

Interrupts and Events

This chapter describes interrupts and events for the CC13x4x10 and CC26x4x10 device platform and the different Arm® documentation listed in Section Related Documentation are used as reference.

5.1 Exception Model 280
5.2 Fault Handling 288
5.3 Security State Switches 290
5.4 Event Fabric 291
5.5 AON Event Fabric 295
5.6 MCU Event Fabric 295
5.7 AON Events 300
5.8 Interrupts and Events Registers 301

5.1 Exception Model

The Arm® Cortex®-M33 processor and the nested vectored interrupt controller (NVIC) prioritize and handle all exceptions in handler mode. The state of the processor is automatically stored to the stack on an exception and automatically restored from the stack at the end of the interrupt service routine (ISR). The vector is fetched in parallel to state saving, thus enabling efficient interrupt entry. The processor supports tail-chaining, which enables performance of back-to-back interrupts without the overhead of state saving and restoration.

Table 5-1 lists all exception types. Software can set eight priority levels on all the exceptions except reset and hard fault.

Internally, the highest user-programmable priority (0) is treated as third priority, after a reset, and a hard fault, in that order.

Note

The default priority is 0 for all the programmable priorities.

CAUTION

After a write to clear an interrupt source, it may take several processor cycles for the NVIC to detect the interrupt source de-assertion due to the write buffer. Thus, if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while the NVIC detects the interrupt as still asserted, causing the interrupt handler to be re-entered errantly. This situation can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read from the same address after the write to clear the interrupt source (and flush the write buffer).

For more information on exceptions and interrupts, see the Arm® Cortex®-M33 Device Generic User Guide.

5.1.1 Exception States

Each exception is in one of the following states:

- **Inactive**: The exception is not active and not pending.
- **Pending**: The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- **Active**: An exception is being serviced by the processor but has not completed. An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.
- **Active and Pending**: The exception is being serviced by the processor, and there is a pending exception from the same source.

5.1.2 Exception Types

The exception types are:

- **Reset**: The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When either power-on or warm reset is de-asserted, execution restarts from the address provided by the reset entry in the ROM vector table. Execution restarts as privileged execution in Secure state in Thread mode. This exception is not banked between security states.
- **NMI**: A Non-Maskable Interrupt (NMI) can be signaled by a peripheral or triggered by software. It is permanently enabled and has a fixed priority of -2. NMI can only be preempted by reset and, when it is Non-secure, by a Secure HardFault. If CPU_SCB:AIRCR.BFHFNMINS = 0, then the NMI is Secure. If CPU_SCB:AIRCR.BFHFNMINS = 1, then NMI is Non-secure.
- **HardFault**: A HardFault is an exception that occurs because of an error during normal or exception processing. HardFaults have a fixed priority of at least -1, meaning they have higher priority than any exception with configurable priority. This exception is not banked between security states.

  If CPU_SCB:AIRCR.BFHFNMINS = 0, HardFault handles all faults that are unable to preempt the current execution. The HardFault handler is always Secure.

  If CPU_SCB:AIRCR.BFHFNMINS = 1, HardFault handles faults that target Non-secure state that are unable to preempt the current execution.

  HardFaults that specifically target the Secure state when CPU_SCB:AIRCR.BFHFNMINS is set to 1 have a priority of -3 to ensure they can preempt any execution. A Secure HardFault at priority -3 is only enabled when CPU_SCB:AIRCR.BFHFNMINS is set to 1. Secure HardFault handles Secure faults that are unable to preempt current execution.
- **MemManage**: A MemManage fault is an exception that occurs because of a memory protection violation, compared to the MPU or the fixed memory protection constraints, for both instruction and data memory transactions. This fault is always used to abort instruction accesses to Execute Never (XN) memory regions. This exception is banked between security states.
- **BusFault**: A BusFault is an exception that occurs because of a memory-related violation for an instruction or data memory transaction. This might be from an error that is detected on a bus in the memory system. This exception is not banked between security states.

  If CPU_SCB:AIRCR.BFHFNMINS = 0, BusFaults target the Secure state

  If CPU_SCB:AIRCR.BFHFNMINS = 1, BusFaults target the Non-secure state
- **UsageFault**: A UsageFault is an exception that occurs because of a fault related to instruction execution. This includes:

  – An undefined instruction
  – An illegal unaligned access
  – Invalid state on instruction execution
  – An error on exception return

  The following can cause a usage fault:

  – An unaligned address on word and halfword memory access
  – Division by zero

  This exception is banked between security states.
- **SecureFault**: This exception is triggered by the various security checks that are performed. It is triggered, for example, when jumping from Non-secure code to an address in Secure code that is not marked as a valid entry point. Most systems choose to treat a Secure fault as a terminal condition that either halts or restarts the system. Any other handling of the SecureFault must be checked carefully to make sure that it does not inadvertently introduce a security vulnerability. SecureFaults always target the Secure state.
- **SVCall**: A Supervisor Call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers. This exception is banked between security states.

- **DebugMonitor**: A DebugMonitor exception. If halting debug is disabled and the debug monitor is enabled, a debug event causes a debug monitor exception when the group priority of the debug monitor exception is greater than the current execution priority.
- **PendSV:** PendSV is an asynchronous request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. This exception is banked between Security states.
- **SysTick**: A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as a system tick. This exception is banked between Security states.
- **Interrupt (IRQ)**: An interrupt, or IRQ, is an exception signaled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. This exception is not banked between security states. Secure code can assign each interrupt to Secure or Non-secure state. By default all interrupts are assigned to Secure state.

For an asynchronous exception, other than reset, the processor can execute extra instructions between the moment the exception is triggered and the moment the processor enters the exception handler. Privileged software can disable the exceptions that have configurable priority, as shown in the table above. An exception that targets Secure state cannot be disabled by Non-secure code.

Table 5-1. Exception Types

Exception Number	IRQ number	Exception Type	Priority	Vector Address	Activation
1	-	Reset	-4, the highest	0x00000004	Asynchronous
2	-14	NMI	-2	0x00000008	Asynchronous
3	-13	Secure HardFault when CPU_SCB:AIRCR.BFHFNMINS is 1	-3	0x0000000C	Synchronous
		Secure HardFault when CPU_SCB:AIRCR.BFHFNMINS is 0	-1		
		HardFault	-1		
4	-12	MemManage	Configurable	0x00000010	Synchronous
7	-9	SecureFault	Configurable	0x0000001C	Synchronous
8-10	-	Reserved	-	-	-
11	-5	SVCall	Configurable	0x0000002C	Synchronous
12	-4	DebugMonitor	Configurable	0x00000030	Synchronous
13	-	Reserved	-	-	-
14	-2	PendSV	Configurable	0x00000038	Asynchronous
15	-1	SysTick	Configurable	0x0000003C	Asynchronous
16 and above	0 and above	Interrupt (IRQ)	Configurable	0x00000040 and above. Increasing in steps of 4	Asynchronous

Note

To simplify the software layer, the CMSIS only uses IRQ numbers. It uses negative values for exceptions other than interrupts. The IPSR returns the Exception number

5.1.3 Exception Handlers

The exception handlers are the following:

- **Interrupt Service Routines (ISRs)**: Interrupts IRQ0-IRQ46 are the exceptions that are handled by ISRs. Each interrupt is configured by Secure software in Secure or Non-secure state, using CPU_NVIC:ITNS0 and CPU_NVIC:ITNS1.
- **Fault Handler**: The fault handler handles the following exceptions:
  - HardFault
  - MemManage
  - BusFault
  - UsageFault
  - SecureFault

  There can be separate MemManage and UsageFault handlers in Secure and Non-secure state. The CPU_SCB:AIRCR.BFHFNMINS bit controls the target state for HardFault and BusFault. SecureFault always targets Secure state.
- **System Handlers**: The system handlers handle the following system exceptions:
  - NMI
  - PendSV
  - SVCall
  - SysTick

Most system handlers can be banked with separate handlers between Secure and Non-secure state. The CPU_SCB:AIRCR.BFHFNMINS bit controls the target state for NMI.

5.1.4 Vector Table

The Vector Table Offset Register (CPU_SCB:VTOR) in the System Control Block (SCB) determines the starting address of the vector table. The VTOR is banked so there is a VTOR_S and a VTOR_NS. The initial values of VTOR_Sand VTOR_NS are 0. The vector table used depends on the target state of the exception. For exceptions targeting the Secure state, VTOR_S is used. For exceptions targeting the Non-secure state, VTOR_NS is used.

Table 5-2 shows the order of the exception vectors in the Secure and Non-secure vector tables. The least-significant bit of each vector is 1, indicating that the exception handler is written in Thumb® code.

Table 5-2. Vector Table With Security Extension

Exception Number	IRQ Number	Secure Vector	Non-secure Vector	Offset
62	46	IRQ46	IRQ46	0xF8
.			.	.
.			.	.
.			.	.
18	2	IRQ2	IRQ2	0x48
17	1	IRQ1	IRQ1	0x44
16	0	IRQ0	IRQ0	0x40
15	-1	SysTick_S	SysTick_NS	0x3C
14	-2	PendSV_S	PendSV_NS	0x38
13		Reserved	Reserved	0x30
12	-3	DebugMonitor	DebugMonitor	
11	-5	SVCall_S	SVCall_NS	0x2C
10		Reserved	Reserved	
9				
8				
7	-9	SecureFault		0x1C
6	-11	UsageFault_S	UsageFault_NS	0x18
5	-12	BusFault_S	BusFault_NS	0x14
4	-13	MemManage_S	MemManage_NS	0x10
3	-13	HardFault_S	HardFault_NS	0x0C
2	-14	NMI_S	NMI_NS	0x08
1		Reset		0x04
		Initial SP Value		0x00

Because reset always targets Secure state, the Non-secure reset and Non-secure initial SP value are ignored by the hardware.

5.1.5 Exception Priorities

All exceptions have an assigned priority that is used to control both pre-emption and prioritization between pending exceptions. A lower priority value indicates a higher priority. You can configure priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0.

Note

Configurable priorities are in the range 0-255. The Reset, HardFault, and NMI exceptions, with fixed negative priority values always have higher priority than any other exception.

For configurable priority exceptions, the target Security state alsoaffects the programmed priority. Depending on the value of CPU_SCB:AIRCR.PRIS, the priority can be extended

In Table 5-3, the values in columns 2 and 3 must match, and increase from zero in increments of 32. The values in column 4 start from 128 and increase in increments of 16.

Table 5-3. Extended Priority

Priority Value [7:5]	Secure Priority	Non-secure priority when CPU_SCB:AIRCR.PRIS = 0	Non-secure priority when CPU_SCB:AIRCR.PRIS = 1
0	0	0	128
1	32	32	144
2	64	64	160
3	96	96	176
4	128	128	192
5	160	160	208
6	192	192	224
7	224	224	240

Assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

5.1.6 Interrupt Priority Grouping

The NVIC supports software assigned priority levels. A priority level from 0 to 8 can be assign to an interrupt by writing to the most significant bits of the PRI_N field in the CPU_NVIC:IPRn register corrsponding to the interrupt, see Section 2.5.4.

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields, an upper field that defines the group priority, and a lower field that defines a subpriority within the group.

Only the group priority determines pre-emption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not pre-empt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

If a pending Secure exception and a pending Non-secure exception both have the same group priority field value, the same subpriority field value, and the same exception number, the Secure exception takes precedence.

5.1.7 Exception Entry and Return

Descriptions of exception handling use the following terms.

- **Preemption**: An exception can preempt the current execution if its priority is higher than the current execution priority. When one exception preempts another, the exceptions are called nested exceptions.
- **Return**: This occurs when the exception handler is completed. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred.
- **Tail-Chaining**: This mechanism speeds up exception servicing. On completion of an exception handler or during the return operation, if there is a pending exception that meets the requirements for exception entry, then the stack pop is skipped and control transfers directly to the new exception handler.
- **Late Arriving Interrupts**: This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving may be affected by the late arrival depending on the stacking requirements of the original exception and the late-arriving exception. On return from the exception handler of the late-arriving exception, the normal tailchaining rules apply.

5.1.7.1 Exception Entry

Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode, or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means that the exception has higher priority than any limits set by the mask registers. An exception with lower priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as stacking and the structure of the data stacked is referred as the stack frame.

The Cortex-M33 processor can automatically stack the architected floating-point state on exception entry.

5.1.7.2 Exception Return

Exception return occurs when the processor is in Handler mode and execution of one of the following instructions attempts to set the PC to an EXC_RETURN value:

- A POP or LDM instruction that loads the PC
- An LDR instruction that loads the PC
- A BX instruction using any register

The processor saves an EXC_RETURN value to the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. When the processor loads a value matching this pattern to the PC it detects that the operation is not a normal branch operation and, instead, that the exception is complete. As a result, it starts the exception return sequence. Bits[6:0] of the EXC_RETURN value indicate the required return stack, processor mode, Security state, and stack frame as shown in Table 5-4.

Table 5-4. Exception Return Behavior

Bits	Name	Function
[31:24]	PREFIX	Indicates that this is an EXC_RETURN value. This field reads as 0b11111111
[23:7]	-	Reserved, RES1.
[6]	S	Indicates whether registers have been pushed to a Secure or Non-secure stack. 0: Non-secure stack used 1: Secure stack used.
[5]	DCRS	Indicates whether the default stacking rules apply, or whether the called registers are already on the stack. 0:Stacking of the called saved registers is skipped 1: Default rules for stacking the called registers are followed
[4]	FType	In a PE with the Main and Floating-point Extensions: 0: The PE allocated space on the stack for FP context 1: The PE did not allocate space on the stack for FP context. In a PE without the Floating-point Extension, this bit is Reserved, RES1
[3]	Mode	Indicates the mode that was stacked from. 0: Handler mode 1: Thread mode
[2]	SPSEL	Indicates which stack contains the exception stack frame. 0: Main stack pointer 1: Process stack pointer
[1]	-	Reserved, RES0
[0]	ES	Indicates the Security state the exception was taken to. 0: Non-secure 1: Secure

5.2 Fault Handling

Faults can occur on instruction fetches, instruction execution, and data accesses. When a fault occurs, information about the cause of the fault is recorded in various registers, according to the type of fault. Faults are a subset of the exceptions.

Faults are generated by:

- A bus error on:
  - An instruction fetch or vector table load
  - A data access
- An internally-detected error such as an undefined instruction
- Attempting to execute an instruction from a memory region marked as Execute Never (XN)
- A privilege violation or an attempt to access an unmanaged region causing an MPU fault
- A security violation

5.2.1 Fault Types

Table 5-5 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred.

Table 5-5. Faults

Fault	Handler	Bit Name	Fault Status Register
Bus error on a vector read	HardFault	VECTTBL	HardFault Status Register (HFSR)
Fault escalated to a hard fault		FORCED	
MPU or default memory map mismatch:	MemManage	-	MemManage Fault Status Register (MMFSR)
On instruction access		IACCVIOL	
On data access		DACCVIOL	
During exception stacking		MSTKERR	
During exception unstacking		MUNSKERR	
During lazy floating-point state preservation		MLSPERR	
Bus error:	BusFault	-	-
During exception stacking		STKERR	BusFault Status Register (BFSR)
During exception unstacking		UNSTKERR	
During instruction prefetch		IBUSERR	
During lazy floating-point state preservation		LSPERR	
Precise data bus error		PRECISERR	
Imprecise data bus error		IMPRECISERR	
Attempt to access a coprocessor	UsageFault	NOCP	UsageFault Status Register (UFSR)
Undefined instruction		UNDEFINSTR	
Attempt to enter an invalid instruction set state		INVSTATE	
Invalid EXC_RETURN value		INVPC	
Illegal unaligned load or store		UNALIGNED	
Stack overflow flag		STKOF	
Divide By 0		DIVBYZERO	

Table 5-5. Faults (continued)

Fault	Handler	Bit Name	Fault Status Register
Lazy state error flag	SecureFault	LSERR	SecureFault Status Register (SFSR)
Lazy state preservation error flag		LSPERR	
Invalid transition flag		INVTRAN	
Attribution unit violation flag		AUVIOL	
Invalid exception return flag		INVER	
Invalid integrity signature flag		INVIS	
Invalid entry point		INVEP	

5.2.2 Fault Escalation and Hard Faults

All fault exceptions other than HardFault have configurable exception priority. Software can disable execution of the handlers for these faults.

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler.

In some situations, a fault with configurable priority is treated as a HardFault. This is called priority escalation, and the fault is described as escalated to HardFault. Escalation to HardFault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to HardFault occurs because a fault handler cannot preempt itself; it must have the same priority as the current execution priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This is because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a BusFault occurs during a stack push when entering a BusFault handler, the BusFault does not escalate to a HardFault. This means that if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

BusFaults and fixed priority exceptions can be designated as Secure or Non-secure under the control of CPU_SCB:AIRCR.BFHFNMINS. When AIRCR.BFHFMNINS is set to:

- 0: BusFaults and fixed priority exceptions are designated as Secure. The exceptions retain the prioritization of HardFault at -1 and NMI at -2.
- 1: BusFaults and fixed priority exceptions are designated as Non-secure. In this case, Secure HardFault is introduced at priority -3 to ensure that faults that target Secure state are recognized.

The Non-secure state cannot inhibit BusFaults and fixed priority exceptions which target Secure state. Therefore when faults and fixed priority exceptions are Secure, Non-secure FAULTMASK (FAULTMASK_NS) only inhibits programmable priority exceptions, making it equivalent to Non-secure PRIMASK (PRIMASK_NS). See Table 2-1 for more info regarding FAULTMASK and PRIMASK.

Non-secure programmable priority exceptions are mapped to the regular priority range 0-255, if CPU_SCB:AIRCR.PRIS is clear. Non-secure programmable priority exceptions are mapped to the bottom half the regular priority range, 128-255, if AIRCR.PRIS is set to 1. Therefore the FAULTMASK_NS sets the execution priority to 0x0 or 0x80, according to AIRCR.PRIS, to mask the Non-secure programmable priority exception only.

When BusFaults and fixed priority exceptions are Secure, FAULTMASK_S sets execution priority to -1 to inhibit everything up to and including HardFault.

When BusFaults and fixed priority exceptions are designated as Non-secure, FAULTMASK_NS boosts priority to -1 to inhibit everything up to Non-secure HardFault at priority -1, while FAULTMASK_S boosts priority to -3 to inhibit all faults and fixed priority exceptions including the Secure HardFault at priority -3.

Note

Only Reset can preempt the fixed priority Secure HardFault when CPU_SCB:AIRCR.BFHFNMINS is set to 1. A Secure HardFault when AIRCR.BFHFNMINS is set to 1 can preempt any exception other than Reset. A

Secure HardFault when AIRCR.BFHFNMINS is set to 0 can preempt any exception other than Reset, NMI, or another HardFault.

5.2.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For BusFaults, MemManage faults, and SecureFaults, the fault address register indicates the address that is accessed by the operation that caused the fault.

The processor has two physical fault address registers. One shared between the MMFAR_S (CPU_SCB:MMFAR), CPU_SAU:SFAR, and CPU_SCB:BFAR (only if AIRCR.BFHFNMINS is set to 0), and the other shared between the MMFAR_NS (CPU_SCB:MMFAR) and CPU_SCB:BFAR (only if AIRCR.BFHFNMINS is set to 1). These are targeted by Secure and Non-secure faults respectively.

For each physical fault address register, it is only possible to report the address of one fault at a time.

Each fault address register is updated when one of the *FARVALID bits is set for their respective faults in the associated *FSR register. BFARVALID is located in CPU_SCB:CFSR and SFARVALID is located in CPU_SAU:SFSR. Any fault that targets a fault address register with one of its *FARVALID bits already set does not update the fault address. The *FARVALID bits must be cleared before another fault address can be reported.

5.2.4 Lockup

The processor enters a lockup state if a fault occurs when it cannot be serviced or escalated. When the processor is in lockup state, it does not execute any instructions.

The processor remains in lockup state until either:

- It is reset
- Preemption by a higher priority exception occurs
- It is halted by a debugger

Note

If lockup state occurs from a Secure HardFault when AIRCR.BFHFNMINS is set to 1 or the NMI handler, a subsequent NMI does not cause the processor to leave lockup state

5.3 Security State Switches

Table 5-6 presents the possible security transitions, the instructions that can cause them, and any faults that may be generated.Table 5-6

Table 5-6. Security State Transitions

Current Security State	Security Attribute of the Branch Target Address	Security State Change
Secure	Non-secure	Change to Non-escure state if the branch was a BXNS or BLXNS instruction with the LSB of the target address set to 0. Otherwise, a SecureFault is generated.
Non-secure	Secure and Non-secure callable	Change to Secure state if the branch target address contains an SG instruction. If the target address does not contain an SG a SecureFault is generated
Non-Secure	Secure and not Non-secure callable	A SecureFault is generated

Secure software can call a Non-secure function using the BLXNS instruction. When this happens, the LR is set to a special value called FNC_RETURN, and the return address and XPSR is saved onto the Secure stack. Return from Non-secure state to Secure state is triggered when one of the following instructions attempts to set the PC to an FNC_RETURN value:

- A POP or LDM instruction that loads the PC
- An LDR instruction that loads the PC
- A BX instruction using any register

When a return from Non-secure state to Secure state occurs the processor restores the program counter and XPSR from the Secure stack.

Any scenario not listed in the table triggers a SecureFault. For example:

- Sequential instructions that cross security attributes from Secure to Non-secure.
  - A 32-bit instruction fetch that crosses regions with different security attributes.

5.4 Event Fabric

5.4.1 Introduction

The event fabric is a combinational router between event sources and event subscribers. The event inputs are routed to a central event-bus where a subscriber can select the appropriate events and output those as inputs to peripherals. Figure 5-1 shows the general concept of the event fabric. The event fabric is strictly combinational logic. Because this chapter provides only a general overview of the event fabric and the system CPU, NMI, and Freeze subscriber, refer to the specific peripheral chapters in this user's guide to understand how to use and configure the events for the different subscribers and peripherals.

Most of the events (signals) are statically routed, meaning that only a small number of configurable output lines go to the event subscribers. A configurable output line from a subscriber can choose from a list of several input events available to the specific subscriber in question. Subscribers output event signaling identical to input signaling. That is, events are simply passed through the event fabric as presented to the input ports. Possible event types include system hardware interrupts, software programmable interrupts, and µDMA triggers. All event inputs are considered level-triggered events active high. Events like µDMA triggers may or may not be level-type signals.

Figure 5-1 shows a simple illustration of the event fabric concept. Clearly the event fabric is not a peripheral in itself, but rather a block of routing between the peripherals and more. The lines that have configurable inputs are controlled by selection registers that are connected to a MUX, which forwards the selected input in the subscriber to the peripherals.

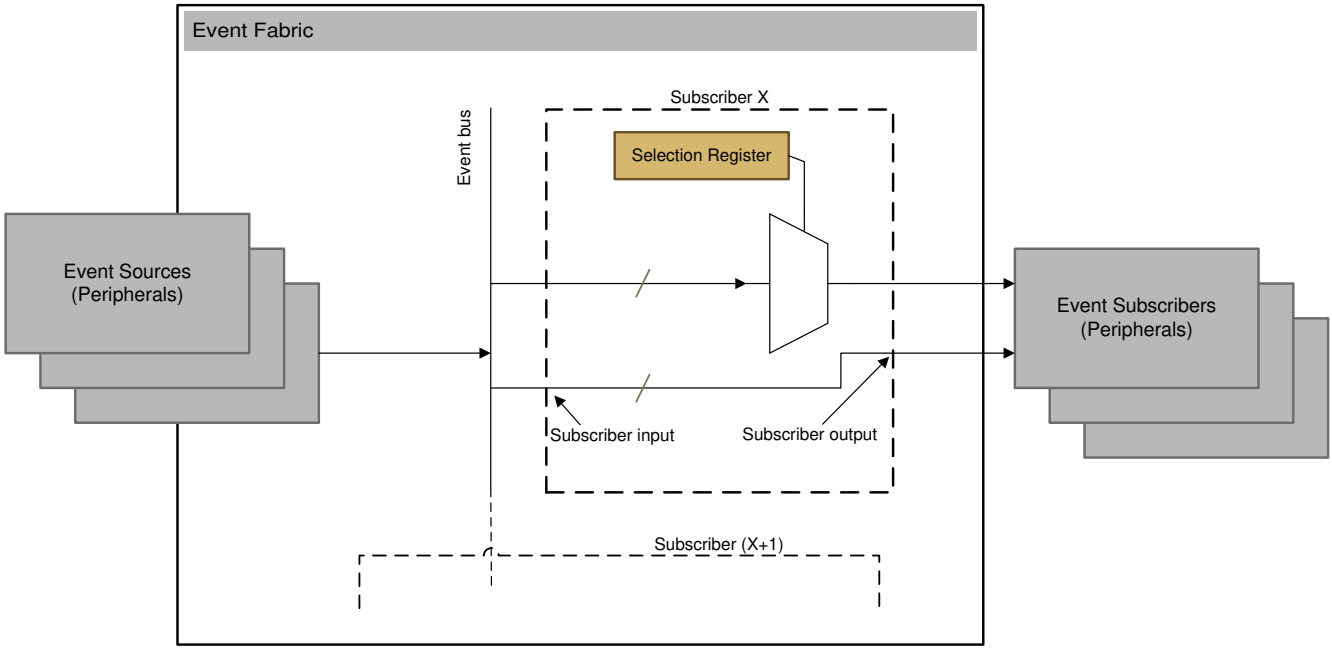


Figure 5-1. Event Fabric Concept

5.4.2 Event Fabric Overview

There are two main event fabric blocks in the CC13x4x10 and CC26x4x10 devices. One in the MCU power domain (MCU event fabric) and the other in the AON power domain (AON event fabric). Figure 5-2 shows a simplified overview of the two modules together. The MCU event fabric is one of the subscribers to the AON event fabric.

The AUX Domain Sensor Controller and peripherals has an internal simplified event fabric for routing of events internally and as inputs to the AON event fabric and MCU event fabric. See Section 20.6 for details.

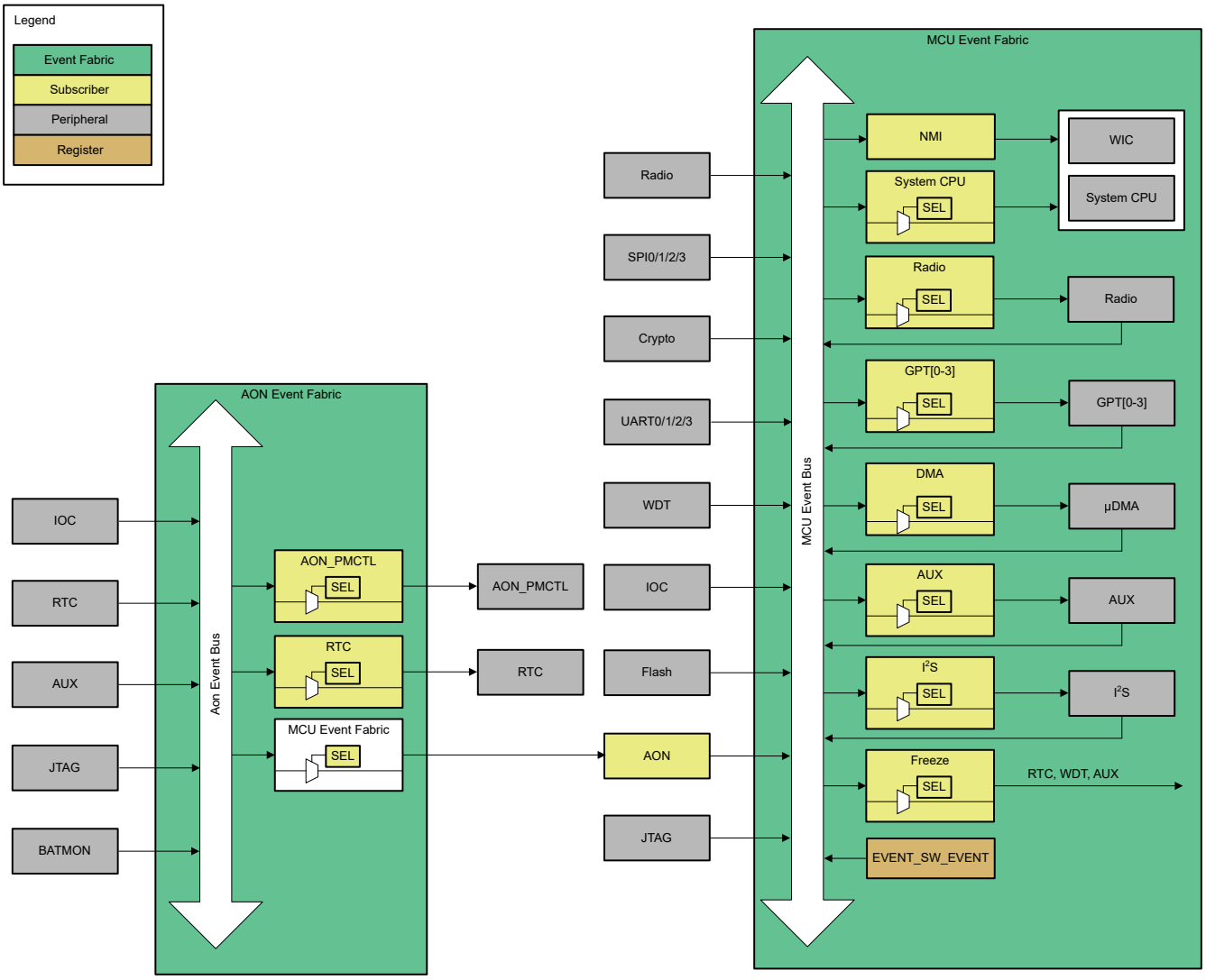


Figure 5-2. Event Fabric Overview (Simplified)

5.4.2.1 Registers

The event fabric has two types of registers. The first type, a configuration register, is used to control and report the selection settings for a subscriber output. For each subscriber output, an address is mapped for a read register that contains a value representing the selection of the input event currently set for that subscriber output. For nonconfigurable outputs, only a read-only register is implemented. A read to that address returns the static, predefined value. The second type of register in the event fabric, of which there is only one, is an operational register named SWEV. This register sets and clears any of the four software events.

The AON event fabric is controlled through a series of registers residing in the MCU power domain. An AON-MCU interface block in the AON domain shadows these registers, thereby providing them for the AON block when the MCU is in power-down states.

5.5 AON Event Fabric

The AON event fabric resides in the AON voltage domain.

5.5.1 Common Input Event List

Section 5.6.1 lists the input events for the AON event fabric. The sources for these events are considered level-triggered active high.

5.5.2 Event Subscribers

There are three subscribers in the AON event fabric as can be seen in Figure 5-2. The first subscriber is the MCU event fabric, which resides in the MCU power domain. The other two subscribers, the AON_PMCTL and RTC, both reside in the AON voltage domain and are presented in the following subsections.

5.5.2.1 AON Power Management Controller (AON_PMCTL)

The AON_PMCTL subscriber has 8 programmable events in AON event fabric, which are ORed together to form a single wake-up event to AON_PMCTL. This wake-up event is used to trigger the wakeup of the MCU domain from Standby mode. Any of the events listed in Table 5-9 can be chosen as input by selecting the appropriate event ID. By default, these IDs are set to 63 (NULL, no event), where the lines always stay logic low.

5.5.2.2 Real-Time Clock

The RTC has a programmable event, which can be configured in the RTCSEL register, and a fixed event with ID 46 (Channel 2 clear – from AUX).

5.5.2.3 MCU Event Fabric

Five output events from the AON event fabric are routed as inputs to the MCU event fabric. These events are:

- AON programmable 0
- AON programmable 1
- AON programmable 2
- AON edge detect
- AON RTC

There are three programmable lines from which any of the input events from Section 5.5.1can be chosen. This can be set in the AON_EVENT:EVTOMCUSEL register.

5.6 MCU Event Fabric

The MCU event fabric resides in the MCU power domain and routes signals between most of the peripherals and different internal blocks. Only a few of the subscribers in the MCU event fabric are described in this section. For more information on the remaining subscribers, refer to the specific peripheral chapters for the appropriate consumer (peripheral) for that specific subscriber.

5.6.1 Common Input Event List

Table 5-7 lists the input events for the MCU event fabric. The sources for these events are considered level-triggered active high.

Table 5-7. MCU Events

Event No.	Name	Description
0x0	NONE	Always inactive
0x1 to 0x3	AON_PROG0 to AON_PROG2	AON programmable event n, n=0..2
0x4	AON_GPIO_EDGE	Edge detect event from IOC
0x5	BATMON_COMB	Combined event from battery monitor
0x6	OSC_COMB	Combined event from Oscillator control
0x7	AON_RTC_COMB	Event from AON_RTC
0x8	I2S_IRQ	Interrupt event from I2S
0x9	I2C_IRQ	Interrupt event from I2C

Table 5-7. MCU Events (continued)

Event No.	Name	Description
0xA	AON_AUX_SWEV0	AUX Software event 0
0xB	AUX_COMB	AUX combined event
0xC	GPT2A	GPT2A interrupt event
0xD	GPT2B	GPT2B interrupt event
0xE	GPT3A	GPT3A interrupt event
0xF	GPT3B	GPT3B interrupt event
0x10	GPT0A	GPT0A interrupt event
0x11	GPT0B	GPT0B interrupt event
0x12	GPT1A	GPT1A interrupt event
0x13	GPT1B	GPT1B interrupt event
0x14	DMA_CH0_DONE	DMA done for software tiggered UDMA channel 0
0x15	FLASH	FLASH controller error event
0x16	DMA_CH18_DONE	DMA done for software tiggered UDMA channel 18
0x18	WDT_IRQ	Watchdog interrupt event
0x19	RFC_CMD_ACK	RFC Doorbell Command Acknowledgement Interrupt
0x1A	RFC_HW_COMB	Combined RFC hardware interrupt
0x1B	RFC_CPE_0	Combined Interrupt for CPE Generated events
0x1C to 0x1D	AUX_SWEV0 to AUX_SWEV1	AUX software event n, n=0..1
0x1E	RFC_CPE_1	Combined Interrupt for CPE Generated events
0x1F	PKA_IRQ	PKA Interrupt event
0x22	SPI0_COMB	SPI0 combined interrupt
0x23	SPI1_COMB	SPI1 combined interrupt
0x24	UART0_COMB	UART0 combined interrupt
0x25	UART1_COMB	UART1 combined interrupt
0x26	DMA_ERR	DMA bus error
0x27	DMA_DONE_COMB	Combined DMA done
0x28	SPI0_RX_DMABREQ	SPI0 RX DMA burst request
0x29	SPI0_RX_DMASREQ	SPI0 RX DMA single request
0x2A	SPI0_TX_DMABREQ	SPI0 TX DMA burst request
0x2B	SPI0_TX_DMASREQ	SPI0 TX DMA single request
0x2C	SPI1_RX_DMABREQ	SPI1 RX DMA burst request
0x2D	SPI1_RX_DMASREQ	SPI1 RX DMA single request
0x2E	SPI1_TX_DMABREQ	SPI1 TX DMA burst request
0x2F	SPI1_TX_DMASREQ	SPI1 TX DMA single request
0x30	UART0_RX_DMABREQ	UART0 RX DMA burst request
0x31	UART0_RX_DMASREQ	UART0 RX DMA single request
0x32	UART0_TX_DMABREQ	UART0 TX DMA burst request
0x33	UART0_TX_DMASREQ	UART0 TX DMA single request
0x34	UART1_RX_DMABREQ	UART1 RX DMA burst request
0x35	UART1_RX_DMASREQ	UART1 RX DMA single request
0x36	UART1_TX_DMABREQ	UART1 TX DMA burst request
0x37	UART1_TX_DMASREQ	UART1 TX DMA single request
0x38	AUX_TIMER2_EV0	AUX Timer2 event 0
0x39	AUX_TIMER2_EV1	AUX Timer2 event 1
0x3A	AUX_TIMER2_EV2	AUX Timer2 event 2

Table 5-7. MCU Events (continued)

Event No.	Name	Description
0x3B	AUX_TIMER2_EV3	AUX Timer2 event 3
0x3C	AUX_TIMER2_PULSE	AUX Timer2 pulse
0x3D	GPT0A_CMP	GPT0A compare event
0x3E	GPT0B_CMP	GPT0B compare event
0x3F	GPT1A_CMP	GPT1A compare event
0x40	GPT1B_CMP	GPT1B compare event
0x41	GPT2A_CMP	GPT2A compare event
0x42	GPT2B_CMP	GPT2B compare event
0x43	GPT3A_CMP	GPT3A compare event
0x44	GPT3B_CMP	GPT3B compare event
0x45 to 0x48	TIE_LOW	Not used tied to 0
0x4D	GPT0A_DMABREQ	GPT0A DMA trigger event
0x4E	GPT0B_DMABREQ	GPT0B DMA trigger event
0x4F	GPT1A_DMABREQ	GPT1A DMA trigger event
0x50	GPT1B_DMABREQ	GPT1B DMA trigger event
0x51	GPT2A_DMABREQ	GPT2A DMA trigger event
0x52	GPT2B_DMABREQ	GPT2B DMA trigger event
0x53	GPT3A_DMABREQ	GPT3A DMA trigger event
0x54	GPT3B_DMABREQ	GPT3B DMA trigger event
0x55 to 0x5C	PORT_EVENT0 to PORT_EVENT7	Port capture event from IOC, n=0..7
0x5D	CRYPTO_RESULT_AVAIL_IRQ	CRYPTO result available interupt event
0x5E	CRYPTO_DMA_DONE_IRQ	CRYPTO DMA input done event
0x63	WDT_NMI	Watchdog non maskable interrupt event
0x64 to 0x67	SWEV0 to SWEV3	Software event n, n=0..3
0x68	TRNG_IRQ	TRNG Interrupt event
0x69	AUX_AON_WU_EV	AON wakeup event
0x6A	AUX_COMPA	AUX Compare A event
0x6B	AUX_COMPB	AUX Compare B event
0x6C	AUX_TDC_DONE	AUX TDC measurement done event
0x6D to 0x6E	AUX_TIMER0_EV to AUX_TIMER1_EV	AUX timer n event, n=0..1
0x6F	AUX_SMPH_AUTOTAKE_DONE	Autotake event from AUX semaphore
0x70	AUX_ADC_DONE	AUX ADC done
0x71	AUX_ADC_FIFO_ALMOST_FULL	AUX ADC FIFO watermark event
0x72	AUX_OBSMUX0	Loopback of OBSMUX0 through AUX
0x73	AUX_ADC_IRQ	AUX ADC interrupt event
0x74	AUX_SW_DMABREQ	DMA sofware trigger from AUX
0x75	AUX_DMASREQ	DMA single request event from AUX
0x76	AUX_DMABREQ	DMA burst request event from AUX
0x77	AON_RTC_UPD	RTC periodic event controlled by AON_RTC:CTL
0x78	CPU_HALTED	CPU halted
0x79	ALWAYS_ACTIVE	Always asserted
0x7A	SPI2_COMB	SPI2 combined interrupt
0x7B	SPI3_COMB	SPI3 combined interrupt
0x7C	UART2_COMB	UART2 combined interrupt
0x7D	UART3_COMB	UART3 combined interrupt

Table 5-7. MCU Events (continued)

Event No.	Name	Description
0x7E	I2C1_IRQ	Interrupt event from I2C1
0x7F	SPI2_RX_DMABREQ	SPI2 RX DMA burst request
0x80	SPI2_RX_DMASREQ	SPI2 RX DMA single request
0x81	SPI2_TX_DMABREQ	SPI2 TX DMA burst request
0x82	SPI2_TX_DMASREQ	SPI2 TX DMA single request
0x83	SPI3_RX_DMABREQ	SPI3 RX DMA burst request
0x84	SPI3_RX_DMASREQ	SPI3 RX DMA single request
0x85	SPI3_TX_DMABREQ	SPI3 TX DMA burst request
0x86	SPI3_TX_DMASREQ	SPI3 TX DMA single request
0x87	UART2_RX_DMABREQ	UART2 RX DMA burst request
0x88	UART2_RX_DMASREQ	UART2 RX DMA single request
0x89	UART2_TX_DMABREQ	UART2 TX DMA burst request
0x8A	UART2_TX_DMASREQ	UART2 TX DMA single request
0x8B	UART3_RX_DMABREQ	UART3 RX DMA burst request
0x8C	UART3_RX_DMASREQ	UART3 RX DMA single request
0x8D	UART3_TX_DMABREQ	UART3 TX DMA burst request
0x8E	UART3_TX_DMASREQ	UART3 TX DMA single request

5.6.2 Event Subscribers

There are eleven subscribers for the MCU event fabric. Most of these subscribers are different peripherals that must be configured differently according to the purpose of those specific peripherals. The following eight subscribers are not described in this chapter, but rather in each of the corresponding peripheral chapters:

- Micro Direct Memory Access (µDMA) (see Chapter 16)
- Four General-Purpose Timers (see Chapter 17)
- Sensor Controllers with Digital and Analog Peripherals (AUX) (see Chapter 20)
- Inter-IC Sound (I^2S) (see Chapter 25)
- Radio (see Chapter 26)

The following three subscribers are described as they are related to the CPU and the CPU interrupts:

- System CPU
- Non-maskable Interrupt (NMI) to System CPU
- Freeze

5.6.2.1 System CPU

Table 5-9 shows that the interrupts with vector number from 16 to 49 are sourced by the events routed in the MCU event fabric to the system CPU. The event fabric routes all level interrupt events to the system CPU. The event/interrupt called AON programmable 0 can be configured in the AON event fabric. EVENT:CPUIRQSEL29 is a read-only register for routing within the MCU event fabric and cannot be configured, but the input event within the AON event fabric going to this line can be configured. One dynamic event/interrupt called Dynamic Programmable Event has the valid selections that are shown in Table 5-9. The EVENT:CPUIRQSEL29 register is used to configure the input (see Section 5.8.2).

See the EVENT:CPUIRQSEL30 register in see Section 5.8.2.

5.6.2.2 NMI

The NMI subscriber has one nonconfigurable input that comes from the WDT. The read-only register (CM3NMISEL0) shows the only valid input event.

5.6.2.3 Freeze

In the CC13x4x10 and CC26x4x10 device platform, the freeze subscriber passes the halted debug signal to peripherals such as the General-Purpose Timer, the Sensor Controller with digital and analog peripherals (AUX), the Radio, and the RTC. When the system CPU halts, the connected peripherals that have freeze enabled also halt. The programmable output can be set to static values of 0 or 1, and can also be set to pass the halted signal. The possible events listed in Table 5-8 can be selected in the FRZSEL0 register.

Table 5-8. Freeze Subscriber Event Selection

Event Number	Event Enumeration
0x0	NONE
0x78	CPU_HALTED
0x79	ALWAYS_ACTIVE

Note

When freeze is asserted, RTC stops incrementing the main counter, but the update event from RTC (goes to RF core and AON event fabric) does not stop. The update event is a down division of SCLK_LF and has no dependency on the main counter. So in practice, when you are halting the CPU for debugging, there is no way to stop these update events to RFC.

5.7 AON Events

Table 5-9. AON Events

Event No.	Name	Description
0x0	IOEV_MCU_WU	Edge detect IO event from the DIO
0x1	AUX_TIMER2_EV0	Event 0 from AUX TImer2
0x2	AUX_TIMER2_EV1	Event 1 from AUX TImer2
0x3	AUX_TIMER2_EV2	Event 2 from AUX TImer2
0x4	AUX_TIMER2_EV3	Event 3 from AUX TImer2
0x5	BATMON_BATT_UL	BATMON event: Battery level above upper limit
0x6	BATMON_BATT_LL	BATMON event: Battery level below lower limit
0x7	BATMON_TEMP_UL	BATMON event: Temperature level above upper limit
0x8	BATMON_TEMP_LL	BATMON event: Temperature level below lower limit
0x9	BATMON_COMBINED	Combined event from BATMON
0x20	PAD	Edge detect on any PAD
0x23 to 0x25	RTC_CH0 to RTC_CH2	RTC channel n event, n=0..2
0x26 to 0x28	RTC_CH0_DLY to RTC_CH2_DLY	RTC channel n - delayed event, n=0..2
0x29	RTC_COMB_DLY	RTC combined delayed event
0x2A	RTC_UPD	RTC Update Tick
0x2B	JTAG	JTAG generated event
0x2C to 0x2E	AUX_SWEV0 to AUX_SWEV2	AUX Software triggered event #n, n=0..2
0x2F	AUX_COMPA	Comparator A triggered
0x30	AUX_COMPB	Comparator B triggered
0x31	AUX_ADC_DONE	ADC conversion completed
0x32	AUX_TDC_DONE	TDC completed or timed out
0x33 to 0x34	AUX_TIMER0_EV to AUX_TIMER1_EV	AUX Timer n Event, n=0..1
0x35	BATMON_TEMP	BATMON temperature update event
0x36	BATMON_VOLT	BATMON voltage update event
0x37	AUX_COMPB_ASYNC	Comparator B triggered
0x38	AUX_COMPB_ASYNC_N	Comparator B not triggered
0x3F	NONE	No event

5.8 Interrupts and Events Registers

5.8.1 AON_EVENT Registers

Table 5-10 lists the memory-mapped registers for the AON_EVENT registers. All register offset addresses not listed in Table 5-10 should be considered as reserved locations and the register contents should not be modified.

Table 5-10. AON_EVENT Registers

Offset	Acronym	Register Name	Section
0h	MCUWUSEL	Wake-up Selector For MCU	Section 5.8.1.1
4h	MCUWUSEL1	Wake-up Selector For MCU	Section 5.8.1.2
8h	EVTOMCUSEL	Event Selector For MCU Event Fabric	Section 5.8.1.3
Ch	RTCSEL	RTC Capture Event Selector For AON_RTC	Section 5.8.1.4

Complex bit access types are encoded to fit into small table cells. Table 5-11 shows the codes that are used for access types in this section.

Table 5-11. AON_EVENT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

5.8.1.1 MCUWUSEL Register (Offset = 0h) [Reset = 3F3F3F3Fh]

MCUWUSEL is shown in Table 5-12.

Return to the Summary Table.

Wake-up Selector For MCU
This register contains pointers to 4 of 8 events (events 0 to 3) which are routed to AON_PMCTRL as wakeup sources for MCU. AON_PMCTRL will start a wakeup sequence for the MCU domain when either of the 8 selected events are asserted. A wakeup sequence will guarantee that the MCU power switches are turned on, LDO resources are available and SCLK_HF is available and selected as clock source for MCU.
Note: It is required to setup a wakeup event in AON_EVENT before MCU is requesting powerdown (PRCM requests uLDO, see conditions in PRCM:VDCTL.ULDO).

Table 5-12. MCUWUSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-24	WU3_EV	R/W	3Fh	MCU Wakeup Source #3 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
23-22	RESERVED	R	0h	Reserved

Table 5-12. MCUWUSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-16	WU2_EV	R/W	3Fh	MCU Wakeup Source #2 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
15-14	RESERVED	R	0h	Reserved

Table 5-12. MCUWUSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	WU1_EV	R/W	3Fh	MCU Wakeup Source #1 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
7-6	RESERVED	R	0h	Reserved

Table 5-12. MCUWUSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU0_EV	R/W	3Fh	MCU Wakeup Source #0 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low

5.8.1.2 MCUWUSEL1 Register (Offset = 4h) [Reset = 3F3F3F3Fh]

MCUWUSEL1 is shown in Table 5-13.

Return to the Summary Table.

Wake-up Selector For MCU
This register contains pointers to 4 of 8 events (events 4 to 7) which are routed to AON_PMCTRL as wakeup sources for MCU. AON_PMCTRL will start a wakeup sequence for the MCU domain when either of the 8 selected events are asserted. A wakeup sequence will guarantee that the MCU power switches are turned on, LDO resources are available and SCLK_HF is available and selected as clock source for MCU.
Note: It is required to setup a wakeup event in AON_EVENT before MCU is requesting powerdown (PRCM requests uLDO, see conditions in PRCM:VDCTL.ULDO).

Table 5-13. MCUWUSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-24	WU7_EV	R/W	3Fh	MCU Wakeup Source #7 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
23-22	RESERVED	R	0h	Reserved

Table 5-13. MCUWUSEL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-16	WU6_EV	R/W	3Fh	MCU Wakeup Source #6 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
15-14	RESERVED	R	0h	Reserved

Table 5-13. MCUWUSEL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	WU5_EV	R/W	3Fh	MCU Wakeup Source #5 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
7-6	RESERVED	R	0h	Reserved

Table 5-13. MCUWUSEL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU4_EV	R/W	3Fh	MCU Wakeup Source #4 AON Event Source selecting 1 of 8 events routed to AON_PMCTRL for waking up the MCU domain from Power Off or Power Down. Note: 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_MCU_WU in [MCU_IOC:IOCFGx.IOEV_MCU_WU_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low

5.8.1.3 EVTOMCUSEL Register (Offset = 8h) [Reset = 002B2B2Bh]

EVTOMCUSEL is shown in Table 5-14.

Return to the Summary Table.

Event Selector For MCU Event Fabric
This register contains pointers for 3 AON events that are routed to the MCU Event Fabric EVENT

Table 5-14. EVTOMCUSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21-16	AON_PROG2_EV	R/W	2Bh	Event selector for AON_PROG2 event. AON Event Source id# selecting event routed to EVENT as AON_PROG2 event. 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_AON_PROG2 in [MCU_IOC:IOCFGx.IOEV_AON_PROG2_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = RTC channel 1 - delayed event 28h = RTC channel 2 - delayed event 29h = RTC combined delayed event 2Ah = RTC Update Tick (16 kHz signal, i.e. event line toggles value every 32 kHz clock period) 2Bh = JTAG generated event 2Ch = AUX Software triggered event #0. Triggered by AUX_EVCTL:SWEVSET.SWEV0 2Dh = AUX Software triggered event #1. Triggered by AUX_EVCTL:SWEVSET.SWEV1 2Eh = AUX Software triggered event #2. Triggered by AUX_EVCTL:SWEVSET.SWEV2 2Fh = Comparator A triggered 30h = Comparator B triggered 31h = ADC conversion completed 32h = TDC completed or timed out 33h = AUX Timer 0 Event 34h = AUX Timer 1 Event 35h = BATMON temperature update event 36h = BATMON voltage update event 37h = Comparator B triggered. Asynchronous signal directly from the AUX Comparator B as opposed to AUX_COMPB which is synchronized in AUX 38h = Comparator B not triggered. Asynchronous signal directly from AUX Comparator B (inverted) as opposed to AUX_COMPB which is synchronized in AUX 3Fh = No event, always low
15-14	RESERVED	R	0h	Reserved

Table 5-14. EVTOMCUSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	AON_PROG1_EV	R/W	2Bh	Event selector for AON_PROG1 event. AON Event Source id# selecting event routed to EVENT as AON_PROG1 event. 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_AON_PROG1 in [MCU_IOC:IOCFGx.IOEV_AON_PROG1_EN] 1h = Event 0 from AUX TImer2 2h = Event 1 from AUX TImer2 3h = Event 2 from AUX TImer2 4h = Event 3 from AUX TImer2 5h = BATMON event: Battery level above upper limit 6h = BATMON event: Battery level below lower limit 7h = BATMON event: Temperature level above upper limit 8h = BATMON event: Temperature level below lower limit 9h = Combined event from BATMON 20h = Edge detect on any PAD 23h = RTC channel 0 event 24h = RTC channel 1 event 25h = RTC channel 2 event 26h = RTC channel 0 - delayed event 27h = 0 28h = 0 29h = 0 2Ah = 0 2Bh = 0 2Ch = 0 2Dh = 0 2Eh = 0 2Fh = 0 30h = 0 31h = 0 32h = 0 33h = 0 34h = 0 35h = 0 36h = 0 37h = 0 38h = 0 3Fh = 0
7-6	RESERVED	R	0h	Reserved

Table 5-14. EVTOMCUSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	AON_PROG0_EV	R/W	2Bh	Event selector for AON_PROG0 event. AON Event Source id# selecting event routed to EVENT as AON_PROG0 event. 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_AON_PROG0 in [MCU_IOC:IOCFGx.IOEV_AON_PROG0_EN] 1h = 0 2h = 0 3h = 0 4h = 0 5h = 0 6h = 0 7h = 0 8h = 0 9h = 0 20h = 0 23h = 0 24h = 0 25h = 0 26h = 0 27h = 0 28h = 0 29h = 0 2Ah = 0 2Bh = 0 2Ch = 0 2Dh = 0 2Eh = 0 2Fh = 0 30h = 0 31h = 0 32h = 0 33h = 0 34h = 0 35h = 0 36h = 0 37h = 0 38h = 0 3Fh = 0

5.8.1.4 RTCSEL Register (Offset = Ch) [Reset = 0000003Fh]

RTCSEL is shown in Table 5-15.

Return to the Summary Table.

RTC Capture Event Selector For AON_RTC
This register contains a pointer to select an AON event for RTC capture. Please refer to AON_RTC:CH1CAPT

Table 5-15. RTCSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	RTC_CH1_CAPT_EV	R/W	3Fh	AON Event Source id# for RTCSEL event which is fed to AON_RTC. Please refer to AON_RTC:CH1CAPT 0h = Edge detect IO event from the DIO(s) which have enabled contribution to IOEV_RTC in [MCU_IOC:IOCFGx.IOEV_RTC_EN] 1h = 0 2h = 0 3h = 0 4h = 0 5h = 0 6h = 0 7h = 0 8h = 0 9h = 0 20h = 0 23h = 0 24h = 0 25h = 0 26h = 0 27h = 0 28h = 0 29h = 0 2Ah = 0 2Bh = 0 2Ch = 0 2Dh = 0 2Eh = 0 2Fh = 0 30h = 0 31h = 0 32h = 0 33h = 0 34h = 0 35h = 0 36h = 0 37h = 0 38h = 0 3Fh = 0

5.8.2 EVENT Registers

Table 5-16 lists the memory-mapped registers for the EVENT registers. All register offset addresses not listed in Table 5-16 should be considered as reserved locations and the register contents should not be modified.

Table 5-16. EVENT Registers

Offset	Acronym	Register Name	Section
0h	CPUIRQSEL0	Output Selection for CPU Interrupt 0	Section 5.8.2.1
4h	CPUIRQSEL1	Output Selection for CPU Interrupt 1	Section 5.8.2.2
8h	CPUIRQSEL2	Output Selection for CPU Interrupt 2	Section 5.8.2.3
Ch	CPUIRQSEL3	Output Selection for CPU Interrupt 3	Section 5.8.2.4
10h	CPUIRQSEL4	Output Selection for CPU Interrupt 4	Section 5.8.2.5
14h	CPUIRQSEL5	Output Selection for CPU Interrupt 5	Section 5.8.2.6
18h	CPUIRQSEL6	Output Selection for CPU Interrupt 6	Section 5.8.2.7
1Ch	CPUIRQSEL7	Output Selection for CPU Interrupt 7	Section 5.8.2.8
20h	CPUIRQSEL8	Output Selection for CPU Interrupt 8	Section 5.8.2.9
24h	CPUIRQSEL9	Output Selection for CPU Interrupt 9	Section 5.8.2.10
28h	CPUIRQSEL10	Output Selection for CPU Interrupt 10	Section 5.8.2.11
2Ch	CPUIRQSEL11	Output Selection for CPU Interrupt 11	Section 5.8.2.12
30h	CPUIRQSEL12	Output Selection for CPU Interrupt 12	Section 5.8.2.13
34h	CPUIRQSEL13	Output Selection for CPU Interrupt 13	Section 5.8.2.14
38h	CPUIRQSEL14	Output Selection for CPU Interrupt 14	Section 5.8.2.15
3Ch	CPUIRQSEL15	Output Selection for CPU Interrupt 15	Section 5.8.2.16
40h	CPUIRQSEL16	Output Selection for CPU Interrupt 16	Section 5.8.2.17
44h	CPUIRQSEL17	Output Selection for CPU Interrupt 17	Section 5.8.2.18
48h	CPUIRQSEL18	Output Selection for CPU Interrupt 18	Section 5.8.2.19
4Ch	CPUIRQSEL19	Output Selection for CPU Interrupt 19	Section 5.8.2.20
50h	CPUIRQSEL20	Output Selection for CPU Interrupt 20	Section 5.8.2.21
54h	CPUIRQSEL21	Output Selection for CPU Interrupt 21	Section 5.8.2.22
58h	CPUIRQSEL22	Output Selection for CPU Interrupt 22	Section 5.8.2.23
5Ch	CPUIRQSEL23	Output Selection for CPU Interrupt 23	Section 5.8.2.24
60h	CPUIRQSEL24	Output Selection for CPU Interrupt 24	Section 5.8.2.25
64h	CPUIRQSEL25	Output Selection for CPU Interrupt 25	Section 5.8.2.26
68h	CPUIRQSEL26	Output Selection for CPU Interrupt 26	Section 5.8.2.27
6Ch	CPUIRQSEL27	Output Selection for CPU Interrupt 27	Section 5.8.2.28
70h	CPUIRQSEL28	Output Selection for CPU Interrupt 28	Section 5.8.2.29
74h	CPUIRQSEL29	Output Selection for CPU Interrupt 29	Section 5.8.2.30
78h	CPUIRQSEL30	Output Selection for CPU Interrupt 30	Section 5.8.2.31
7Ch	CPUIRQSEL31	Output Selection for CPU Interrupt 31	Section 5.8.2.32
80h	CPUIRQSEL32	Output Selection for CPU Interrupt 32	Section 5.8.2.33
84h	CPUIRQSEL33	Output Selection for CPU Interrupt 33	Section 5.8.2.34
88h	CPUIRQSEL34	Output Selection for CPU Interrupt 34	Section 5.8.2.35
8Ch	CPUIRQSEL35	Output Selection for CPU Interrupt 35	Section 5.8.2.36
90h	CPUIRQSEL36	Output Selection for CPU Interrupt 36	Section 5.8.2.37
94h	CPUIRQSEL37	Output Selection for CPU Interrupt 37	Section 5.8.2.38
98h	CPUIRQSEL38	Output Selection for CPU Interrupt 38	Section 5.8.2.39
9Ch	CPUIRQSEL39	Output Selection for CPU Interrupt 39	Section 5.8.2.40
A0h	CPUIRQSEL40	Output Selection for CPU Interrupt 40	Section 5.8.2.41
A4h	CPUIRQSEL41	Output Selection for CPU Interrupt 41	Section 5.8.2.42

Table 5-16. EVENT Registers (continued)

Offset	Acronym	Register Name	Section
A8h	CPUIRQSEL42	Output Selection for CPU Interrupt 42	Section 5.8.2.43
100h	RFCSEL0	Output Selection for RFC Event 0	Section 5.8.2.44
104h	RFCSEL1	Output Selection for RFC Event 1	Section 5.8.2.45
108h	RFCSEL2	Output Selection for RFC Event 2	Section 5.8.2.46
10Ch	RFCSEL3	Output Selection for RFC Event 3	Section 5.8.2.47
110h	RFCSEL4	Output Selection for RFC Event 4	Section 5.8.2.48
114h	RFCSEL5	Output Selection for RFC Event 5	Section 5.8.2.49
118h	RFCSEL6	Output Selection for RFC Event 6	Section 5.8.2.50
11Ch	RFCSEL7	Output Selection for RFC Event 7	Section 5.8.2.51
120h	RFCSEL8	Output Selection for RFC Event 8	Section 5.8.2.52
124h	RFCSEL9	Output Selection for RFC Event 9	Section 5.8.2.53
200h	GPT0ACAPTSEL	Output Selection for GPT0 0	Section 5.8.2.54
204h	GPT0BCAPTSEL	Output Selection for GPT0 1	Section 5.8.2.55
300h	GPT1ACAPTSEL	Output Selection for GPT1 0	Section 5.8.2.56
304h	GPT1BCAPTSEL	Output Selection for GPT1 1	Section 5.8.2.57
400h	GPT2ACAPTSEL	Output Selection for GPT2 0	Section 5.8.2.58
404h	GPT2BCAPTSEL	Output Selection for GPT2 1	Section 5.8.2.59
508h	UDMACH1SSEL	Output Selection for DMA Channel 1 SREQ	Section 5.8.2.60
50Ch	UDMACH1BSEL	Output Selection for DMA Channel 1 REQ	Section 5.8.2.61
510h	UDMACH2SSEL	Output Selection for DMA Channel 2 SREQ	Section 5.8.2.62
514h	UDMACH2BSEL	Output Selection for DMA Channel 2 REQ	Section 5.8.2.63
518h	UDMACH3SSEL	Output Selection for DMA Channel 3 SREQ	Section 5.8.2.64
51Ch	UDMACH3BSEL	Output Selection for DMA Channel 3 REQ	Section 5.8.2.65
520h	UDMACH4SSEL	Output Selection for DMA Channel 4 SREQ	Section 5.8.2.66
524h	UDMACH4BSEL	Output Selection for DMA Channel 4 REQ	Section 5.8.2.67
528h	UDMACH5SSEL	Output Selection for DMA Channel 5 SREQ	Section 5.8.2.68
52Ch	UDMACH5BSEL	Output Selection for DMA Channel 5 REQ	Section 5.8.2.69
530h	UDMACH6SSEL	Output Selection for DMA Channel 6 SREQ	Section 5.8.2.70
534h	UDMACH6BSEL	Output Selection for DMA Channel 6 REQ	Section 5.8.2.71
538h	UDMACH7SSEL	Output Selection for DMA Channel 7 SREQ	Section 5.8.2.72
53Ch	UDMACH7BSEL	Output Selection for DMA Channel 7 REQ	Section 5.8.2.73
540h	UDMACH8SSEL	Output Selection for DMA Channel 8 SREQ	Section 5.8.2.74
544h	UDMACH8BSEL	Output Selection for DMA Channel 8 REQ	Section 5.8.2.75
548h	UDMACH9SSEL	Output Selection for DMA Channel 9 SREQ	Section 5.8.2.76
54Ch	UDMACH9BSEL	Output Selection for DMA Channel 9 REQ	Section 5.8.2.77
550h	UDMACH10SSEL	Output Selection for DMA Channel 10 SREQ	Section 5.8.2.78
554h	UDMACH10BSEL	Output Selection for DMA Channel 10 REQ	Section 5.8.2.79
558h	UDMACH11SSEL	Output Selection for DMA Channel 11 SREQ	Section 5.8.2.80
55Ch	UDMACH11BSEL	Output Selection for DMA Channel 11 REQ	Section 5.8.2.81
560h	UDMACH12SSEL	Output Selection for DMA Channel 12 SREQ	Section 5.8.2.82
564h	UDMACH12BSEL	Output Selection for DMA Channel 12 REQ	Section 5.8.2.83
56Ch	UDMACH13BSEL	Output Selection for DMA Channel 13 REQ	Section 5.8.2.84
574h	UDMACH14BSEL	Output Selection for DMA Channel 14 REQ	Section 5.8.2.85
57Ch	UDMACH15BSEL	Output Selection for DMA Channel 15 REQ	Section 5.8.2.86
580h	UDMACH16SSEL	Output Selection for DMA Channel 16 SREQ	Section 5.8.2.87

Table 5-16. EVENT Registers (continued)

Offset	Acronym	Register Name	Section
584h	UDMACH16BSEL	Output Selection for DMA Channel 16 REQ	Section 5.8.2.88
588h	UDMACH17SSEL	Output Selection for DMA Channel 17 SREQ	Section 5.8.2.89
58Ch	UDMACH17BSEL	Output Selection for DMA Channel 17 REQ	Section 5.8.2.90
598h	UDMACH19SSEL	Output Selection for DMA Channel 19 SREQ	Section 5.8.2.91
59Ch	UDMACH19BSEL	Output Selection for DMA Channel 19 REQ	Section 5.8.2.92
5A0h	UDMACH20SSEL	Output Selection for DMA Channel 20 SREQ	Section 5.8.2.93
5A4h	UDMACH20BSEL	Output Selection for DMA Channel 20 REQ	Section 5.8.2.94
5A8h	UDMACH21SSEL	Output Selection for DMA Channel 21 SREQ	Section 5.8.2.95
5ACh	UDMACH21BSEL	Output Selection for DMA Channel 21 REQ	Section 5.8.2.96
5B0h	UDMACH22SSEL	Output Selection for DMA Channel 22 SREQ	Section 5.8.2.97
5B4h	UDMACH22BSEL	Output Selection for DMA Channel 22 REQ	Section 5.8.2.98
5B8h	UDMACH23SSEL	Output Selection for DMA Channel 23 SREQ	Section 5.8.2.99
5BCh	UDMACH23BSEL	Output Selection for DMA Channel 23 REQ	Section 5.8.2.100
5C0h	UDMACH24SSEL	Output Selection for DMA Channel 24 SREQ	Section 5.8.2.101
5C4h	UDMACH24BSEL	Output Selection for DMA Channel 24 REQ	Section 5.8.2.102
5C8h	UDMACH25SSEL	Output Selection for DMA Channel 25 SREQ	Section 5.8.2.103
5CCh	UDMACH25BSEL	Output Selection for DMA Channel 25 REQ	Section 5.8.2.104
5D0h	UDMACH26SSEL	Output Selection for DMA Channel 26 SREQ	Section 5.8.2.105
5D4h	UDMACH26BSEL	Output Selection for DMA Channel 26 REQ	Section 5.8.2.106
5E0h	UDMACH28SSEL	Output Selection for DMA Channel 28 SREQ	Section 5.8.2.107
5E4h	UDMACH28BSEL	Output Selection for DMA Channel 28 REQ	Section 5.8.2.108
5E8h	UDMACH29SSEL	Output Selection for DMA Channel 29 SREQ	Section 5.8.2.109
5ECh	UDMACH29BSEL	Output Selection for DMA Channel 29 REQ	Section 5.8.2.110
5F0h	UDMACH30SSEL	Output Selection for DMA Channel 30 SREQ	Section 5.8.2.111
5F4h	UDMACH30BSEL	Output Selection for DMA Channel 30 REQ	Section 5.8.2.112
5F8h	UDMACH31SSEL	Output Selection for DMA Channel 31 SREQ	Section 5.8.2.113
5FCh	UDMACH31BSEL	Output Selection for DMA Channel 31 REQ	Section 5.8.2.114
600h	GPT3ACAPTSEL	Output Selection for GPT3 0	Section 5.8.2.115
604h	GPT3BCAPTSEL	Output Selection for GPT3 1	Section 5.8.2.116
700h	AUXSEL0	Output Selection for AUX Subscriber 0	Section 5.8.2.117
800h	CM3NMISEL0	Output Selection for NMI Subscriber 0	Section 5.8.2.118
900h	I2SSTMPSEL0	Output Selection for I2S Subscriber 0	Section 5.8.2.119
A00h	FRZSEL0	Output Selection for FRZ Subscriber	Section 5.8.2.120
F00h	SWEV	Set or Clear Software Events	Section 5.8.2.121

Complex bit access types are encoded to fit into small table cells. Table 5-17 shows the codes that are used for access types in this section.

Table 5-17. EVENT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

5.8.2.1 CPUIRQSEL0 Register (Offset = 0h) [Reset = 00000004h]

CPUIRQSEL0 is shown in Table 5-18.

Return to the Summary Table.

Output Selection for CPU Interrupt 0

Table 5-18. CPUIRQSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	4h	Read only selection value 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings

5.8.2.2 CPUIRQSEL1 Register (Offset = 4h) [Reset = 00000009h]

CPUIRQSEL1 is shown in Table 5-19.

Return to the Summary Table.

Output Selection for CPU Interrupt 1

Table 5-19. CPUIRQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	9h	Read only selection value 9h = Interrupt event from I2C

5.8.2.3 CPUIRQSEL2 Register (Offset = 8h) [Reset = 0000001Eh]

CPUIRQSEL2 is shown in Table 5-20.

Return to the Summary Table.

Output Selection for CPU Interrupt 2

Table 5-20. CPUIRQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Eh	Read only selection value 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event

5.8.2.4 CPUIRQSEL3 Register (Offset = Ch) [Reset = 0000001Fh]

CPUIRQSEL3 is shown in Table 5-21.

Return to the Summary Table.

Output Selection for CPU Interrupt 3

Table 5-21. CPUIRQSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Fh	Read only selection value 1Fh = PKA Interrupt event

5.8.2.5 CPUIRQSEL4 Register (Offset = 10h) [Reset = 00000007h]

CPUIRQSEL4 is shown in Table 5-22.

Return to the Summary Table.

Output Selection for CPU Interrupt 4

Table 5-22. CPUIRQSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7h	Read only selection value 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting

5.8.2.6 CPUIRQSEL5 Register (Offset = 14h) [Reset = 00000024h]

CPUIRQSEL5 is shown in Table 5-23.

Return to the Summary Table.

Output Selection for CPU Interrupt 5

Table 5-23. CPUIRQSEL5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	24h	Read only selection value 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS

5.8.2.7 CPUIRQSEL6 Register (Offset = 18h) [Reset = 0000001Ch]

CPUIRQSEL6 is shown in Table 5-24.

Return to the Summary Table.

Output Selection for CPU Interrupt 6

Table 5-24. CPUIRQSEL6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Ch	Read only selection value 1Ch = AUX software event 0, triggered by AUX_EVCTL:SWEVSET.SWEV0, also available as AUX_EVENT0 AON wake up event. MCU domain wakeup control AON_EVENT:MCUWUSEL

5.8.2.8 CPUIRQSEL7 Register (Offset = 1Ch) [Reset = 00000022h]

CPUIRQSEL7 is shown in Table 5-25.

Return to the Summary Table.

Output Selection for CPU Interrupt 7

Table 5-25. CPUIRQSEL7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	22h	Read only selection value 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS

5.8.2.9 CPUIRQSEL8 Register (Offset = 20h) [Reset = 00000023h]

CPUIRQSEL8 is shown in Table 5-26.

Return to the Summary Table.

Output Selection for CPU Interrupt 8

Table 5-26. CPUIRQSEL8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	23h	Read only selection value 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS

5.8.2.10 CPUIRQSEL9 Register (Offset = 24h) [Reset = 0000001Bh]

CPUIRQSEL9 is shown in Table 5-27.

Return to the Summary Table.

Output Selection for CPU Interrupt 9

Table 5-27. CPUIRQSEL9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Bh	Read only selection value 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event

5.8.2.11 CPUIRQSEL10 Register (Offset = 28h) [Reset = 0000001Ah]

CPUIRQSEL10 is shown in Table 5-28.

Return to the Summary Table.

Output Selection for CPU Interrupt 10

Table 5-28. CPUIRQSEL10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Ah	Read only selection value 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG

5.8.2.12 CPUIRQSEL11 Register (Offset = 2Ch) [Reset = 00000019h]

CPUIRQSEL11 is shown in Table 5-29.

Return to the Summary Table.

Output Selection for CPU Interrupt 11

Table 5-29. CPUIRQSEL11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	19h	Read only selection value 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG

5.8.2.13 CPUIRQSEL12 Register (Offset = 30h) [Reset = 00000008h]

CPUIRQSEL12 is shown in Table 5-30.

Return to the Summary Table.

Output Selection for CPU Interrupt 12

Table 5-30. CPUIRQSEL12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8h	Read only selection value 8h = Interrupt event from I2S

5.8.2.14 CPUIRQSEL13 Register (Offset = 34h) [Reset = 0000001Dh]

CPUIRQSEL13 is shown in Table 5-31.

Return to the Summary Table.

Output Selection for CPU Interrupt 13

Table 5-31. CPUIRQSEL13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1Dh	Read only selection value 1Dh = AUX software event 1, triggered by AUX_EVCTL:SWEVSET.SWEV1, also available as AUX_EVENT2 AON wake up event. MCU domain wakeup control AON_EVENT:MCUWUSEL

5.8.2.15 CPUIRQSEL14 Register (Offset = 38h) [Reset = 00000018h]

CPUIRQSEL14 is shown in Table 5-32.

Return to the Summary Table.

Output Selection for CPU Interrupt 14

Table 5-32. CPUIRQSEL14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	18h	Read only selection value 18h = Watchdog interrupt event, controlled by WDT:CTL.INTEN

5.8.2.16 CPUIRQSEL15 Register (Offset = 3Ch) [Reset = 00000010h]

CPUIRQSEL15 is shown in Table 5-33.

Return to the Summary Table.

Output Selection for CPU Interrupt 15

Table 5-33. CPUIRQSEL15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	10h	Read only selection value 10h = GPT0A interrupt event, controlled by GPT0:TAMR

5.8.2.17 CPUIRQSEL16 Register (Offset = 40h) [Reset = 00000011h]

CPUIRQSEL16 is shown in Table 5-34.

Return to the Summary Table.

Output Selection for CPU Interrupt 16

Table 5-34. CPUIRQSEL16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	11h	Read only selection value 11h = GPT0B interrupt event, controlled by GPT0:TBMR

5.8.2.18 CPUIRQSEL17 Register (Offset = 44h) [Reset = 00000012h]

CPUIRQSEL17 is shown in Table 5-35.

Return to the Summary Table.

Output Selection for CPU Interrupt 17

Table 5-35. CPUIRQSEL17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	12h	Read only selection value 12h = GPT1A interrupt event, controlled by GPT1:TAMR

5.8.2.19 CPUIRQSEL18 Register (Offset = 48h) [Reset = 00000013h]

CPUIRQSEL18 is shown in Table 5-36.

Return to the Summary Table.

Output Selection for CPU Interrupt 18

Table 5-36. CPUIRQSEL18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	13h	Read only selection value 13h = GPT1B interrupt event, controlled by GPT1:TBMR

5.8.2.20 CPUIRQSEL19 Register (Offset = 4Ch) [Reset = 0000000Ch]

CPUIRQSEL19 is shown in Table 5-37.

Return to the Summary Table.

Output Selection for CPU Interrupt 19

Table 5-37. CPUIRQSEL19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	Ch	Read only selection value Ch = GPT2A interrupt event, controlled by GPT2:TAMR

5.8.2.21 CPUIRQSEL20 Register (Offset = 50h) [Reset = 0000000Dh]

CPUIRQSEL20 is shown in Table 5-38.

Return to the Summary Table.

Output Selection for CPU Interrupt 20

Table 5-38. CPUIRQSEL20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	Dh	Read only selection value Dh = GPT2B interrupt event, controlled by GPT2:TBMR

5.8.2.22 CPUIRQSEL21 Register (Offset = 54h) [Reset = 0000000Eh]

CPUIRQSEL21 is shown in Table 5-39.

Return to the Summary Table.

Output Selection for CPU Interrupt 21

Table 5-39. CPUIRQSEL21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	Eh	Read only selection value Eh = GPT3A interrupt event, controlled by GPT3:TAMR

5.8.2.23 CPUIRQSEL22 Register (Offset = 58h) [Reset = 0000000Fh]

CPUIRQSEL22 is shown in Table 5-40.

Return to the Summary Table.

Output Selection for CPU Interrupt 22

Table 5-40. CPUIRQSEL22 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	Fh	Read only selection value Fh = GPT3B interrupt event, controlled by GPT3:TBMR

5.8.2.24 CPUIRQSEL23 Register (Offset = 5Ch) [Reset = 0000005Dh]

CPUIRQSEL23 is shown in Table 5-41.

Return to the Summary Table.

Output Selection for CPU Interrupt 23

Table 5-41. CPUIRQSEL23 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	5Dh	Read only selection value 5Dh = CRYPTO result available interupt event, the corresponding flag is found here CRYPTO:IRQSTAT.RESULT_AVAIL. Controlled by CRYPTO:IRQSTAT.RESULT_AVAIL

5.8.2.25 CPUIRQSEL24 Register (Offset = 60h) [Reset = 00000027h]

CPUIRQSEL24 is shown in Table 5-42.

Return to the Summary Table.

Output Selection for CPU Interrupt 24

Table 5-42. CPUIRQSEL24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	27h	Read only selection value 27h = Combined DMA done, corresponding flags are here UDMA0:REQDONE

5.8.2.26 CPUIRQSEL25 Register (Offset = 64h) [Reset = 00000026h]

CPUIRQSEL25 is shown in Table 5-43.

Return to the Summary Table.

Output Selection for CPU Interrupt 25

Table 5-43. CPUIRQSEL25 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	26h	Read only selection value 26h = DMA bus error, corresponds to UDMA0:ERROR.STATUS

5.8.2.27 CPUIRQSEL26 Register (Offset = 68h) [Reset = 00000015h]

CPUIRQSEL26 is shown in Table 5-44.

Return to the Summary Table.

Output Selection for CPU Interrupt 26

Table 5-44. CPUIRQSEL26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	15h	Read only selection value 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT

5.8.2.28 CPUIRQSEL27 Register (Offset = 6Ch) [Reset = 00000064h]

CPUIRQSEL27 is shown in Table 5-45.

Return to the Summary Table.

Output Selection for CPU Interrupt 27

Table 5-45. CPUIRQSEL27 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	64h	Read only selection value 64h = Software event 0, triggered by SWEV.SWEV0

5.8.2.29 CPUIRQSEL28 Register (Offset = 70h) [Reset = 0000000Bh]

CPUIRQSEL28 is shown in Table 5-46.

Return to the Summary Table.

Output Selection for CPU Interrupt 28

Table 5-46. CPUIRQSEL28 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	Bh	Read only selection value Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS

5.8.2.30 CPUIRQSEL29 Register (Offset = 74h) [Reset = 00000001h]

CPUIRQSEL29 is shown in Table 5-47.

Return to the Summary Table.

Output Selection for CPU Interrupt 29

Table 5-47. CPUIRQSEL29 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	1h	Read only selection value 1h = AON programmable event 0. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG0_EV

5.8.2.31 CPUIRQSEL30 Register (Offset = 78h) [Reset = 00000000h]

CPUIRQSEL30 is shown in Table 5-48.

Return to the Summary Table.

Output Selection for CPU Interrupt 30

Table 5-48. CPUIRQSEL30 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	0h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 2h = AON programmable event 1. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG1_EV 3h = AON programmable event 2. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG2_EV 8h = Interrupt event from I2S Ah = AUX Software event 0, AUX_EVCTL:SWEVSET.SWEV0 14h = DMA done for software tiggered UDMA channel 0, see UDMA0:SOFTREQ 16h = DMA done for software tiggered UDMA channel 18, see UDMA0:SOFTREQ 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 5Eh = CRYPTO DMA input done event, the correspondingg flag is CRYPTO:IRQSTAT.DMA_IN_DONE. Controlled by CRYPTO:IRQEN.DMA_IN_DONE 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB 6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted

5.8.2.32 CPUIRQSEL31 Register (Offset = 7Ch) [Reset = 0000006Ah]

CPUIRQSEL31 is shown in Table 5-49.

Return to the Summary Table.

Output Selection for CPU Interrupt 31

Table 5-49. CPUIRQSEL31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	6Ah	Read only selection value 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA

5.8.2.33 CPUIRQSEL32 Register (Offset = 80h) [Reset = 00000073h]

CPUIRQSEL32 is shown in Table 5-50.

Return to the Summary Table.

Output Selection for CPU Interrupt 32

Table 5-50. CPUIRQSEL32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	73h	Read only selection value 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS

5.8.2.34 CPUIRQSEL33 Register (Offset = 84h) [Reset = 00000068h]

CPUIRQSEL33 is shown in Table 5-51.

Return to the Summary Table.

Output Selection for CPU Interrupt 33

Table 5-51. CPUIRQSEL33 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	68h	Read only selection value 68h = TRNG Interrupt event, controlled by TRNG:IRQEN.EN

5.8.2.35 CPUIRQSEL34 Register (Offset = 88h) [Reset = 00000006h]

CPUIRQSEL34 is shown in Table 5-52.

Return to the Summary Table.

Output Selection for CPU Interrupt 34

Table 5-52. CPUIRQSEL34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	6h	Read only selection value 6h = Combined event from Oscillator control

5.8.2.36 CPUIRQSEL35 Register (Offset = 8Ch) [Reset = 00000038h]

CPUIRQSEL35 is shown in Table 5-53.

Return to the Summary Table.

Output Selection for CPU Interrupt 35

Table 5-53. CPUIRQSEL35 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	38h	Read only selection value 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0

5.8.2.37 CPUIRQSEL36 Register (Offset = 90h) [Reset = 00000025h]

CPUIRQSEL36 is shown in Table 5-54.

Return to the Summary Table.

Output Selection for CPU Interrupt 36

Table 5-54. CPUIRQSEL36 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	25h	Read only selection value 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS

5.8.2.38 CPUIRQSEL37 Register (Offset = 94h) [Reset = 00000005h]

CPUIRQSEL37 is shown in Table 5-55.

Return to the Summary Table.

Output Selection for CPU Interrupt 37

Table 5-55. CPUIRQSEL37 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	5h	Read only selection value 5h = Combined event from battery monitor

5.8.2.39 CPUIRQSEL38 Register (Offset = 98h) [Reset = 0000007Ah]

CPUIRQSEL38 is shown in Table 5-56.

Return to the Summary Table.

Output Selection for CPU Interrupt 38

Table 5-56. CPUIRQSEL38 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Ah	Read only selection value 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS

5.8.2.40 CPUIRQSEL39 Register (Offset = 9Ch) [Reset = 0000007Bh]

CPUIRQSEL39 is shown in Table 5-57.

Return to the Summary Table.

Output Selection for CPU Interrupt 39

Table 5-57. CPUIRQSEL39 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Bh	Read only selection value 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS

5.8.2.41 CPUIRQSEL40 Register (Offset = A0h) [Reset = 0000007Ch]

CPUIRQSEL40 is shown in Table 5-58.

Return to the Summary Table.

Output Selection for CPU Interrupt 40

Table 5-58. CPUIRQSEL40 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Ch	Read only selection value 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS

5.8.2.42 CPUIRQSEL41 Register (Offset = A4h) [Reset = 0000007Dh]

CPUIRQSEL41 is shown in Table 5-59.

Return to the Summary Table.

Output Selection for CPU Interrupt 41

Table 5-59. CPUIRQSEL41 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Dh	Read only selection value 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS

5.8.2.43 CPUIRQSEL42 Register (Offset = A8h) [Reset = 0000007Eh]

CPUIRQSEL42 is shown in Table 5-60.

Return to the Summary Table.

Output Selection for CPU Interrupt 42

Table 5-60. CPUIRQSEL42 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Eh	Read only selection value 7Eh = Interrupt event from I2C1

5.8.2.44 RFCSEL0 Register (Offset = 100h) [Reset = 0000003Dh]

RFCSEL0 is shown in Table 5-61.

Return to the Summary Table.

Output Selection for RFC Event 0

Table 5-61. RFCSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	3Dh	Read only selection value 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT

5.8.2.45 RFCSEL1 Register (Offset = 104h) [Reset = 0000003Eh]

RFCSEL1 is shown in Table 5-62.

Return to the Summary Table.

Output Selection for RFC Event 1

Table 5-62. RFCSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	3Eh	Read only selection value 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT

5.8.2.46 RFCSEL2 Register (Offset = 108h) [Reset = 0000003Fh]

RFCSEL2 is shown in Table 5-63.

Return to the Summary Table.

Output Selection for RFC Event 2

Table 5-63. RFCSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	3Fh	Read only selection value 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT

5.8.2.47 RFCSEL3 Register (Offset = 10Ch) [Reset = 00000040h]

RFCSEL3 is shown in Table 5-64.

Return to the Summary Table.

Output Selection for RFC Event 3

Table 5-64. RFCSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	40h	Read only selection value 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT

5.8.2.48 RFCSEL4 Register (Offset = 110h) [Reset = 00000041h]

RFCSEL4 is shown in Table 5-65.

Return to the Summary Table.

Output Selection for RFC Event 4

Table 5-65. RFCSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	41h	Read only selection value 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT

5.8.2.49 RFCSEL5 Register (Offset = 114h) [Reset = 00000042h]

RFCSEL5 is shown in Table 5-66.

Return to the Summary Table.

Output Selection for RFC Event 5

Table 5-66. RFCSEL5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	42h	Read only selection value 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT

5.8.2.50 RFCSEL6 Register (Offset = 118h) [Reset = 00000043h]

RFCSEL6 is shown in Table 5-67.

Return to the Summary Table.

Output Selection for RFC Event 6

Table 5-67. RFCSEL6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	43h	Read only selection value 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT

5.8.2.51 RFCSEL7 Register (Offset = 11Ch) [Reset = 00000044h]

RFCSEL7 is shown in Table 5-68.

Return to the Summary Table.

Output Selection for RFC Event 7

Table 5-68. RFCSEL7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	44h	Read only selection value 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT

5.8.2.52 RFCSEL8 Register (Offset = 120h) [Reset = 00000077h]

RFCSEL8 is shown in Table 5-69.

Return to the Summary Table.

Output Selection for RFC Event 8

Table 5-69. RFCSEL8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	77h	Read only selection value 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN

5.8.2.53 RFCSEL9 Register (Offset = 124h) [Reset = 00000002h]

RFCSEL9 is shown in Table 5-70.

Return to the Summary Table.

Output Selection for RFC Event 9

Table 5-70. RFCSEL9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-70. RFCSEL9 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	2h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 1h = AON programmable event 0. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG0_EV 2h = AON programmable event 1. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG1_EV 8h = Interrupt event from I2S Ah = AUX Software event 0, AUX_EVCTL:SWEVSET.SWEV0 18h = Watchdog interrupt event, controlled by WDT:CTL.INTEN 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 27h = Combined DMA done, corresponding flags are here UDMA0:REQDONE 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 5Dh = CRYPTO result available interupt event, the corresponding flag is found here CRYPTO:IRQSTAT.RESULT_AVAIL. Controlled by CRYPTO:IRQSTAT.RESULT_AVAIL 64h = Software event 0, triggered by SWEV.SWEV0 65h = Software event 1, triggered by SWEV.SWEV1 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB 6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS

Table 5-70. RFCSEL9 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS

5.8.2.54 GPT0ACAPTSEL Register (Offset = 200h) [Reset = 00000055h]

GPT0ACAPTSEL is shown in Table 5-71.

Return to the Summary Table.

Output Selection for GPT0 0

Table 5-71. GPT0ACAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-71. GPT0ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	55h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 55h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT0 wil be routed here. 56h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT1 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-71. GPT0ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.55 GPT0BCAPTSEL Register (Offset = 204h) [Reset = 00000056h]

GPT0BCAPTSEL is shown in Table 5-72.

Return to the Summary Table.

Output Selection for GPT0 1

Table 5-72. GPT0BCAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-72. GPT0BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	56h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 55h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT0 wil be routed here. 56h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT1 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-72. GPT0BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.56 GPT1ACAPTSEL Register (Offset = 300h) [Reset = 00000057h]

GPT1ACAPTSEL is shown in Table 5-73.

Return to the Summary Table.

Output Selection for GPT1 0

Table 5-73. GPT1ACAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-73. GPT1ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	57h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 57h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT2 wil be routed here. 58h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT3 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-73. GPT1ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.57 GPT1BCAPTSEL Register (Offset = 304h) [Reset = 00000058h]

GPT1BCAPTSEL is shown in Table 5-74.

Return to the Summary Table.

Output Selection for GPT1 1

Table 5-74. GPT1BCAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-74. GPT1BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	58h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 57h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT2 wil be routed here. 58h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT3 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-74. GPT1BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.58 GPT2ACAPTSEL Register (Offset = 400h) [Reset = 00000059h]

GPT2ACAPTSEL is shown in Table 5-75.

Return to the Summary Table.

Output Selection for GPT2 0

Table 5-75. GPT2ACAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-75. GPT2ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	59h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 59h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 5Ah = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-75. GPT2ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.59 GPT2BCAPTSEL Register (Offset = 404h) [Reset = 0000005Ah]

GPT2BCAPTSEL is shown in Table 5-76.

Return to the Summary Table.

Output Selection for GPT2 1

Table 5-76. GPT2BCAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-76. GPT2BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	5Ah	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 59h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 5Ah = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-76. GPT2BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.60 UDMACH1SSEL Register (Offset = 508h) [Reset = 00000031h]

UDMACH1SSEL is shown in Table 5-77.

Return to the Summary Table.

Output Selection for DMA Channel 1 SREQ

Table 5-77. UDMACH1SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	31h	Read only selection value 31h = UART0 RX DMA single request, controlled by UART0:DMACTL.RXDMAE

5.8.2.61 UDMACH1BSEL Register (Offset = 50Ch) [Reset = 00000030h]

UDMACH1BSEL is shown in Table 5-78.

Return to the Summary Table.

Output Selection for DMA Channel 1 REQ

Table 5-78. UDMACH1BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	30h	Read only selection value 30h = UART0 RX DMA burst request, controlled by UART0:DMACTL.RXDMAE

5.8.2.62 UDMACH2SSEL Register (Offset = 510h) [Reset = 00000033h]

UDMACH2SSEL is shown in Table 5-79.

Return to the Summary Table.

Output Selection for DMA Channel 2 SREQ

Table 5-79. UDMACH2SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	33h	Read only selection value 33h = UART0 TX DMA single request, controlled by UART0:DMACTL.TXDMAE

5.8.2.63 UDMACH2BSEL Register (Offset = 514h) [Reset = 00000032h]

UDMACH2BSEL is shown in Table 5-80.

Return to the Summary Table.

Output Selection for DMA Channel 2 REQ

Table 5-80. UDMACH2BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	32h	Read only selection value 32h = UART0 TX DMA burst request, controlled by UART0:DMACTL.TXDMAE

5.8.2.64 UDMACH3SSEL Register (Offset = 518h) [Reset = 00000029h]

UDMACH3SSEL is shown in Table 5-81.

Return to the Summary Table.

Output Selection for DMA Channel 3 SREQ

Table 5-81. UDMACH3SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	29h	Read only selection value 29h = SPI0 RX DMA single request, controlled by SPI0:DMACR.RXDMAE

5.8.2.65 UDMACH3BSEL Register (Offset = 51Ch) [Reset = 00000028h]

UDMACH3BSEL is shown in Table 5-82.

Return to the Summary Table.

Output Selection for DMA Channel 3 REQ

Table 5-82. UDMACH3BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	28h	Read only selection value 28h = SPI0 RX DMA burst request , controlled by SPI0:DMACR.RXDMAE

5.8.2.66 UDMACH4SSEL Register (Offset = 520h) [Reset = 0000002Bh]

UDMACH4SSEL is shown in Table 5-83.

Return to the Summary Table.

Output Selection for DMA Channel 4 SREQ

Table 5-83. UDMACH4SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Bh	Read only selection value 2Bh = SPI0 TX DMA single request, controlled by SPI0:DMACR.TXDMAE

5.8.2.67 UDMACH4BSEL Register (Offset = 524h) [Reset = 0000002Ah]

UDMACH4BSEL is shown in Table 5-84.

Return to the Summary Table.

Output Selection for DMA Channel 4 REQ

Table 5-84. UDMACH4BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Ah	Read only selection value 2Ah = SPI0 TX DMA burst request , controlled by SPI0:DMACR.TXDMAE

5.8.2.68 UDMACH5SSEL Register (Offset = 528h) [Reset = 00000035h]

UDMACH5SSEL is shown in Table 5-85.

Return to the Summary Table.

Output Selection for DMA Channel 5 SREQ

Table 5-85. UDMACH5SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	35h	Read only selection value 35h = UART1 RX DMA single request, controlled by UART1:DMACTL.RXDMAE

5.8.2.69 UDMACH5BSEL Register (Offset = 52Ch) [Reset = 00000034h]

UDMACH5BSEL is shown in Table 5-86.

Return to the Summary Table.

Output Selection for DMA Channel 5 REQ

Table 5-86. UDMACH5BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	34h	Read only selection value 34h = UART1 RX DMA burst request, controlled by UART1:DMACTL.RXDMAE

5.8.2.70 UDMACH6SSEL Register (Offset = 530h) [Reset = 00000037h]

UDMACH6SSEL is shown in Table 5-87.

Return to the Summary Table.

Output Selection for DMA Channel 6 SREQ

Table 5-87. UDMACH6SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	37h	Read only selection value 37h = UART1 TX DMA single request, controlled by UART1:DMACTL.TXDMAE

5.8.2.71 UDMACH6BSEL Register (Offset = 534h) [Reset = 00000036h]

UDMACH6BSEL is shown in Table 5-88.

Return to the Summary Table.

Output Selection for DMA Channel 6 REQ

Table 5-88. UDMACH6BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	36h	Read only selection value 36h = UART1 TX DMA burst request, controlled by UART1:DMACTL.TXDMAE

5.8.2.72 UDMACH7SSEL Register (Offset = 538h) [Reset = 00000075h]

UDMACH7SSEL is shown in Table 5-89.

Return to the Summary Table.

Output Selection for DMA Channel 7 SREQ

Table 5-89. UDMACH7SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	75h	Read only selection value 75h = DMA single request event from AUX, configured by AUX_EVCTL:DMACTL

5.8.2.73 UDMACH7BSEL Register (Offset = 53Ch) [Reset = 00000076h]

UDMACH7BSEL is shown in Table 5-90.

Return to the Summary Table.

Output Selection for DMA Channel 7 REQ

Table 5-90. UDMACH7BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	76h	Read only selection value 76h = DMA burst request event from AUX, configured by AUX_EVCTL:DMACTL

5.8.2.74 UDMACH8SSEL Register (Offset = 540h) [Reset = 00000074h]

UDMACH8SSEL is shown in Table 5-91.

Return to the Summary Table.

Output Selection for DMA Channel 8 SREQ
Single request is ignored for this channel

Table 5-91. UDMACH8SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	74h	Read only selection value 74h = DMA sofware trigger from AUX, triggered by AUX_EVCTL:DMASWREQ.START

5.8.2.75 UDMACH8BSEL Register (Offset = 544h) [Reset = 00000074h]

UDMACH8BSEL is shown in Table 5-92.

Return to the Summary Table.

Output Selection for DMA Channel 8 REQ

Table 5-92. UDMACH8BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	74h	Read only selection value 74h = DMA sofware trigger from AUX, triggered by AUX_EVCTL:DMASWREQ.START

5.8.2.76 UDMACH9SSEL Register (Offset = 548h) [Reset = 00000045h]

UDMACH9SSEL is shown in Table 5-93.

Return to the Summary Table.

Output Selection for DMA Channel 9 SREQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT0 as GPT0:RIS.DMAARIS

Table 5-93. UDMACH9SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	45h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 45h = Not used tied to 0 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.77 UDMACH9BSEL Register (Offset = 54Ch) [Reset = 0000004Dh]

UDMACH9BSEL is shown in Table 5-94.

Return to the Summary Table.

Output Selection for DMA Channel 9 REQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT0 as GPT0:RIS.DMAARIS

Table 5-94. UDMACH9BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	4Dh	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.78 UDMACH10SSEL Register (Offset = 550h) [Reset = 00000046h]

UDMACH10SSEL is shown in Table 5-95.

Return to the Summary Table.

Output Selection for DMA Channel 10 SREQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT0 as GPT0:RIS.DMABRIS

Table 5-95. UDMACH10SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	46h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 46h = Not used tied to 0 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.79 UDMACH10BSEL Register (Offset = 554h) [Reset = 0000004Eh]

UDMACH10BSEL is shown in Table 5-96.

Return to the Summary Table.

Output Selection for DMA Channel 10 REQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT0 as GPT0:RIS.DMABRIS

Table 5-96. UDMACH10BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	4Eh	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.80 UDMACH11SSEL Register (Offset = 558h) [Reset = 00000047h]

UDMACH11SSEL is shown in Table 5-97.

Return to the Summary Table.

Output Selection for DMA Channel 11 SREQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT1 as GPT1:RIS.DMAARIS

Table 5-97. UDMACH11SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	47h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 47h = Not used tied to 0 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.81 UDMACH11BSEL Register (Offset = 55Ch) [Reset = 0000004Fh]

UDMACH11BSEL is shown in Table 5-98.

Return to the Summary Table.

Output Selection for DMA Channel 11 REQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT1 as GPT1:RIS.DMAARIS

Table 5-98. UDMACH11BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	4Fh	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.82 UDMACH12SSEL Register (Offset = 560h) [Reset = 00000048h]

UDMACH12SSEL is shown in Table 5-99.

Return to the Summary Table.

Output Selection for DMA Channel 12 SREQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT1 as GPT1:RIS.DMABRIS

Table 5-99. UDMACH12SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	48h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 48h = Not used tied to 0 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.83 UDMACH12BSEL Register (Offset = 564h) [Reset = 00000050h]

UDMACH12BSEL is shown in Table 5-100.

Return to the Summary Table.

Output Selection for DMA Channel 12 REQ
DMA_DONE for the corresponding DMA channel is available as interrupt on GPT1 as GPT1:RIS.DMABRIS

Table 5-100. UDMACH12BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	50h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 79h = Always asserted

5.8.2.84 UDMACH13BSEL Register (Offset = 56Ch) [Reset = 00000003h]

UDMACH13BSEL is shown in Table 5-101.

Return to the Summary Table.

Output Selection for DMA Channel 13 REQ

Table 5-101. UDMACH13BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	3h	Read only selection value 3h = AON programmable event 2. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG2_EV

5.8.2.85 UDMACH14BSEL Register (Offset = 574h) [Reset = 00000001h]

UDMACH14BSEL is shown in Table 5-102.

Return to the Summary Table.

Output Selection for DMA Channel 14 REQ

Table 5-102. UDMACH14BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-102. UDMACH14BSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	1h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 1h = AON programmable event 0. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG0_EV 2h = AON programmable event 1. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG1_EV 3h = AON programmable event 2. Event selected by AON_EVENT MCU event selector, AON_EVENT:EVTOMCUSEL.AON_PROG2_EV 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 8h = Interrupt event from I2S 9h = Interrupt event from I2C Ah = AUX Software event 0, AUX_EVCTL:SWEVSET.SWEV0 Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS Ch = GPT2A interrupt event, controlled by GPT2:TAMR Dh = GPT2B interrupt event, controlled by GPT2:TBMR Eh = GPT3A interrupt event, controlled by GPT3:TAMR Fh = GPT3B interrupt event, controlled by GPT3:TBMR 10h = GPT0A interrupt event, controlled by GPT0:TAMR 11h = GPT0B interrupt event, controlled by GPT0:TBMR 12h = GPT1A interrupt event, controlled by GPT1:TAMR 13h = GPT1B interrupt event, controlled by GPT1:TBMR 14h = DMA done for software tiggered UDMA channel 0, see UDMA0:SOFTREQ 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 16h = DMA done for software tiggered UDMA channel 18, see UDMA0:SOFTREQ 18h = Watchdog interrupt event, controlled by WDT:CTL.INTEN 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Dh = AUX software event 1, triggered by AUX_EVCTL:SWEVSET.SWEV1, also available as AUX_EVENT2 AON wake up event. MCU domain wakeup control AON_EVENT:MCUWUSEL 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 1Fh = PKA Interrupt event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 26h = DMA bus error, corresponds to UDMA0:ERROR.STATUS

Table 5-102. UDMACH14BSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				27h = Combined DMA done, corresponding flags are here UDMA0:REQDONE 28h = SPI0 RX DMA burst request , controlled by SPI0:DMACR.RXDMAE 29h = SPI0 RX DMA single request, controlled by SPI0:DMACR.RXDMAE 2Ah = SPI0 TX DMA burst request , controlled by SPI0:DMACR.TXDMAE 2Bh = SPI0 TX DMA single request, controlled by SPI0:DMACR.TXDMAE 2Ch = SPI1 RX DMA burst request , controlled by SPI0:DMACR.RXDMAE 2Dh = SPI1 RX DMA single request, controlled by SPI0:DMACR.RXDMAE 2Eh = SPI1 TX DMA burst request , controlled by SPI0:DMACR.TXDMAE 2Fh = SPI1 TX DMA single request, controlled by SPI0:DMACR.TXDMAE 30h = UART0 RX DMA burst request, controlled by UART0:DMACTL.RXDMAE 31h = UART0 RX DMA single request, controlled by UART0:DMACTL.RXDMAE 32h = UART0 TX DMA burst request, controlled by UART0:DMACTL.TXDMAE 33h = UART0 TX DMA single request, controlled by UART0:DMACTL.TXDMAE 34h = UART1 RX DMA burst request, controlled by UART1:DMACTL.RXDMAE 35h = UART1 RX DMA single request, controlled by UART1:DMACTL.RXDMAE 36h = UART1 TX DMA burst request, controlled by UART1:DMACTL.TXDMAE 37h = UART1 TX DMA single request, controlled by UART1:DMACTL.TXDMAE 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 4Dh = GPT0A DMA trigger event. Configured by GPT0:DMAEV 4Eh = GPT0B DMA trigger event. Configured by GPT0:DMAEV 4Fh = GPT1A DMA trigger event. Configured by GPT1:DMAEV 50h = GPT1B DMA trigger event. Configured by GPT1:DMAEV 51h = GPT2A DMA trigger event. Configured by GPT2:DMAEV 52h = GPT2B DMA trigger event. Configured by GPT2:DMAEV 53h = GPT3A DMA trigger event. Configured by GPT3:DMAEV 54h = GPT3B DMA trigger event. Configured by GPT3:DMAEV 55h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT0 wil be routed here.

Table 5-102. UDMACH14BSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				56h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT1 wil be routed here. 57h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT2 wil be routed here. 58h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT3 wil be routed here. 59h = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 5Ah = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT4 wil be routed here. 5Bh = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT6 wil be routed here. 5Ch = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT7 wil be routed here. 5Dh = CRYPTO result available interupt event, the corresponding flag is found here CRYPTO:IRQSTAT.RESULT_AVAIL. Controlled by CRYPTO:IRQSTAT.RESULT_AVAIL 5Eh = CRYPTO DMA input done event, the correspondingg flag is CRYPTO:IRQSTAT.DMA_IN_DONE. Controlled by CRYPTO:IRQEN.DMA_IN_DONE 63h = Watchdog non maskable interrupt event, controlled by WDT:CTL.INTTYPE 64h = Software event 0, triggered by SWEV.SWEV0 65h = Software event 1, triggered by SWEV.SWEV1 66h = Software event 2, triggered by SWEV.SWEV2 67h = Software event 3, triggered by SWEV.SWEV3 68h = TRNG Interrupt event, controlled by TRNG:IRQEN.EN 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB 6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 74h = DMA sofware trigger from AUX, triggered by AUX_EVCTL:DMASWREQ.START 75h = DMA single request event from AUX, configured by AUX_EVCTL:DMACTL 76h = DMA burst request event from AUX, configured by AUX_EVCTL:DMACTL

Table 5-102. UDMACH14BSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 78h = CPU halted 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1 7Fh = SPI2 RX DMA burst request , controlled by SPI2:DMACR.RXDMAE 80h = SPI2 RX DMA single request, controlled by SPI2:DMACR.RXDMAE 81h = SPI2 TX DMA burst request , controlled by SPI2:DMACR.TXDMAE 82h = SPI2 TX DMA single request, controlled by SPI2:DMACR.TXDMAE 83h = SPI3 RX DMA burst request , controlled by SPI2:DMACR.RXDMAE 84h = SPI3 RX DMA single request, controlled by SPI2:DMACR.RXDMAE 85h = SPI3 TX DMA burst request , controlled by SPI2:DMACR.TXDMAE 86h = SPI3 TX DMA single request, controlled by SPI2:DMACR.TXDMAE 87h = UART2 RX DMA burst request, controlled by UART2:DMACTL.RXDMAE 88h = UART2 RX DMA single request, controlled by UART2:DMACTL.RXDMAE 89h = UART2 TX DMA burst request, controlled by UART2:DMACTL.TXDMAE 8Ah = UART2 TX DMA single request, controlled by UART2:DMACTL.TXDMAE 8Bh = UART3 RX DMA burst request, controlled by UART3:DMACTL.RXDMAE 8Ch = UART3 RX DMA single request, controlled by UART3:DMACTL.RXDMAE 8Dh = UART3 TX DMA burst request, controlled by UART3:DMACTL.TXDMAE 8Eh = UART3 TX DMA single request, controlled by UART3:DMACTL.TXDMAE

5.8.2.86 UDMACH15BSEL Register (Offset = 57Ch) [Reset = 00000007h]

UDMACH15BSEL is shown in Table 5-103.

Return to the Summary Table.

Output Selection for DMA Channel 15 REQ

Table 5-103. UDMACH15BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7h	Read only selection value 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting

5.8.2.87 UDMACH16SSEL Register (Offset = 580h) [Reset = 0000002Dh]

UDMACH16SSEL is shown in Table 5-104.

Return to the Summary Table.

Output Selection for DMA Channel 16 SREQ

Table 5-104. UDMACH16SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Dh	Read only selection value 2Dh = SPI1 RX DMA single request, controlled by SPI0:DMACR.RXDMAE

5.8.2.88 UDMACH16BSEL Register (Offset = 584h) [Reset = 0000002Ch]

UDMACH16BSEL is shown in Table 5-105.

Return to the Summary Table.

Output Selection for DMA Channel 16 REQ

Table 5-105. UDMACH16BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Ch	Read only selection value 2Ch = SPI1 RX DMA burst request , controlled by SPI0:DMACR.RXDMAE

5.8.2.89 UDMACH17SSEL Register (Offset = 588h) [Reset = 0000002Fh]

UDMACH17SSEL is shown in Table 5-106.

Return to the Summary Table.

Output Selection for DMA Channel 17 SREQ

Table 5-106. UDMACH17SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Fh	Read only selection value 2Fh = SPI1 TX DMA single request, controlled by SPI0:DMACR.TXDMAE

5.8.2.90 UDMACH17BSEL Register (Offset = 58Ch) [Reset = 0000002Eh]

UDMACH17BSEL is shown in Table 5-107.

Return to the Summary Table.

Output Selection for DMA Channel 17 REQ

Table 5-107. UDMACH17BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	2Eh	Read only selection value 2Eh = SPI1 TX DMA burst request , controlled by SPI0:DMACR.TXDMAE

5.8.2.91 UDMACH19SSEL Register (Offset = 598h) [Reset = 00000080h]

UDMACH19SSEL is shown in Table 5-108.

Return to the Summary Table.

Output Selection for DMA Channel 19 SREQ

Table 5-108. UDMACH19SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	80h	Read only selection value 80h = SPI2 RX DMA single request, controlled by SPI2:DMACR.RXDMAE

5.8.2.92 UDMACH19BSEL Register (Offset = 59Ch) [Reset = 0000007Fh]

UDMACH19BSEL is shown in Table 5-109.

Return to the Summary Table.

Output Selection for DMA Channel 19 REQ

Table 5-109. UDMACH19BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	7Fh	Read only selection value 7Fh = SPI2 RX DMA burst request , controlled by SPI2:DMACR.RXDMAE

5.8.2.93 UDMACH20SSEL Register (Offset = 5A0h) [Reset = 00000082h]

UDMACH20SSEL is shown in Table 5-110.

Return to the Summary Table.

Output Selection for DMA Channel 20 SREQ

Table 5-110. UDMACH20SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	82h	Read only selection value 82h = SPI2 TX DMA single request, controlled by SPI2:DMACR.TXDMAE

5.8.2.94 UDMACH20BSEL Register (Offset = 5A4h) [Reset = 00000081h]

UDMACH20BSEL is shown in Table 5-111.

Return to the Summary Table.

Output Selection for DMA Channel 20 REQ

Table 5-111. UDMACH20BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	81h	Read only selection value 81h = SPI2 TX DMA burst request , controlled by SPI2:DMACR.TXDMAE

5.8.2.95 UDMACH21SSEL Register (Offset = 5A8h) [Reset = 00000064h]

UDMACH21SSEL is shown in Table 5-112.

Return to the Summary Table.

Output Selection for DMA Channel 21 SREQ

Table 5-112. UDMACH21SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	64h	Read only selection value 64h = Software event 0, triggered by SWEV.SWEV0

5.8.2.96 UDMACH21BSEL Register (Offset = 5ACh) [Reset = 00000064h]

UDMACH21BSEL is shown in Table 5-113.

Return to the Summary Table.

Output Selection for DMA Channel 21 REQ

Table 5-113. UDMACH21BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	64h	Read only selection value 64h = Software event 0, triggered by SWEV.SWEV0

5.8.2.97 UDMACH22SSEL Register (Offset = 5B0h) [Reset = 00000065h]

UDMACH22SSEL is shown in Table 5-114.

Return to the Summary Table.

Output Selection for DMA Channel 22 SREQ

Table 5-114. UDMACH22SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	65h	Read only selection value 65h = Software event 1, triggered by SWEV.SWEV1

5.8.2.98 UDMACH22BSEL Register (Offset = 5B4h) [Reset = 00000065h]

UDMACH22BSEL is shown in Table 5-115.

Return to the Summary Table.

Output Selection for DMA Channel 22 REQ

Table 5-115. UDMACH22BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	65h	Read only selection value 65h = Software event 1, triggered by SWEV.SWEV1

5.8.2.99 UDMACH23SSEL Register (Offset = 5B8h) [Reset = 00000066h]

UDMACH23SSEL is shown in Table 5-116.

Return to the Summary Table.

Output Selection for DMA Channel 23 SREQ

Table 5-116. UDMACH23SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	66h	Read only selection value 66h = Software event 2, triggered by SWEV.SWEV2

5.8.2.100 UDMACH23BSEL Register (Offset = 5BCh) [Reset = 00000066h]

UDMACH23BSEL is shown in Table 5-117.

Return to the Summary Table.

Output Selection for DMA Channel 23 REQ

Table 5-117. UDMACH23BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	66h	Read only selection value 66h = Software event 2, triggered by SWEV.SWEV2

5.8.2.101 UDMACH24SSEL Register (Offset = 5C0h) [Reset = 00000067h]

UDMACH24SSEL is shown in Table 5-118.

Return to the Summary Table.

Output Selection for DMA Channel 24 SREQ

Table 5-118. UDMACH24SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	67h	Read only selection value 67h = Software event 3, triggered by SWEV.SWEV3

5.8.2.102 UDMACH24BSEL Register (Offset = 5C4h) [Reset = 00000067h]

UDMACH24BSEL is shown in Table 5-119.

Return to the Summary Table.

Output Selection for DMA Channel 24 REQ

Table 5-119. UDMACH24BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	67h	Read only selection value 67h = Software event 3, triggered by SWEV.SWEV3

5.8.2.103 UDMACH25SSEL Register (Offset = 5C8h) [Reset = 00000084h]

UDMACH25SSEL is shown in Table 5-120.

Return to the Summary Table.

Output Selection for DMA Channel 25 SREQ

Table 5-120. UDMACH25SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	84h	Read only selection value 84h = SPI3 RX DMA single request, controlled by SPI2:DMACR.RXDMAE

5.8.2.104 UDMACH25BSEL Register (Offset = 5CCh) [Reset = 00000083h]

UDMACH25BSEL is shown in Table 5-121.

Return to the Summary Table.

Output Selection for DMA Channel 25 REQ

Table 5-121. UDMACH25BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	83h	Read only selection value 83h = SPI3 RX DMA burst request , controlled by SPI2:DMACR.RXDMAE

5.8.2.105 UDMACH26SSEL Register (Offset = 5D0h) [Reset = 00000086h]

UDMACH26SSEL is shown in Table 5-122.

Return to the Summary Table.

Output Selection for DMA Channel 26 SREQ

Table 5-122. UDMACH26SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	86h	Read only selection value 86h = SPI3 TX DMA single request, controlled by SPI2:DMACR.TXDMAE

5.8.2.106 UDMACH26BSEL Register (Offset = 5D4h) [Reset = 00000085h]

UDMACH26BSEL is shown in Table 5-123.

Return to the Summary Table.

Output Selection for DMA Channel 26 REQ

Table 5-123. UDMACH26BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	85h	Read only selection value 85h = SPI3 TX DMA burst request , controlled by SPI2:DMACR.TXDMAE

5.8.2.107 UDMACH28SSEL Register (Offset = 5E0h) [Reset = 00000088h]

UDMACH28SSEL is shown in Table 5-124.

Return to the Summary Table.

Output Selection for DMA Channel 28 SREQ

Table 5-124. UDMACH28SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	88h	Read only selection value 88h = UART2 RX DMA single request, controlled by UART2:DMACTL.RXDMAE

5.8.2.108 UDMACH28BSEL Register (Offset = 5E4h) [Reset = 00000087h]

UDMACH28BSEL is shown in Table 5-125.

Return to the Summary Table.

Output Selection for DMA Channel 28 REQ

Table 5-125. UDMACH28BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	87h	Read only selection value 87h = UART2 RX DMA burst request, controlled by UART2:DMACTL.RXDMAE

5.8.2.109 UDMACH29SSEL Register (Offset = 5E8h) [Reset = 0000008Ah]

UDMACH29SSEL is shown in Table 5-126.

Return to the Summary Table.

Output Selection for DMA Channel 29 SREQ

Table 5-126. UDMACH29SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8Ah	Read only selection value 8Ah = UART2 TX DMA single request, controlled by UART2:DMACTL.TXDMAE

5.8.2.110 UDMACH29BSEL Register (Offset = 5ECh) [Reset = 00000089h]

UDMACH29BSEL is shown in Table 5-127.

Return to the Summary Table.

Output Selection for DMA Channel 29 REQ

Table 5-127. UDMACH29BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	89h	Read only selection value 89h = UART2 TX DMA burst request, controlled by UART2:DMACTL.TXDMAE

5.8.2.111 UDMACH30SSEL Register (Offset = 5F0h) [Reset = 0000008Ch]

UDMACH30SSEL is shown in Table 5-128.

Return to the Summary Table.

Output Selection for DMA Channel 30 SREQ

Table 5-128. UDMACH30SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8Ch	Read only selection value 8Ch = UART3 RX DMA single request, controlled by UART3:DMACTL.RXDMAE

5.8.2.112 UDMACH30BSEL Register (Offset = 5F4h) [Reset = 0000008Bh]

UDMACH30BSEL is shown in Table 5-129.

Return to the Summary Table.

Output Selection for DMA Channel 30 REQ

Table 5-129. UDMACH30BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8Bh	Read only selection value 8Bh = UART3 RX DMA burst request, controlled by UART3:DMACTL.RXDMAE

5.8.2.113 UDMACH31SSEL Register (Offset = 5F8h) [Reset = 0000008Eh]

UDMACH31SSEL is shown in Table 5-130.

Return to the Summary Table.

Output Selection for DMA Channel 31 SREQ

Table 5-130. UDMACH31SSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8Eh	Read only selection value 8Eh = UART3 TX DMA single request, controlled by UART3:DMACTL.TXDMAE

5.8.2.114 UDMACH31BSEL Register (Offset = 5FCh) [Reset = 0000008Dh]

UDMACH31BSEL is shown in Table 5-131.

Return to the Summary Table.

Output Selection for DMA Channel 31 REQ

Table 5-131. UDMACH31BSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	8Dh	Read only selection value 8Dh = UART3 TX DMA burst request, controlled by UART3:DMACTL.TXDMAE

5.8.2.115 GPT3ACAPTSEL Register (Offset = 600h) [Reset = 0000005Bh]

GPT3ACAPTSEL is shown in Table 5-132.

Return to the Summary Table.

Output Selection for GPT3 0

Table 5-132. GPT3ACAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-132. GPT3ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	5Bh	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 5Bh = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT6 wil be routed here. 5Ch = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT7 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-132. GPT3ACAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.116 GPT3BCAPTSEL Register (Offset = 604h) [Reset = 0000005Ch]

GPT3BCAPTSEL is shown in Table 5-133.

Return to the Summary Table.

Output Selection for GPT3 1

Table 5-133. GPT3BCAPTSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved

Table 5-133. GPT3BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-0	EV	R/W	5Ch	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 4h = Edge detect event from IOC. Configureded by the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET settings 5h = Combined event from battery monitor 6h = Combined event from Oscillator control 7h = Event from AON_RTC, controlled by the AON_RTC:CTL.COMB_EV_MASK setting 9h = Interrupt event from I2C Bh = AUX combined event, the corresponding flag register is here AUX_EVCTL:EVTOMCUFLAGS 15h = FLASH controller error event, the status flags are FLASH:FEDACSTAT.FSM_DONE and FLASH:FEDACSTAT.RVF_INT 19h = RFC Doorbell Command Acknowledgement Interrupt, equvialent to RFC_DBELL:RFACKIFG.ACKFLAG 1Ah = Combined RFC hardware interrupt, corresponding flag is here RFC_DBELL:RFHWIFG 1Bh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE0 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_0 event 1Eh = Combined Interrupt for CPE Generated events. Corresponding flags are here RFC_DBELL:RFCPEIFG. Only interrupts selected with CPE1 in RFC_DBELL:RFCPEIFG can trigger a RFC_CPE_1 event 22h = SPI0 combined interrupt, interrupt flags are found here SPI0:MIS 23h = SPI1 combined interrupt, interrupt flags are found here SPI1:MIS 24h = UART0 combined interrupt, interrupt flags are found here UART0:MIS 25h = UART1 combined interrupt, interrupt flags are found here UART1:MIS 38h = AUX Timer2 event 0, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV0 39h = AUX Timer2 event 1, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV1 3Ah = AUX Timer2 event 2, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV2 3Bh = AUX Timer2 event 3, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_EV3 3Ch = AUX Timer2 pulse, corresponding to flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER2_PULSE 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 5Bh = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT6 wil be routed here. 5Ch = Port capture event from IOC, configured by IOC:IOCFGn.PORT_ID. Events on ports configured with ENUM PORT_EVENT7 wil be routed here. 69h = AON wakeup event, the corresponding flag is here AUX_EVCTL:EVTOMCUFLAGS.AUX_WU_EV 6Ah = AUX Compare A event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPA 6Bh = AUX Compare B event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_COMPB

Table 5-133. GPT3BCAPTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				6Ch = AUX TDC measurement done event, corresponds to the flag AUX_EVCTL:EVTOMCUFLAGS.AUX_TDC_DONE and the AUX_TDC status AUX_TDC:STAT.DONE 6Dh = AUX timer 0 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER0_EV 6Eh = AUX timer 1 event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_TIMER1_EV 6Fh = Autotake event from AUX semaphore, configured by AUX_SMPH:AUTOTAKE 70h = AUX ADC done, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_DONE 71h = AUX ADC FIFO watermark event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL 72h = Loopback of OBSMUX0 through AUX, corresponds to AUX_EVCTL:EVTOMCUFLAGS.MCU_OBSMUX0 73h = AUX ADC interrupt event, corresponds to AUX_EVCTL:EVTOMCUFLAGS.AUX_ADC_IRQ. Status flags are found here AUX_EVCTL:EVTOMCUFLAGS 77h = RTC periodic event controlled by AON_RTC:CTL.RTC_UPD_EN 79h = Always asserted 7Ah = SPI2 combined interrupt, interrupt flags are found here SPI2:MIS 7Bh = SPI3 combined interrupt, interrupt flags are found here SPI3:MIS 7Ch = UART2 combined interrupt, interrupt flags are found here UART2:MIS 7Dh = UART3 combined interrupt, interrupt flags are found here UART3:MIS 7Eh = Interrupt event from I2C1

5.8.2.117 AUXSEL0 Register (Offset = 700h) [Reset = 00000010h]

AUXSEL0 is shown in Table 5-134.

Return to the Summary Table.

Output Selection for AUX Subscriber 0

Table 5-134. AUXSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	10h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive Ch = GPT2A interrupt event, controlled by GPT2:TAMR Dh = GPT2B interrupt event, controlled by GPT2:TBMR Eh = GPT3A interrupt event, controlled by GPT3:TAMR Fh = GPT3B interrupt event, controlled by GPT3:TBMR 10h = GPT0A interrupt event, controlled by GPT0:TAMR 11h = GPT0B interrupt event, controlled by GPT0:TBMR 12h = GPT1A interrupt event, controlled by GPT1:TAMR 13h = GPT1B interrupt event, controlled by GPT1:TBMR 3Dh = GPT0A compare event. Configured by GPT0:TAMR.TCACT 3Eh = GPT0B compare event. Configured by GPT0:TBMR.TCACT 3Fh = GPT1A compare event. Configured by GPT1:TAMR.TCACT 40h = GPT1B compare event. Configured by GPT1:TBMR.TCACT 41h = GPT2A compare event. Configured by GPT2:TAMR.TCACT 42h = GPT2B compare event. Configured by GPT2:TBMR.TCACT 43h = GPT3A compare event. Configured by GPT3:TAMR.TCACT 44h = GPT3B compare event. Configured by GPT3:TBMR.TCACT 79h = Always asserted

5.8.2.118 CM3NMISEL0 Register (Offset = 800h) [Reset = 00000063h]

CM3NMISEL0 is shown in Table 5-135.

Return to the Summary Table.

Output Selection for NMI Subscriber 0

Table 5-135. CM3NMISEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R	63h	Read only selection value 63h = Watchdog non maskable interrupt event, controlled by WDT:CTL.INTTYPE

5.8.2.119 I2SSTMPSEL0 Register (Offset = 900h) [Reset = 0000005Fh]

I2SSTMPSEL0 is shown in Table 5-136.

Return to the Summary Table.

Output Selection for I2S Subscriber 0

Table 5-136. I2SSTMPSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	5Fh	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 79h = Always asserted

5.8.2.120 FRZSEL0 Register (Offset = A00h) [Reset = 00000078h]

FRZSEL0 is shown in Table 5-137.

Return to the Summary Table.

Output Selection for FRZ Subscriber
The halted debug signal is passed to peripherals such as the General Purpose Timer, Sensor Controller with Digital and Analog Peripherals (AUX), Radio, and RTC. When the system CPU halts, the connected peripherals that have freeze enabled also halt. The programmable output can be set to static values of 0 or 1, and can also be set to pass the halted signal.

Table 5-137. FRZSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	EV	R/W	78h	Read/write selection value Writing any other value than values defined by a ENUM may result in undefined behavior. 0h = Always inactive 78h = CPU halted 79h = Always asserted

5.8.2.121 SWEV Register (Offset = F00h) [Reset = 00000000h]

SWEV is shown in Table 5-138.

Return to the Summary Table.

Set or Clear Software Events

Table 5-138. SWEV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	SWEV3	R/W	0h	Writing "1" to this bit when the value is "0" triggers the Software 3 event.
23-17	RESERVED	R	0h	Reserved
16	SWEV2	R/W	0h	Writing "1" to this bit when the value is "0" triggers the Software 2 event.
15-9	RESERVED	R	0h	Reserved
8	SWEV1	R/W	0h	Writing "1" to this bit when the value is "0" triggers the Software 1 event.
7-1	RESERVED	R	0h	Reserved
0	SWEV0	R/W	0h	Writing "1" to this bit when the value is "0" triggers the Software 0 event.

Chapter 6

JTAG Interface

This chapter describes the cJTAG and JTAG interface for on-chip debug support.

Table 6-1. References

ID	Description
[JTAG 1]	IEEE Standard Test Access Port and Boundary Scan Architecture, IEEE Std 1149.1a 1993 and Supplement Std. 1149.1b 1994, The Institute of Electrical and Electronics Engineers, Inc.
[JTAG 2]	IEEE 1149.7 Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

6.1 Overview 462
6.2 cJTAG 464
6.3 ICEPick 465
6.4 ICEMelter 475
6.5 Serial Wire Viewer (SWV) 476
6.6 Halt In Boot (HIB) 476
6.7 Debug and Shutdown 476
6.8 Boundary Scan 476

6.1 Overview

In the CC13x4x10 and CC26x4x10 device platform, the JTAG subsystem implements two IEEE standards for debug and test purposes:

- **IEEE standard 1149.1**: Standard Test Access Port and Boundary Scan Architecture Test Access Port (TAP) [JTAG 1]. This standard is known by the acronym JTAG.
- **Class 4 IEEE 1149.7**: Standard for Reduced-pin and Enhanced-functionality Test Access Port and Boundary-scan Architecture [JTAG 2]. This is known by acronym cJTAG (compact JTAG). This standard serializes the IEEE 1149.1 transactions using a variety of compression formats to reduce the number of pins needed to implement a JTAG debug port.

The JTAG interface is used for the following:

- Flash programming
- Debug access
- Test / Boundary Scan
- Power Profiling

The JTAG subsystem also implements a firewall for unauthorized access to debug/test ports. Figure 6-1 shows a block diagram of the JTAG subsystem.

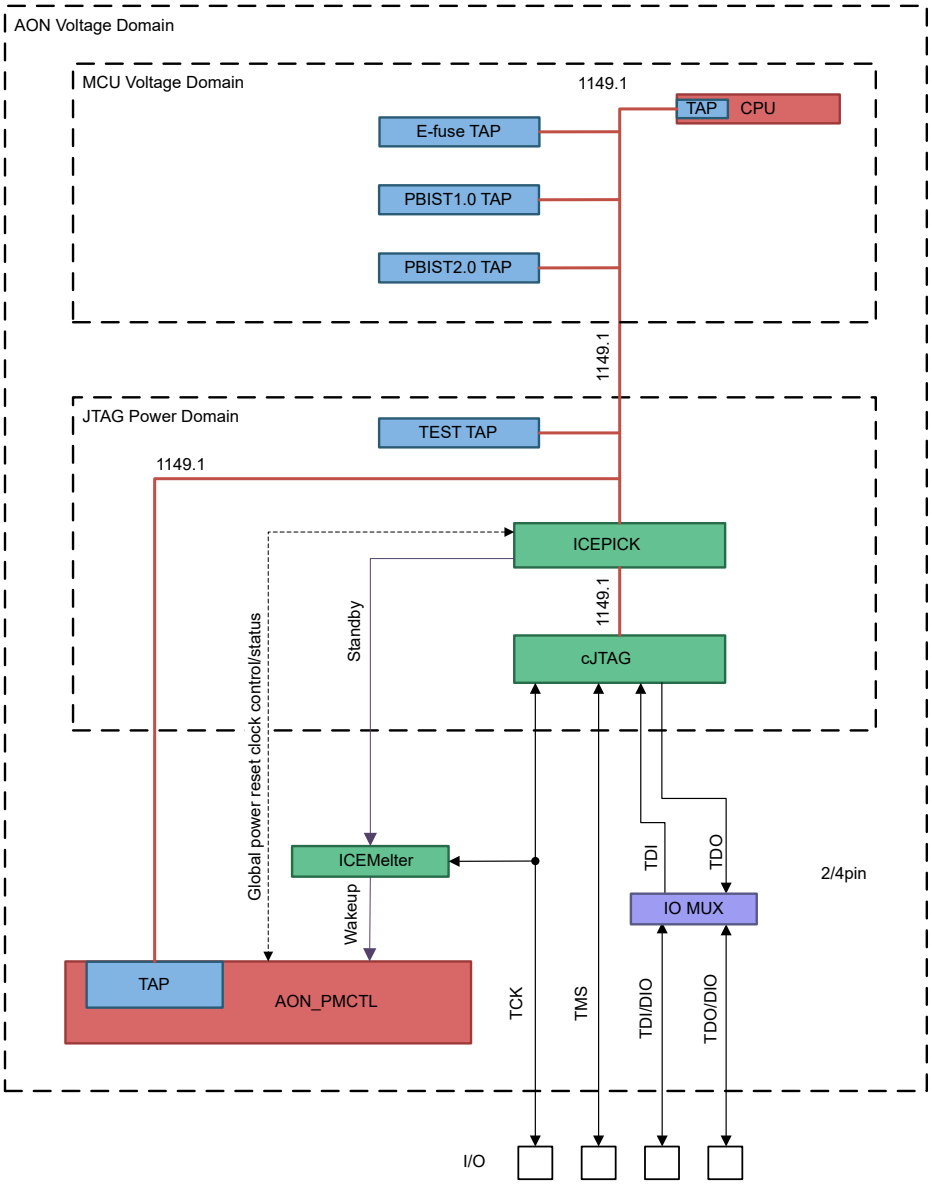


Figure 6-1. JTAG Subsystem

The IEEE 1149.1 TAP uses the following signals to support the operation:

- **TCK (Test Clock):** This signal synchronizes the internal state machine operations.
- **TMS (Test Mode Select):** This signal is sampled at the rising edge of TCK to determine the next state.
- **TDI (Test Data In):** This signal represents the data shifted into the test or programming logic of the device. TDI is sampled at the rising edge of TCK when the internal state machine is in the correct state.
- **TDO (Test Data Out):** This signal represents the data shifted out of the test or programming logic of the device and is valid on the falling edge of TCK when the internal state machine is in the correct state.

There is no dedicated I/O pin for TRST. The JTAG subsystem is reset with system-wide resets and power-on reset.

6.2 cJTAG

This module implements IEEE 1149.7 compliant compact JTAG (cJTAG) adapter, which runs a 2-pin communication protocol on top of a IEEE 1149.1 JTAG test access port (TAP). The 2-pin JTAG mode using only TCK and TMS is the default configuration after power up. The cJTAG configuration in the CC13x4x10 and CC26x4x10 device platform implements a subset of class 4 feature scan modes. Class 4 inherits features from classes 0, 1, 2, and 3 (except the features mentioned in Table 6-2). Figure 6-2 shows a conceptual diagram of the cJTAG module.

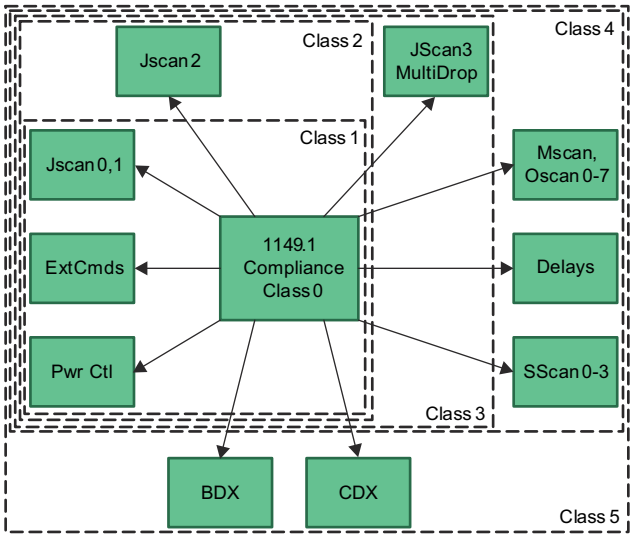


Figure 6-2. cJTAG Conceptual Diagram

- **Class 0:** Strict compliance to IEEE 1149.1 specification with internal TAP selection
- **Class 1:** Adds cJTAG command protocol, some optional discrete commands
- **Class 2:** Adds serial select capability
- **Class 3:** Adds JTAG star configuration, controller IDs and scan selection directives
- **Class 4:** Adds advanced scan protocols

Table 6-2 lists the features in IEEE 1149.7 that are supported in the CC13x4x10 and CC26x4x10 device platform. The cJTAG module in the CC13x4x10 and CC26x4x10 device platform supports 12 scan formats. The scan formats use a variety of compression protocols ranging from 1 to 4 clocks per bit to serialize each packet.

Table 6-2. IEEE 1149.7 Feature Subset

	IEEE 1149.7 Feature	Device Support Through cJTAG	Comment
Configuration	Class 4 TAP	Yes	Supports 2-pin operation
	Class 5 TAP	No	Data and custom channels for background data transfer

Table 6-2. IEEE 1149.7 Feature Subset (continued)

	IEEE 1149.7 Feature	Device Support Through cJTAG	Comment
Optional components	FRST	No	Functional reset
	TRST	No	Test reset
	RDBK capability	No	Readback of register data
	Aux pin functions	Yes	Reuse of TDI and TDO pins
	TCKWID	No	Programmable TCK width
Power control	Power down logic	No	Power down logic capability for cJTAG module
Scan formats	JScan0	Yes	Parallel mode
	JScan1	Yes	Parallel with firewall
	JScan2	Yes	Parallel with super bypass select
	JScan3	Yes	Parallel with register select
	MScan	No	Multidevice mode, supports stalls
	OScan0	Yes	Supports stalls
	OScan1	Yes	Non-stall mode
	OScan2	Yes	Bidirectional transfers, pipelined
	OScan3	Yes	Host to target only, pipelined
	OScan4	Yes	Supports stalls
	OScan5	Yes	Pipelined
	OScan6	Yes	Bidirectional transfers, pipelined
	OScan7	Yes	Host to target only, pipelined
	SScan0	No	Segmented scan
	SScan1	No	Segmented scan, supports stalls
	SScan2	No	Segmented scan
	SScan3	No	Segmented scan, supports stalls

Table 6-3. OScan Scan Packet Contents

Scan Format	Nonshift States				Shift States			
OScan0	nTDI	TMS	RDY	TDO	nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS		TDO	nTDI	TMS		TDO
OScan2		TMS			nTDI	TMS		TDO
OScan3		TMS			nTDI	TMS		
OScan4	nTDI	TMS	RDY	TDO	nTDI		RDY	TDO
OScan5	nTDI	TMS		TDO	nTDI			TDO
OScan6		TMS			nTDI	TMS(1)		TDO
OScan7		TMS			nTDI			

(1) TMS is present for the first packet of the shift.

6.3 ICEPick

ICEPick is the primary TAP in the chip. It acts as the IEEE 1149.1 JTAG-compliant top-level router for the chip. Conceptually, ICEPick can be viewed as a bank of switches that can connect or isolate module-level TAPs to and from the higher level chip TAP. The module-level TAPs are called secondary TAPs, while the primary TAP and external JTAG signals are called the master scan path. The ICEPick TAP appears as the first TAP and only TAP in the scan path following a power on. None of the secondary TAPs are selected or visible in the master scan path. From the perspective of the external JTAG interface, secondary TAPs that are not selected appear to not exist. The ICEPick TAP has several scan paths of its own to support secondary TAP selection, control, and

status. ICEPick enables dynamic scan chain management and can select one or several slave TAPs and link them in the scan chain.

A number of control bits are associated with each secondary TAP within ICEPick. Some of these bits apply strictly to the TAP being managed by ICEPick, while others apply to the whole subsystem or power domain in which the secondary TAP resides. These control bits deal with the TAP selection for inclusion in the scan path, secondary TAP test reset management, and debug attention needed.

A number of status bits are associated with each secondary TAP within ICEPick. These status bits report the accessibility, visibility, power, clock states, and reset status.

The communication protocol can be changed to 4-pin configuration after establishing connection between debug application and on chip cJTAG TAP using 2-pin mode. When cJTAG switches to 4-pin mode, TDI and TDO are mapped automatically to pins through IOC and this has precedence over any other function that was mapped to corresponding DIOs before switching occurs. Switching from 4-pin to 2-pin mode is also supported.

6.3.1 Secondary TAPs

Each secondary TAP is assigned a number. The TAP numbering is linear and starts with 0. The number assigned to a secondary TAP corresponds to its location within the secondary control and status registers in ICEPick. The first selected TAP is the TAP with the lowest number, while the last selected TAP is the TAP with the highest number. The ICEPick module has a firewall for unauthorized access of slave TAPs. Table 6-4 lists the available TAPs, their corresponding order, and the availability of these TAPs for end user. The open TAPs can be locked by writing to the corresponding field in the customer configuration area.

Table 6-4. Slave TAP Order

Number	Test TAP Name	Description	Availability for End User
Test Banks			
0	TEST	DFT functionalities and profiler	See (1)
1	PBIST1.0	RAM BIST controller interface	Locked
2	PBIST2.0	ROM BIST controller interface	Locked
3	eFuse	eFuse interface for SRAM repair	Locked
4	Reserved	Reserved	Reserved
5	AON_PMCTL	AON Power Management Controller	See (2)
Debug Banks			
0	CM33	DAP for Cortex-M33 debug	See (2) (3)

(1) All features in the TEST TAP are locked for end user except the power profiler register. The access to the power profiler register can be blocked by writing to the corresponding field in the customer configuration area.
(2) Some of the features in AON_PMCTL TAP are open for end user. This includes registers for requesting chip erase, system reset, and MCU reset.
(3) The access to debug port of the CPU can be blocked by writing to corresponding field in customer configuration area.

6.3.1.1 Slave DAP (CPU DAP)

The debug subsystem has only one slave DAP (CPU DAP). This debug port implements Cortex®- DAP interface, which allows external access to an Advanced High-performance Bus Access Port (AHB-AP) interface for debug accesses in the CPU.

The Cortex®- DAP is a standard Arm® CoreSight™ debug port that can be configured as JTAG Debug Port (JTAG-DP), Serial Wire Debug Port (SW-DP) or Serial Wire/JTAG Debug Port (SWJ-DP). Even though the SW-DP interface is supported by SWJ-DP, the CC13x4x10 and CC26x4x10 device platform does not use this mode. The Cortex®- DAP is configured to work in JTAG-DP mode only. The key reason is that SW-DP becomes redundant for the design in the presence of the 2-pin JTAG (1149.7) mode.

6.3.2 ICEPick Registers

Table 6-5 lists the control and status registers in ICEPick.

Table 6-5. Register Summary

Register	Abbreviation	Width	Number	Description
Data Shift register	DSR	32	1	TAP Data register
Instruction register	IR	6	1	TAP Instruction register
Bypass register	Bypass	1	1	Used by the BYPASS instruction
Device Identification register	TAPID	32	1	Device ID used with IDCODE
User Code register	UC	32	1	User Code used with USERCODE
ICEPick Identification	IPID	32	1	Version of ICEPick
Connect	Connect	7	1	Connect code
Secondary Debug TAP register (SDTR)	SDTR	24	1	One register exists for each debug TAP instantiated. It is used to control selection, power, reset, and the clock associated with each TAP.
Secondary Test TAP register (STTR)	STTR	24	6	One register exists for each test TAP instantiated. It is used to control selection of each TAP.
Reserved		24	1	Reserved
Linking Mode	LMR	24	1	Specifies how ICEPick manages the TAP selection.
ICEPick Control	IPCR	24	1	General ICEPick control

6.3.2.1 IR Instructions

The ICEPick TAP supports the instructions listed in Table 6-6. All unused TAP controller instructions default to the bypass register. Several instructions are reserved for extensions to the ICEPick opcodes. See Section 6.3.2.5 for device identification register descriptions.

Table 6-6. Instruction Register Opcodes

IR	ICEPick Instruction	Access
000000, 111111	BYPASS	Always-open
10	ROUTER	Connected
100	IDCODE	Always-open
101	ICEPICKCODE	Always-open
111	CONNECT	Always-open
1000	USERCODE	Always-open
000001, 000011, 000110, 001001–111110	Reserved	Reserved

6.3.2.2 Data Shift Register

Figure 6-3 is the register used to shift bits between the ICEPick TDI and TDO. This register is 32 bits wide. The data shift register has multiple shift in points to facilitate shifts on the instruction path and several of the data paths.

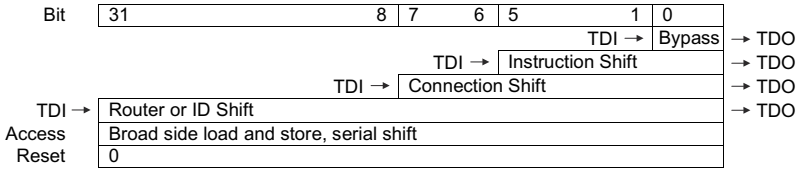


Figure 6-3. Data Shift Register

When asked to shift, 1 bit is shifted from each bit into the next lower bit. A new value is shifted in from TDI while the least significant bit is shifted out to TDO. The shift register has several insertion points based on the current TAP state or value in the instruction register.

6.3.2.3 Instruction Register

This register contains the current TAP instruction. The ICEPick IR is 6 bits wide (see Figure 6-4).

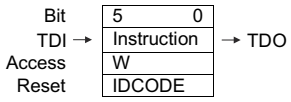


Figure 6-4. Instruction Register

For valid IR opcodes, see Table 6-6.

6.3.2.4 Bypass Register

This register is a 1-bit register (see Figure 6-5). The value that is scanned in TDI is preserved and scanned out of TDO one TCK cycle later.

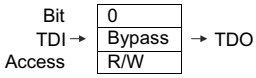


Figure 6-5. Bypass Register

6.3.2.5 Device Identification Register

This register allows the manufacturer, part number, and version of the device to be determined through the TAP (see Figure 6-6). The device identification register is scanned in response to the IDCODE instruction.

IDCODE has three fields: version, part number, and manufacturer.

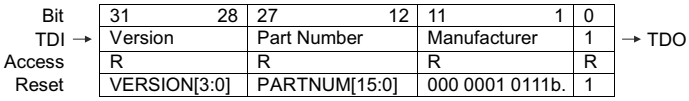


Figure 6-6. Device Identification Register

The contents of this register are replicated to a device configuration area that is memory mapped. For details of this register, see FCFG1:ICEPICK_DEVICE_ID in Section 11.4.

Table 6-7. Device Identification Register Description

Field	Width	Description
Version	4	Revision of the device
Part Number	16	Part number of the device
Manufacturer	11	TI's JEDEC bank and company code: 00000010111b
0	1	This bit is always 1.

6.3.2.6 User Code Register

The User Code register helps to distinguish between the devices built from the same chip. The User Code register value is set through eFuse. Each variant is uniquely identified by feature set or pinned out interface.

The User Code register is a 32-bit register that specifies the version and part number of the component. The content of this register is replicated to the device configuration area that is memory mapped. For details of this register, see Figure 6-7 and Table 6-8.

Bit	31 28	27 12	11 1	0	
TDI →	Version	Variant Number	Reserved	1	→ TDO
Access	R	R	R	R	
Reset	VERSION[3:0]	VARIANT[15:0]	0	1	

Figure 6-7. User Code Register

Table 6-8. User Code Register Description

Field	Width	Description
Version	4	Revision of the device. This field must change each time that the logic or mask set of the device is revised. The initial value is 0.
Variant Number	16	Variant of chip. The decoding of this field is shown in FCFG1:USER_ID (see Section 11.4).
Reserved	11	0
0	1	Bit 0 is always 1

6.3.2.7 ICEPick Identification Register

This register indicates the features and version of the ICEPick module (do not confuse the ICEPick IR with the device IR).

The ID register is a 32-bit register that specifies the version and features of the ICEPick module. For a description of the ICEPick IR, see Figure 6-8 and Table 6-9.

Bit	31 24	23 20	19 16	15 4	3 0	
TDI →	Version	Test TAPs	EMU TAPs	ICEPick Type	Capabilities	→ TDO
Access	R	R	R	R	R	
Reset	0x41	6	1	0x1CC	*	

Figure 6-8. ICEPick Identification Register

Table 6-9. ICEPick Identification Register Description

Field	Width	Description
Version	8	Revision of ICEPick
Test TAPs	4	Number of Test TAPs
EMU TAPs	4	Number of EMU TAPs
ICEpick Type	12	An identifier of the ICEpick Type This field is set to 0x1CC, which corresponds to Type C.
Capabilities	4	Reserved

6.3.2.8 Connect Register

This register guards the device from noise, hot connection of a debug probe cable, or accidental scan by a misconfigured scan controller. This register reduces the chances of accidentally engaging debug functions due to noise or accidental scans. For more details, see Figure 6-9 and Table 6-10.

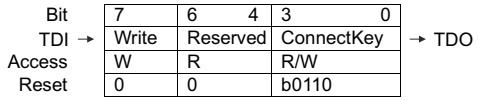


Figure 6-9. Connect Register

Table 6-10. Connect Register Description

Bit	Field	Width	Type	Reset	Description
7	Write Enable	1	W	0	Must be 1 to write the Connect Key. When read, a value of 0 is returned.
6–4	Reserved	3	R	0	Reserved
3–0	ConnectKey	4	R/W	0110	When this field holds the key code of 1001, the scan controller is considered to be connected. All other values are in the not-connected state. In this state, only a limited number of IR instructions are valid.

6.3.3 Router Scan Chain

This register accesses all TAP linking and control registers. The scan chain is 32 bits long. For more information, see Figure 6-10 and Table 6-11.

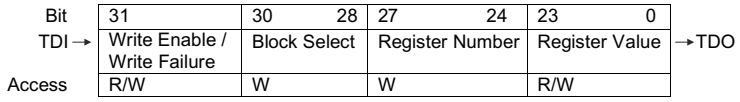


Figure 6-10. ROUTER DR Scan Chain

Table 6-11. ROUTER DR Scan Chain Description

Bit	Field	Width	Type	Reset	Description
31	Write Enable	1	W	0	On scan-in: 0: Only a read is performed. 1: A write to the specified register is performed. On scan-out: If the previous scan resulted in a write to a ROUTER addressed register, then when bit 31 is scanned out during the next trip through the Shift DR state, it indicates whether the previous write succeeded. If 1, the previous write failed. If 0, the previous write was successful. A write to a debug or test secondary TAP control and status register may fail for a number of reasons including: • ICEPick is in the disconnected state. • The TapPresent bit is 0, which indicates that a TAP does not exist at this location. • The TapEnable bit is 0, which indicates that security or other reasons are currently preventing access to this TAP. • A previous programming of the ResetControl or ReleaseFromWIR bits has not been processed yet.
30–28	Block Select	3	R/W	000	Block select: 000: ICEPick Control (see Section 6.3.4.1) 001: Test TAP Linking Control Block (see Section 6.3.4.2) 010: Debug TAP Linking Control Block (see Section 6.3.4.3) 011–111: Reserved
27–24	Register Number	4	R/W	0000	This field specifies the register within the selected block (see Table 6-12, Table 6-14, and Table 6-19).

Table 6-11. ROUTER DR Scan Chain Description (continued)

Bit	Field	Width	Type	Reset	Description
23–0	Selected Register Contents	24	–	–	Based on the values in Block Select and Register Number fields; the corresponding register is mapped to this field.

During the Capture DR state, the Data Shift register is inspected. The register specified by the Block and Register fields is read and the value is placed in the lower 24 bits of the Data Shift register.

Note

The current contents of the Data Shift register were those loaded by the previous scan.

The register specified in DR scan n-1 is read during scan n. Of course, if an intervening IR scan occurs, the contents of the Data Shift register are unpredictable, so a read of the register indicated in DR scan n-1 does not occur.

Sometimes an action on the destination register is still pending when the Update DR state is reached. Some of the bits of the destination register may not be changed while the action is pending, such as the reset controls signals have been written but not acted upon yet. Therefore, the new value indicated by this write may not be applied to the register. If this happens, the write to the ICEPick register is suppressed and the write-failure flag is set to 1. The write-failure bit is captured into the Data Shift register at bit 31. When the value is captured, the WF flag is cleared.

If bit 31 indicates that a read must be performed, the ICEPick register specified is not touched at this point. The ICEPick register contents remain undisturbed.

If the contents of the Data Shift register remain constant until the next Capture DR state, then the specified register is read at that point. An intervening IR scan disturbs the Data Shift register contents and as a consequence, it cannot be assured that the register specified will be read.

There is no address buffering within the ICEPick for the read block and register other than the Data Shift register. No extra storage is needed when the proper scan sequence is followed. For the sequence, see Section 6.5.

6.3.4 TAP Routing Registers

This section describes the TAP routing registers that can be accessed using router scan.

6.3.4.1 ICEPick Control Block

The ICEPick Control Block implements the Table 6-12. Reads of unused registers return all 0s.

Table 6-12. Control Block Registers

Register	Register Name
0x0	All0s
0x1	Control
0x2	Linking Mode
0x3 to 0xF	Reserved

6.3.4.1.1 All0s Register

This register is a dummy register that returns 0 when read. Writes are ignored. There are not any side effects to writing or reading this register.

Table 6-13. All0s Register

Bit	Field	Width	Type	Reset	Description
23–0	Zero	24	R	0	Read zero

6.3.4.1.2 ICEPick Control Register

Table 6-14. ICEPick Control Register

Bit	Field	Width	Type	Reset	Description
23–8	Reserved	16	R/W	0	Reserved
7	KeepPoweredinTLR	1	R/W	0	When 1, the JTAG power domain stays on even in the Test Logic Reset (TLR) state. When 0, the JTAG power domain will be powered down in the Test Logic Reset (TLR) state if ICEPick is visible and TMS is 1.
6	BlockSysReset	1	R/W	0	When 1, the device system reset signal is blocked.
5–1	Reserved	5	R/W	0	Reserved
0	SystemReset	1	R/W	0	Debug probe controlled System Reset This signal provides the scan controller with the ability to assert the system warm reset when a 1 is written to this field. Warm reset is automatically converted to pin reset within the hardware. This would reset even the ICEPICK logic. Writing a 0 has no effect.

6.3.4.1.3 Linking Mode Register

Table 6-15. ICEPick Linking Mode Register

Bit	Field	Width	Type	Reset	Description
23–4	Reserved	20	R/W	0x0	Reserved
3–1	TAPLinkMode	3	R/W	000	See Table 6-16
0	ActivateMode	1	R/W	0	When a 1 is written to this bit, the currently selected TAPLinkMode is activated. ICEPick links the TAPs according to these settings when the ICEPick TAP is advanced to Run-Test/Idle with any opcode in the IR.

Table 6-16. ICEPick TAP Link Mode

Value	Mode	Behavior
000	Always-first	ICEPick TAP always exists and is linked as the TAP closest to TDI.
011	Disappear-forever	When activated, the ICEPick TAP is no longer visible between the device TDI and TDO. Only a power-on reset makes the TAP visible again.
001–010, 100–111	Reserved	Reserved

6.3.4.2 Test TAP Linking Block

The Test TAP Linking block contains the control and status registers shown in Table 6-17. These registers are used in the selection of secondary TAPs into the master scan path. Each TAP has its own Test TAP Control and Status register.

Table 6-17. Test TAP Linking Registers

Register	Register Name
0x0	Secondary Test TAP 0 register
0x1	Secondary Test TAP 1 register
0x2	Secondary Test TAP 2 register
0x3	Secondary Test TAP 3 register
0x4	Secondary Test TAP 4 register
0x5	Secondary Test TAP 5 register
0x6–0xF	Reserved

6.3.4.2.1 Secondary Test TAP Register

Table 6-18. Secondary Test TAP Register (STTR)

Bit	Field	Width	Type	Reset	Description
23–10	Reserved	14	R/W	0	Reserved
9	VisibleTAP	1	R	-	See Table 6-20.
8	SelectTAP	1	R/W	0	See Table 6-20.
7–2	Reserved	6	R	0	
1	TapAccessible	1	R	-	See Table 6-20.
0	TapPresent	1	R	-	See Table 6-20.

6.3.4.3 Debug TAP Linking Block

The Debug TAP Linking block contains the control and status registers used in the selection of secondary TAPs into the master scan path. The secondary debug tap has its own Debug TAP Control and Status register. For more details, see Table 6-19.

Table 6-19. Debug TAP Linking Registers

Register	Register Name
0x0	Secondary Debug TAP 0 register
0x1–0xF	Reserved

6.3.4.3.1 Secondary Debug TAP Register

Table 6-20 lists the secondary debug TAP register (SDTR). Table 6-21 lists the reset control.

Table 6-20. Secondary Debug TAP Register (SDTR)

Bit	Field	Width	Type	Reset	Description
23–21	Reserved	3	R/W	0	Reserved
20	InhibitSleep	1	W	0	When 0, this bit does not influence the clock and the power settings to the module. While this bit is 1, power or clock for the module of the TAP is not allowed to be turned off once it is turned on. If the target does not have power or clock when setting this bit, InhibitSleep does not change the power/clock state until the target is powered and clocked again.
			R	-	The value read does not reflect the value written until the power and clock controller has acted upon a change in the written value.
19–18	Reserved	2	R	-	Reserved
17	InReset	1	R	-	The InReset status and the ReleaseFromWIR control share the same bit. When 1, the module or modules controlled by the secondary TAP is in the reset state. When 0, the module or modules is not in reset.
	ReleaseFromWIR		W	0	The InReset status and the ReleaseFromWIR control share the same bit. When a 1 is written to this bit and the module is held in reset due to the WaitInReset bit, the module reset is released. This only occurs if WaitInReset is 1 and it is the only cause for holding the module in reset. This is a self-clearing bit. Writing a 0 has no effect.
16–14	ResetControl	3	R/W	0	Override the application controls of the functional warm reset to a module. See Table 6-21.
13–10	Reserved	4	R/W	0	Reserved
9	VisibleTAP	1	R	-	When 1, the TAP is currently selected and visible in the active scan chain. The VisibleTap bit indicates that the TAP, which was previously selected with the SelectTap bit, is now part of the device master scan path. The VisibleTap bit is set by ICEPick when the Run-Test-Idle state is reached.
8	SelectTAP	1	R/W	0	The SelectTap bit allows scan controller software to change which secondary TAPs are included in the device level master scan path. When this bit is set to 1, the TAP is selected for inclusion in the master scan path when the TAP state advances to the Run-Test-Idle state. When this bit is changed to 0, the TAP is deselected from the master scan path when the TAP state advances to the Run-Test-Idle state. Selection or deselection occurs in the Run-Test-Idle state regardless of the current IR instruction. If TapPresent is 0, writes to the SelectTap bit are blocked and the bit is held at 0.
7–4	Reserved	4	R/W	0	Reserved
3	ForceActive (ForcePowerAndClock)	1	W	-	When ForceActive is 0, the module's clock and power settings follow the normal application settings unless one of the other emulation controls is affecting the state. Setting the ForceActive bit causes the power and clock held on and to be turned on if necessary. In this sense, the ForceActive bit could be named ForcePowerAndClock. Clearing the ForceActive bit returns control of the power and clock settings to the application. If the application controls indicate that the power and clock must be off, the power and clock to the module is turned off.
			R	-	The value read does not reflect the value written until the power and clock controller has acted upon a change in the written value.

Table 6-20. Secondary Debug TAP Register (SDTR) (continued)

Bit	Field	Width	Type	Reset	Description
2	Reserved	1	R	-	Reserved
1	TapAccessible	1	R	-	When 0, the TAP cannot be accessed due to security. When 1, the TAP can be accessed.
0	TapPresent	1	R	-	When 0, there is not a TAP assigned to this spot. When 1, this TAP exists in the device. If a TAP does not exist, the rest of the controls and status bits in this register are considered to be non-operational.

Table 6-21. Reset Control

Value	Command	Description
000	Normal Operation	Reset operates under the normal control of the application or device controls.
001	Wait in reset (Extend reset)	The module or modules controlled by this secondary TAP remain in the reset state when the reset is asserted. This bit alone does not reset the processor.
010	Reserved	Reserved
011	Reserved	Reserved
1xx	Cancel	Cancels reset command lockout

6.4 ICEMelter

ICEMelter wakes up the JTAG power domain, which contains ICEPick and cJTAG modules and monitors the activities on the TCK pin. When ICEMelter detects traffic on the TCK pin (8 rising edges and 8 falling edges on TCK), it sends a power-up request to the AON_PMCTL that powers up the JTAG power domain. There is a time-out built into the device that will reset the ICEMelter if the window from the third rising edge to the eighth rising edge exceeds 4 ms. The debug probe must allow the JTAG power domain to have a proper power-up time of at least 200 µs before sending remaining commands to the JTAG interface.

TI recommends that care is taken to avoid unintentional traffic on the TCK pin. This can for example happen if the TCK pin is made accessible through a connector which is frequently connected and disconnected, or is located on a pin row that is touched during regular use. Unintentional traffic on the TCK pin can cause the ICEMelter to power up the JTAG domain, which will add approximately 400 µA to any device mode (including Standby), and also set the Halt In Boot (HIB) flag. The HIB flag will halt the device on the subsequent boot (such as after any system reset other than pin reset or POR, or when entering Shutdown). Exiting Halt In Boot, clearing the HIB flag, and disabling the JTAG domain can only be done by a pin reset, a POR, or by using the JTAG interface itself.

The TCK pin has an internal pullup designed to avoid unintentional traffic due to noise, but it will not be sufficient if there is risk of external activity on the TCK pin caused by unintentional touching or shorting of the pin. If it is not possible to prevent such activity from happening, it is recommended that a strong external pullup is used, or even shorting the pin to the supply voltage through a zero-ohm resistor. The size of the pullup or whether the pin is shorted to the supply voltage, will be a tradeoff between robustness against unintentional external activity and the need for an accessible debug port. If the TCK pin is disabled by shorting it to the supply voltage, Flash programming must be done using the bootloader.

It is possible to disable the input driver of the TCK pin by using AON_IOC:TCKCTL. Disabling the TCK pin impairs the debug sessions, so special care is required when disabling the TCK pin if the support for debugging the code is desired. Debugging a running target will not be possible if the TCK pin is disabled.

6.5 Serial Wire Viewer (SWV)

The CPU uses the TPIU macro inside the processor to support the serial wire viewer (SWV) interface (a single-line interface). Use the following sequence to enable SWV output on the CPU:

1. Enable the trace system (CPU_DCB:DEMCR.TRCENA)
2. Enable ITM (CPU_ITM:TCR.ITMENA)
3. Enable the desired stimulus port (0 to 31) (CPU_ITM:TER0)
4. Change formatter configuration, if needed (CPU_TPIU:FFCR)
5. Change the pin protocol, if needed (CPU_TPIU:SPPR)
6. Set the baud rate (CPU_TPIU:ACPR)
7. The SWV can be mapped to DIOn by writing the corresponding port ID in the IOC:IOCFGn . For more details, see Chapter 15.
8. Set which stimulus ports can be accessed by unprivileged code (CPU_ITM:TPR)

Writes to the CPU_ITM:STIMn registers (assuming that they are enabled) trigger a transmit on SWV output if the FIFO is not full.

6.6 Halt In Boot (HIB)

The CC13x4x10 and CC26x4x10 device platform implements a mechanism to ensure that the external debug probe can take control of the device before it executes any application code. This mechanism is called halt in boot (HIB). When HIB detects debug activity, the boot code stops in a wait for interrupt instruction (WFI) at the end of its execution before jumping to the application code in flash.

Detection of activities on the TCK pin (which powers up the JTAG power domain) is the condition for HIB when next boot occurs. If JTAG power domain is turned off by entering the test logic reset (TLR) state before a system reset occurs, the HIB conditions can be cleared. The HIB conditions are not cleared if JTAG power domain turns off after AON_PMCTL:SHUTDOWN.EN is written to 1.

To exit HIB, the external debug probe must connect to the device and first HALT, then RESUME the CPU through DAP. Halting the CPU is a debug event that wakes the CPU from the WFI instruction. After resuming, the program execution continues from the application code.

6.7 Debug and Shutdown

The debugger cannot stay connected in shutdown mode because the power source for debug subsystem turns off in this mode. This means that entering shutdown causes abrupt disconnection from the debug probe. To facilitate debugging of the shutdown scenarios, the CC13x4x10 and CC26x4x10 device platform has the following considerations:

- If a device is in shutdown mode, activity on TCK causes immediate wake up.
- If conditions for HIB are met while entering shutdown mode, the device wakes up as soon as it reaches the shutdown state. If the conditions for HIB are met during the boot which happens after wakeup, the boot code enters a loop until an I/O wakeup event occurs.

6.8 Boundary Scan

The CC13x4x10 and CC26x4x10 devices use standard boundary scan as defined under the IEEE 1149.1 JTAG standard. Each DIO has a dedicated boundary scan cell that contains six registers (two each for output pins, bidirectional control pins, and input pins), as shown in Figure 6-11.

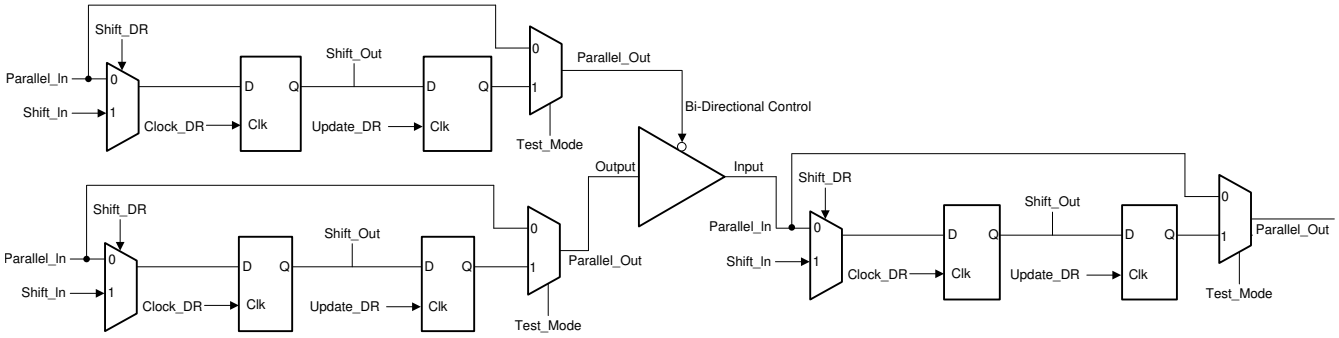


Figure 6-11. Boundary Scan Cell

Of the six registers, three are shift registers and the other three are update registers. The shift registers are in a scan chain and connect across all the DIOs. There are few muxes inside each Boundary Scan Register cell that select between the test data and the functional data. The boundary scan implementation is intended to cover both DC parametric tests as well as IODFT testing. The 48 DIOs available in the CC13x4x10 and CC26x4x10 devices are categorized under different groups that are based on test pin muxing and tester-contacted and non-contacted I/Os. For further information regarding boundary scan I/O across different packages, see Table 6-22.

Table 6-22. Boundary Scan I/O for Different Devices

Pin Name	Tap Interface	Group
TCK	TAP_SCAN_CLOCK	
TMS	TAP_SCAN_MODE	
DIO0 - DIO15		grp1
DIO16	TAP_SCAN_OUT	grp3
DIO17	TAP_SCAN_IN	grp3
DIO18- DIO22		grp1
DIO23 - DIO24		grp2
DIO25 - DIO47		grp1

This page intentionally left blank.

Chapter 7

Power, Reset, and Clock Management (PRCM)

This chapter details the flexible power management and clock control (PRCM) of the CC13x4x10 and CC26x4x10 device platform.

7.1 Introduction 480
7.2 System CPU Mode 481
7.3 Supply System 481
7.4 Digital Power Partitioning 482
7.5 Clock Management 484
7.6 Power Modes 490
7.7 Reset 494
7.8 PRCM Registers 495

7.1 Introduction

Power and clock management (PRCM) in the CC13x4x10 and CC26x4x10 device platform is highly flexible to facilitate low-power applications. The following sections describe details for clock and power control in addition to covering reset features.

The features in this chapter are embedded and optimized in TI's Power Manager. Please see the SimpleLink SDK Power Management: MSP432, MSP432E4, CC13xx/CC26xx, and CC32xx User's Guide for more details.

Figure 7-1 shows the hierarchy of power-saving features in the CC13x4x10 and CC26x4x10 device platform. Low-power consumption and cycling time for a power-saving mode has an inverse relationship. The power-saving mode with the lowest power consumption requires the longest time from initiation to power-saving mode, as well as wake-up time back to active mode. Table 7-1 summarizes the power-saving features.

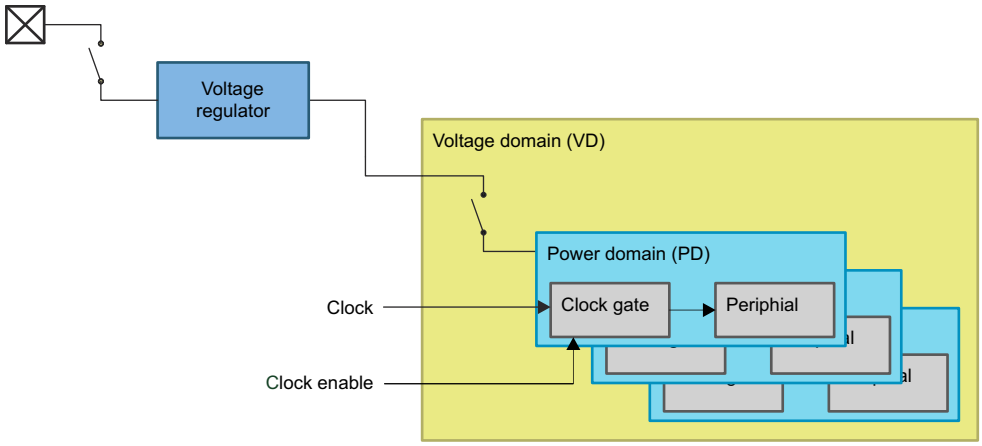


Figure 7-1. Hierarchy of Power Saving Features

Table 7-1. Power Saving Features

Power-Saving Feature	Description
Clock gating	Immediate response - no latency. This feature offers the least amount of power saved
Power domain off (overrides clock gating)	Power cycling down and up takes longer time than clock gating. Modules in power domains without retention must be reinitialized before functionality can be resumed.
Voltage regulator off	Power cycling down and up takes longer time than power domain cycling. The CC13x4x10 and CC26x4x10 device platform loses all configurations and will boot at wake up. This feature offers the least possible current consumption.

Table 7-2 lists the four defined power modes for the power-saving features in TI's Power Manager listed in Table 7-1. Section 7.6 discusses the power modes in detail.

Table 7-2. Power Modes in TI's Power Manager

Power Mode	Description
Active mode	The system CPU is running.
Idle mode	The CPU power domain is powered off.
Standby mode	All power domains are powered off, and the AUX domain is in low-power or power-down mode. The voltage domains are supplied by the micro LDO.
Shutdown mode	Only I/Os maintain their state. All voltage regulators, voltage domains, and power domains are off.

7.2 System CPU Mode

Section 7.3 refers to the system CPU mode, so it is important to understand what this means.

The system CPU has three different operation modes: run, sleep, and deep sleep (see Table 7-3). Each mode is used to gate internal clocks in the system CPU, in addition to peripheral clocks that may be gated in accordance to the current system CPU mode. Deep sleep mode is, in some cases, one of several requirements for powering down voltage and power domains.

Table 7-3. System CPU Modes

System CPU Mode	Description
Run mode	Wait For Interrupt (WFI) and Wait For Event (WFE) both inactive, CPU_SCB:SCR.SLEEPDEEP is don't care
Sleep mode	WFI or WFE active and CPU_SCB:SCR.SLEEPDEEP = 0
Deep sleep mode	WFI or WFE active and CPU_SCB:SCR.SLEEPDEEP = 1

7.3 Supply System

The supply system of the CC13x4x10 and CC26x4x10 device platform is complex and controlled by hardware. Figure 7-2 shows a simplified scheme with focus on parts that can be controlled by software. Registers that affect the different power domains are highlighted in Figure 7-2.

See Figure 7-3 for details about voltage and power domains.

7.3.1 Internal DC/DC Converter and Global LDO

Normally, the VDDS supply pins of the CC13x4x10 and CC26x4x10 device platform are powered from a 1.8 V to 3.8 V supply (for example, batteries), and the VDDR supply pins are powered from the internal DC/DC regulator.

Alternatively, the internal global LDO can be used instead of the DC/DC regulator, but this increases the current consumption of the device. In this mode, disconnect DCDC_SW and connect VDDS_DCDC to the VDDS supply. The Global LDO is connected internally to the VDDR pin, which must be connected externally to the VDDR_RF pin. The Global LDO must be decoupled by a µF-sized capacitor on the VDDR net.

7.3.2 External Regulator Mode

Some CC13x4x10 and CC26x4x10 devices have an option to be supplied by an external regulator with a voltage range of 1.65 V to 1.95 V. In this mode, the VDDS and VDDR pins are tied together. To enable external regulator mode, the VDDS_DCDC pin and the DCDC_SW pins must be connected to ground, which effectively disables both the internal Global LDO and the internal DC/DC regulator.

7.4 Digital Power Partitioning

The CC13x4x10 and CC26x4x10 device platform has two voltage domains, MCU_VD and AON_VD. Both voltage domains contain multiple power domains, *_PD. Each power domain contains digital modules. Figure 7-3 shows details of the power partitioning in the CC13x4x10 and CC26x4x10 device platform.

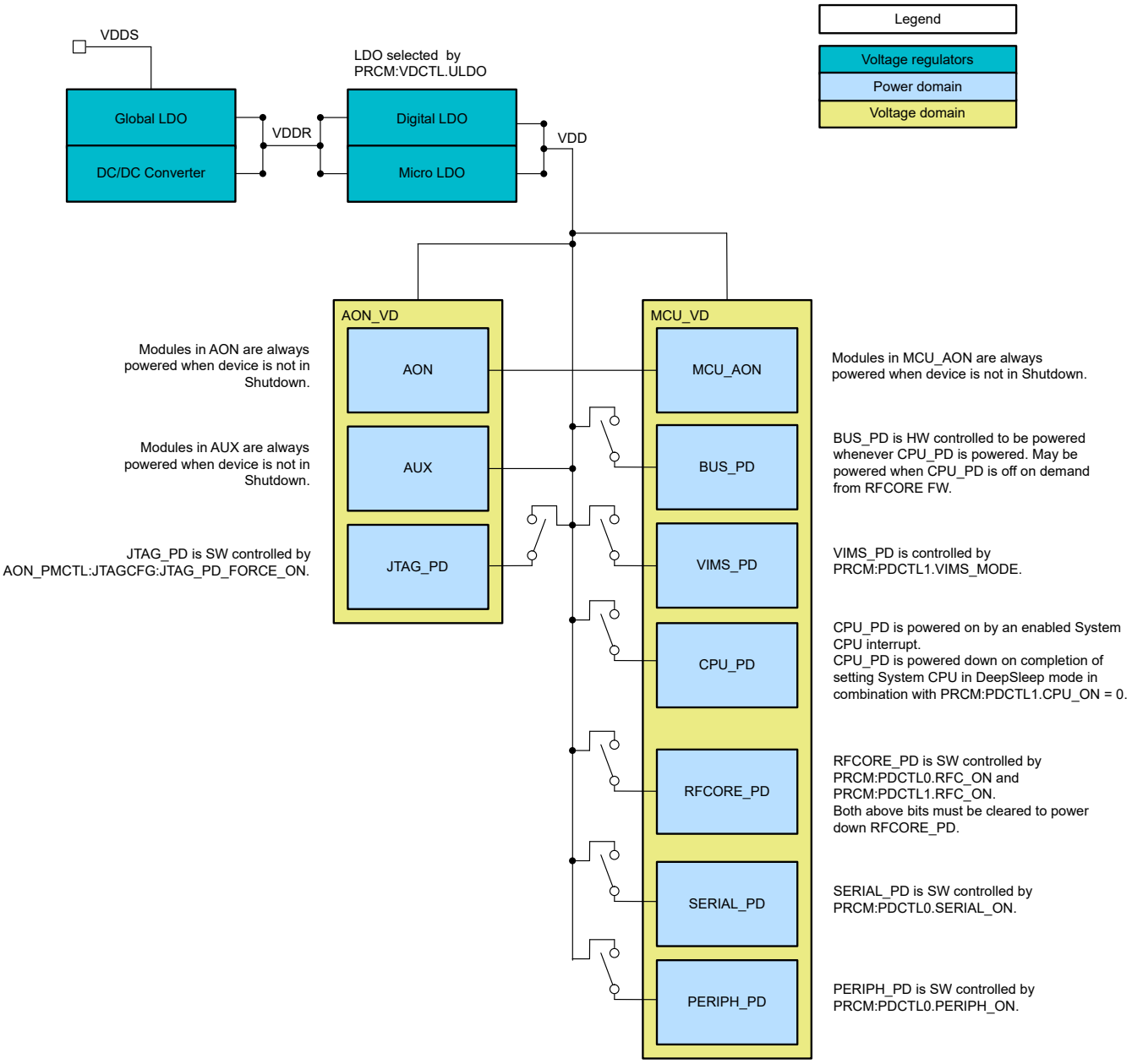


Figure 7-2. CC13x4x10 and CC26x4x10 Supply System

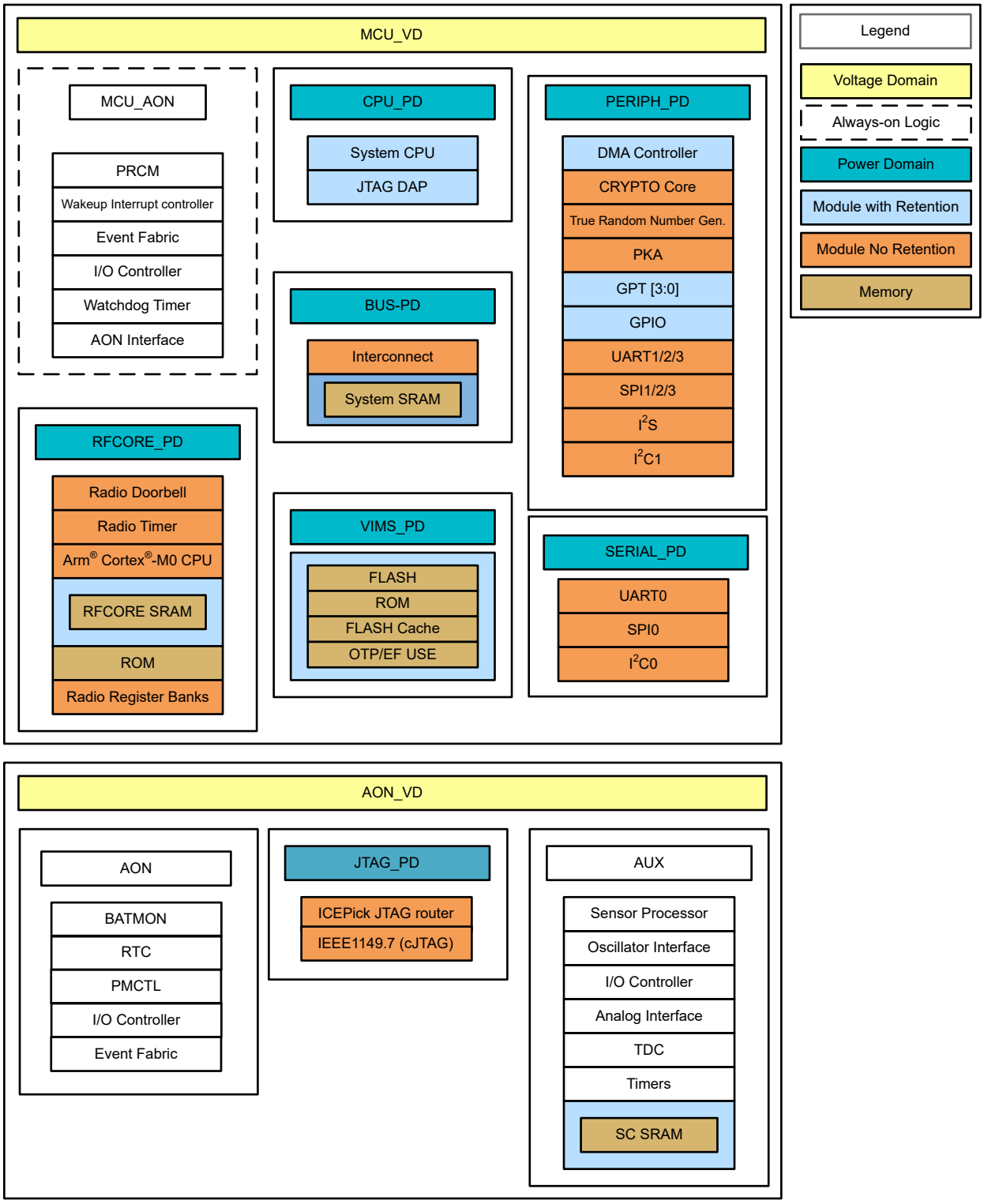


Figure 7-3. Digital Power Partitioning in CC13x4x10 and CC26x4x10

7.4.1 MCU_VD

Figure 7-3 shows that the MCU voltage domain contains the CPU system divided into multiple power domains. MCU_VD also includes always-on logic not encapsulated in a power domain, which is powered whenever the voltage regulator is on. The voltage regulator will be on and MCU_AON will be powered when the CC13x4x10 and CC26x4x10 device platform is not in shutdown or reset mode. Figure 7-3 shows this logic as MCU_AON.

7.4.1.1 MCU_VD Power Domains

Figure 7-2 shows control of MCU_VD power domains and provides descriptions of the registers.

7.4.2 AON_VD

AON_VD contains one power domain and always-on logic marked AON in Figure 7-3.

The AON voltage domain contains one power domain, JTAG_PD. All other domains are always-on logic not encapsulated in a power domian, which is powered whenever the voltage regulator is on. The voltage regulator will be on when the the CC13x4x10 and CC26x4x10 devices are not in shutdown or reset mode.

7.4.2.1 AON_VD Power Domains

Figure 7-2 shows control of AON_VD power domains and provides descriptions of the registers.

7.5 Clock Management

7.5.1 System Clocks

Figure 7-4 and Table 7-4 show that the CC13x4x10 and CC26x4x10 device platform has a flexible clock mux where system clocks can be derived from several sources.

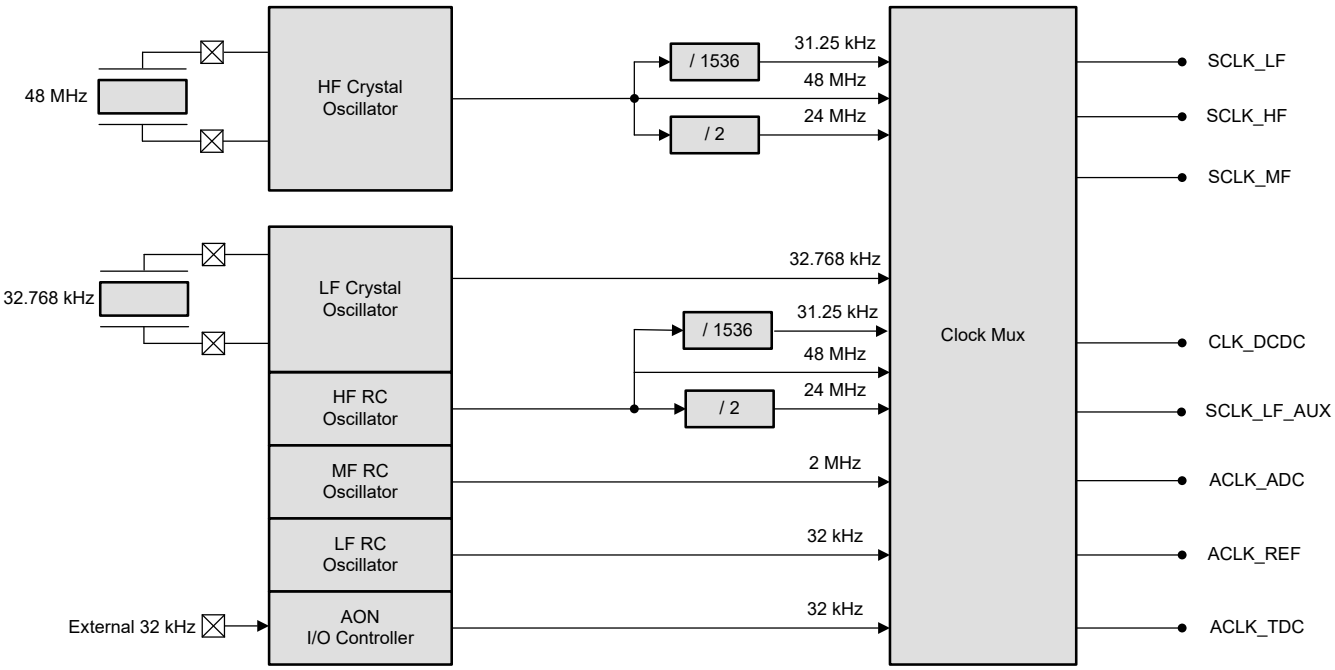


Figure 7-4. Clock Sources

7.5.1.1 Controlling the Oscillators

Table 7-4 lists the system clock descriptions and possible sources and Figure 7-5 shows the system clock muxing.

Table 7-4. System Clocks

Clock	Description	Possible sources
SCLK_LF	Low frequency clock • Always used by AON • Available for AUX in Active and Standby mode	• 31.25 kHz derived from 48-MHz RC oscillator • 31.25 kHz derived from 48-MHz crystal oscillator • 32-kHz RC oscillator • 32.768-kHz crystal oscillator Selectable in DDI_0_OSC:CTL0.SCLK_LF_SRC_SEL
SCLK_MF	Medium frequency clock • Always used by PMCTL in Active and Idle mode • Available for AUX in Active, Idle, and Standby modes	2-MHz RC oscillator
SCLK_HF	High frequency clock • Used by MCU_VD in Active and Idle modes • Available for AUX in Active and Idle mode	• 48 MHz derived from 48-MHz RC oscillator • 48 MHz derived from 48-MHz crystal oscillator Selectable in DDI_0_OSC:CTL0.SCLK_HF_SRC_SEL
SCLK_LF_AUX	Used for low-power comparator in AUX_PD (COMP_B) and as clock to the recharge comparator in REFSYS	Same as SCLK_LF
ACLK_ADC	Used as clock source for ADC	Same as SCLK_HF/2
ACLK_REF	Used as start and stop source for time-to-digital converter (TDC)	• 31.25 kHz derived from 48-MHz RC oscillator • 31.25 kHz derived from 48-MHz crystal oscillator • 32-kHz RC oscillator • 32.768-kHz crystal oscillator • 2-MHz RC oscillator Selectable in DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL
ACLK_TDC	Used as clock for TDC	• 48 MHz derived from 48-MHz RC oscillator • 24 MHz derived from 48-MHz RC oscillator • 24 MHz derived from 48-MHz crystal oscillator Selectable in DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL
CLK_DCDC	High frequency clock • Used as source for DCDC	• 48 MHz derived from 48-MHz RC oscillator • 48 MHz derived from 48-MHz crystal oscillator Selectable in DDI_0_OSC:CTL0.CLK_DCDC_SRC_SEL

Note

When the 48-MHz crystal oscillator is enabled (by selecting XOSCHF as source for SCLK_HF), the XOSCHF must not be turned off, or SCLK_HF source must not be changed to another source, before the XOSCHF is reported as stable and switched to. The XOSCHF is stable when the DDI_0_OSC:STAT0.PENDINGSCLKHFSWITCHING is asserted after starting the crystal. The TI Power Manager switches the SCLK_HF source in a correct manner.

CAUTION

If 31.25 kHz derived from the 48-MHz crystal oscillator is selected as the SCLK_LF source, the device must not be allowed to enter Standby mode. If the device goes to Standby with the 48-MHz crystal oscillator running, the oscillator can stop, putting the device in an unresponsive state. If the DC/DC regulator is also used, stopping the 48-MHz regulator can lead to permanent damage of the device.

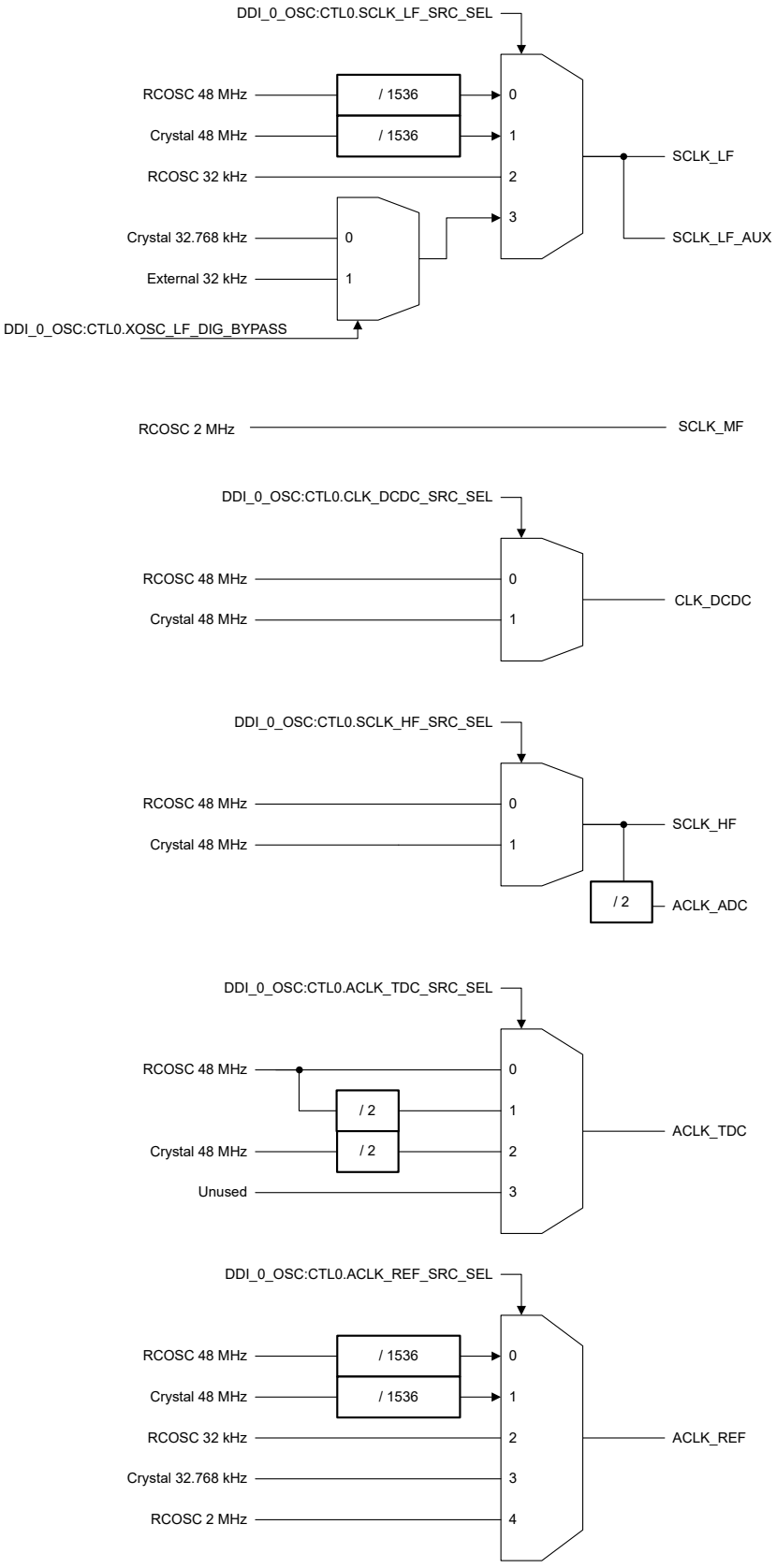


Figure 7-5. System Clock Muxing

7.5.2 Clocks in MCU_VD

AON_PMCTL supports MCU_VD with a clock that is divided and gated by PRCM before being distributed to all modules in MCU_VD. Figure 7-6 shows the registers in PRCM that define division and gate control for all module clocks. When no BUS transactions can occur, hardware automatically gates the SYSBUS clock. The following conditions must be true to gate the SYSBUS:

- System CPU in deep sleep mode
- PRCM:SECDMACLKGDS.DMA_CLK_EN = 0
- PRCM:SECDMACLKGR.DMA_AM_CLK_EN = 0
- PRCM:SECDMACLKGDS.CRYPTO_CLK_EN = 0
- PRCM:SECDMACLKGR.CRYPTO_AM_CLK_EN = 0
- PRCM:I2SCLKGDS.CLK_EN = 0
- PRCM:I2SCLKGR.AM_CLK_EN = 0
- RFCORE FW does not require bus access

The SYSBUS clock may run even when the system CPU is in deep sleep mode when either DMA, CRYPTO, I^2S, or RFCORE requires an active interconnect.

MCU_AON has two clocks, an INFRASTRUCTURE clock that always runs and a PERBUSULL clock that is identical to the INFRASTRUCTURE clock whenever the SYSBUS clock is running. When the SYSBUS clock is gated, the PERBUSULL clock is automatically gated. INFRASTRUCTURE and PERBUSULL clocks are automatically controlled to run at a maximum of half the clock frequency of SCLK_HF, regardless of the settings in PRCM:INFCLKDIVR.RATIO, PRCM:INFCLKDIVS.RATIO, or PRCM:INFCLKDIVDS.RATIO.

7.5.2.1 Clock Gating

As seen in Figure 7-6, the peripheral modules have conditional clock gates that depend on the system CPU mode. The clock of a module may be enabled or disabled when the system CPU mode changes.

Example:

- PRCM:I2CCLKGR.CLK_EN = 1
- PRCM:I2CCLKGS.CLK_EN = 0
- PRCM:I2CCLKGDS.CLK_EN = 1

These settings result in the I^2C clock running when the system CPU is in run mode and deep sleep mode, while the I^2C clock is disabled, and when system CPU is in sleep mode.

Note

When set in deep sleep mode, the system CPU remains in sleep mode for a few clock cycles during the transition. An application that requires a continuous module clock enables all clock-gate registers for the module during the transition while the system CPU changes modes.

Clock control can be controlled independently of system CPU mode by use of the modules clock control for all modes, AM_CLK_EN.

Example:

PRCM:I2CCLKGR.AM_CLK_EN = 1

This setting results in the I^2C clock running independently of system CPU mode.

Clocks in MCU_VD are hardware controlled during power cycling, so it is not required to control the module clocks when power domains are powered up or down.

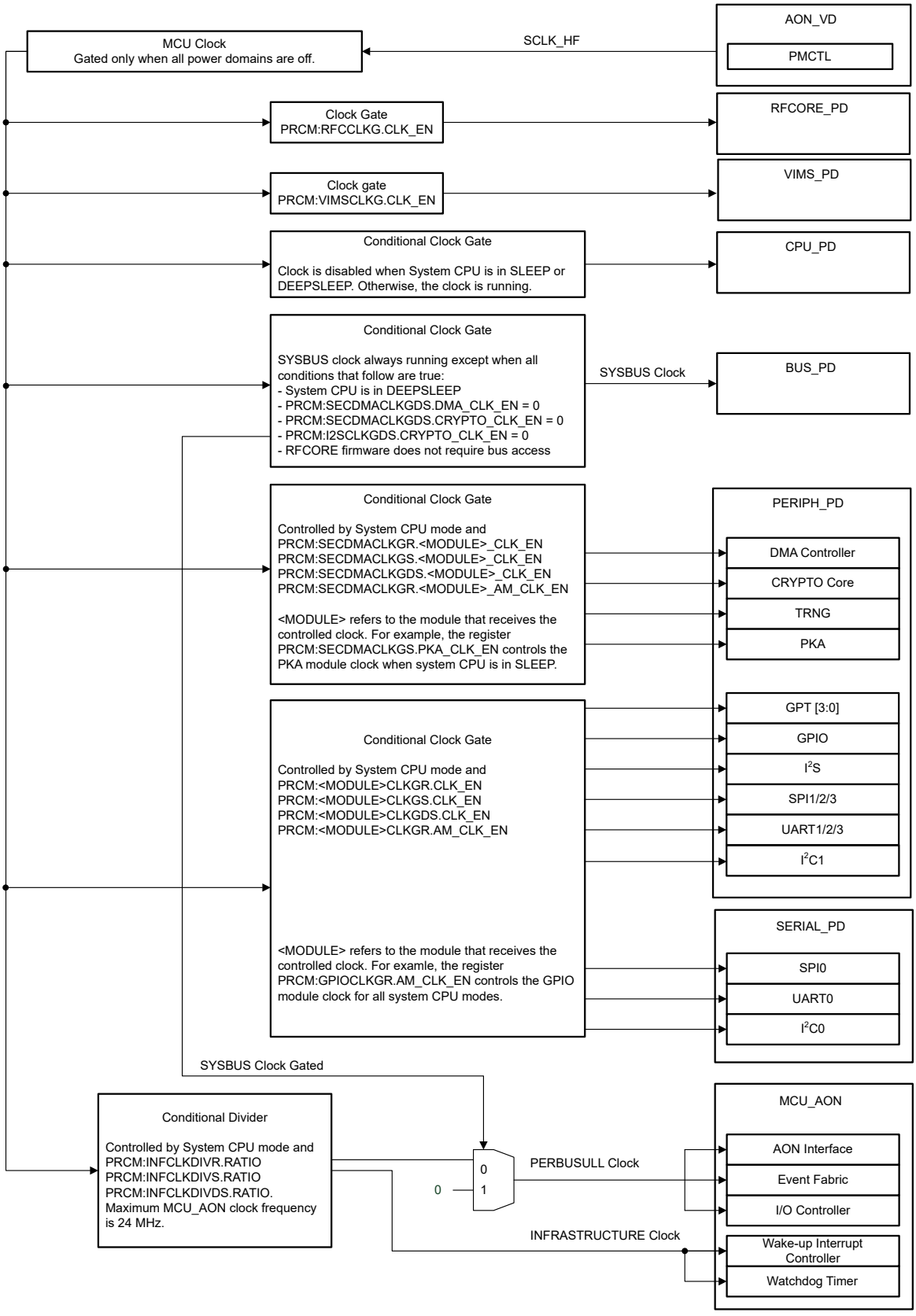


Figure 7-6. Clocks in MCU_VD

7.5.2.2 Scaler to GPTs

A scaler to GPTs is available to enable GPTs to count at a slower frequency than SYSBUS clock. The setting in the PRCM:GPTCLKDIV register is valid for all GPTs in the system.

7.5.2.3 Scaler to WDT

There is a scaler with a fixed-division ratio of 32 of the MCU clock that is present. Regardless of the settings in the PRCM:INFRCLKDIVR, the PRCM:INFRCLKDIVS, and the PRCM:INFRCLKDIVDS registers, the watchdog counts at a constant speed.

7.5.3 Clocks in AON_VD

All modules in AON_VD run on SCLK_LF and SCLK_MF except AUX. Clocks to AUX are user-configurable.

7.6 Power Modes

The flexibility of the power management of the CC13x4x10 and CC26x4x10 device platform allows many different configurations to achieve a low-power application. This section describes the power modes, as defined by TI's Power Manager, which cover a range of power-saving modes (from low-power savings with fast-cycling time to high-power savings with long-cycling time).

Table 7-5 provides an overview of the power modes defined in TI'sPower Manager.

Table 7-5. Power Modes as Defined in TI's Power Manager

Mode	Software Configurable Power Modes				Reset Pin Held
	Active	Idle	Standby	Shutdown	
System CPU	Active	Off	Off	Off	Off
System SRAM	On	On	Retained	Off	Off
Register retention(1)	Full	Full	Partial	No	No
VIMS_PD (Flash)	On	Available	Off	Off	Off
RFCORE_PD (radio)	Available	Available	Off	Off	Off
SERIAL_PD	Available	Available	Off	Off	Off
PERIPH_PD	Available	Available	Off	Off	Off
Sensor controller	Available	Available	Available	Off	Off
Supply system	On	On	Duty-cycled	Off	Off
High-speed clock	XOSC_HF or RCOSC_HF	XOSC_HF or RCOSC_HF	Off	Off	Off
Medium-speed clock	RCOSC_MF	RCOSC_MF	RCOSC_MF available	Off	Off
Low-speed clock	XOSC_LF or RCOSC_LF	XOSC_LF or RCOSC_LF	XOSC_LF or RCOSC_LF	Off	Off
Wakeup on RTC	Available	Available	Available	Off	Off
Wakeup on pin edge	Available	Available	Available	Available	Off
Wakeup on reset pin	Available	Available	Available	Available	Available
Brown Out Detect (BOD)	Active	Active	Partial(2)	Off	N/A
Power On Reset (POR)	Active	Active	Active	Active	N/A
Watchdog (WDT)	Available	Available	Off	Off	Off
AON_WDT	Available	Available	Available	Off	Off

(1) See Figure 7-3 for modules with retention.
(2) Brown Out Detector is disabled between recharge periods in Standby.

7.6.1 Start-Up State

The state of the CC13x4x10 and CC26x4x10 device platform after a system reset, power on, or wake up from shutdown is as follows:

- Global LDO is active
- Digital LDO is active
- AON_VD is powered
  - AON is powered
  - JTAG_PD is powered off (default is on, but it is turned off by BOOT if no debugger is connected)
- MCU_VD is powered.
  - MCU_AON is powered
  - CPU_PD is powered
    - System CPU is in run mode
  - BUS_PD is powered
    - SYSBUS is clock running
  - VIMS_PD is powered
    - VIMS is clock running
  - All other power domains are off
  - All digital module clocks are disabled

7.6.2 Active Mode

Active mode is defined as any possible chip state where CPU_PD is powered, including BUS_PD and VIMS_PD (see Figure 7-2).

In active mode, all modules are available and power consumption is highly application dependent.
Power saving features are:

- Enable the DC/DC converter
- Power only the necessary power domains
- Enable only the necessary module clocks

Note

Wake-up time for a power domain in the CC13x4x10 and CC26x4x10 device platform requires approximately 11 µs. Because clock gating in the CC13x4x10 and CC26x4x10 device platform is efficient, it can be more power efficient to disable all the clocks in a power domain and leave the domain powered than to power cycle it frequently.

7.6.3 Idle Mode

Idle mode is defined as any chip state where CPU_PD is powered off while any other module can be powered. In Idle mode, all modules are available and power consumption is highly application dependent.

The CC13x4x10 and CC26x4x10 device platform is put in Idle mode with the following requirements:

- PRCM:PDCTL1.CPU_ON = 0
- CPU_SCS:SCR.SLEEPDEEP = 1
- WFI or WFE active

Note

When CPU_SCS:SCR.SLEEPDEEPS = 1, CPU_SCS:SCR.SLEEPDEEP is only accessible from the Secure state.

In Idle mode, the CC13x4x10 and CC26x4x10 device platform can wake up from any enabled wakeup source.

7.6.4 Standby Mode

Standby mode is defined as all power domains in the MCU_VD being powered off and the micro LDO supplying AON_VD and MCU_AON (see Figure 7-2). Standby is the lowest power mode where the CC13x4x10 and CC26x4x10 device platform still has functionality other than maintaining I/O output pins and detecting input state changes (see
Table 7-6). The Always On Watchdog Timer (AON_WDT) can be used in this mode.

All parts in MCU_AON are retained in standby mode. All modules in MCU_VD with retention, as shown in Figure 7-3, are retained in standby mode. All other logic in MCU_VD must be reconfigured after wake up from standby mode.

Possible wake-up sources are events from I/O, JTAG, RTC, BATMON, and the sensor controller.

The following are prerequisites for the CC13x4x10 and CC26x4x10 device platform to enter standby mode:

- AUX is in low power or power down mode.
- JTAG_PD is powered off.
- The SCLK_LF clock is derived from one of the following clock sources:
  - 32-kHz RC oscillator
  - 32.768-kHz crystal oscillator
  - External 32-kHz clock
- Request micro LDO to supply digital parts (see Figure 7-2).

Table 7-6 shows the steps for taking the CC13x4x10 and CC26x4x10 device into Standby. TI recommends using the TI Power Manager to enter Standby.

Table 7-6. Example Sequence for Setting CC13x4x10 and CC26x4x10 in Standby Mode

Description	Register	Required Step
Configure recharge configuration	AON_PMCTL:RECHARGECFG	Yes
Freeze the IOs on the boundary between MCU and AON	AON_IOC:IOCLATCH.EN	Yes
Make sure that any possible outstanding AON writes are complete before turning off the power domains	AON_RTC:SYNC.WBUSY	Yes (Read register)
Turn off power domains and verify they are turned off	PRCM:PDCTL0RFC PRCM:PDCTL0SERIAL PRCM:PDCTL0PERIPH PRCM:PDCTL1CPU	Yes
Make sure that no clocks are forced on in any modes for CRYPTO, µDMA and I^2S	PRCM:SECDMACLKGR PRCM: I2SCLKGR	Yes
Gate running deep sleep clocks for CRYPTO, µDMA and I^2S	PRCM:SECDMACLKGDS.CRYPTO_CLK_EN PRCM:SECDMACLKGDS.DMA_CLK_EN PRCM:I2SCLKGDS	Yes
Load the new clock settings	PRCM:CLKLOADCTL	Yes
Configure the VIMS power domain mode	PRCM:PDCTL1VIMS	Yes
Request digital supply to be micro LDO	PRCM:VDCTL	Yes
Make sure that all writes have taken effect	AON_RTC:SYNC.WBUSY	Yes
Make sure that UDMA, CRYPTO and I^2C clocks are turned off	PRCM:CLKLOADCTL.LOAD_DONE	Yes
Make sure that power domains have been turned off	PRCM:PDSTAT0 PRCM:PDSTAT0RFC PRCM:PDSTAT0SERIAL PRCM:PDSTAT0PERIPH	Yes

Table 7-6. Example Sequence for Setting CC13x4x10 and CC26x4x10 in Standby Mode (continued)

Description	Register	Required Step
Turn off cache retention if requested	PRCM:RAMRETEN	No (default: retention enabled)
Set the system CPU SLEEPDEEP bit	CPU_SCS:SCR.SLEEPDEEP	Yes
Stop the system CPU to start the power-down sequence	WFI or WFE	Yes

7.6.5 Shutdown Mode

Shutdown mode is defined as having no active power regulator in the CC13x4x10 and CC26x4x10 device platform. Do not use shutdown as a power mode to conserve power during normal operation of the device.

Before putting the CC13x4x10 and CC26x4x10 device platform in shutdown mode, I/O pins are latched to keep their output values in shutdown—this is the only difference between holding the devices in reset with the reset pin and shutdown mode.

Only an enabled pin interrupt or reset pin can wake up the CC13x4x10 and CC26x4x10 device platform from shutdown mode.

Note

A wake-up event to wake up from shutdown is not detected until the device reaches shutdown. Wake-up events happening after a shutdown is initiated but before actual shutdown are not captured and thus do not cause the device to wake up.

Table 7-7 shows the steps for taking the CC13x4x10 and CC26x4x10 device into Shutdown. TI recommends using the TI Power Manager for this process.

Table 7-7. Example Sequence for Going to Shutdown

Description	Register	Required Step
Configure the wake-up pin	IOC:IOCFGxx.WU_CFG	Yes
Enable pad sleep to latch device outputs before Shutdown	AON_PMCTL:SLEEPCTL	Yes
Synchronize transactions to AON domain	AON_RTC.SYNC	Yes (Read register)
Enable shutdown and latch I/Os	AON_PMCTL:SHUTDOWN.EN	Yes
Stop the system CPU to start the power-down sequence	WFI or WFE	Yes

7.7 Reset

The CC13x4x10 and CC26x4x10 device platform has several sources of reset; some are triggered due to errors or unexpected behavior, while others are user initiated.

All resets cause the entire chip to be reset.

7.7.1 System Resets

A reset resulting in a complete power-up sequence and system CPU boot sequence is defined as a *system reset*. The AON_PMCTL:RESETCTL.RESET_SRC register is readable and always shows the last source of a reset resulting in a system reset.

The following resets cannot be disabled and, when triggered, always result in a system reset:

- Power-on reset
- Pin reset
- VDDS failure
- VDDR failure

7.7.1.1 Clock Loss Detection

When the clock loss feature is enabled with the DDI_0_OSC:CTL0.CLK_LOSS_EN and the AON_PMCTL:RESETCTL.CLK_LOSS_EN registers, a detected loss of SCLK_LF, SCLK_MF, or SCLK_HF results in a system reset. After recovery, the AON_PMCTL:RESETCTL.RESET_SRC register shows clock loss as the source of reset.

Note

The application must set both DDI_0_OSC:CTL0.CLK_LOSS_EN and the AON_PMCTL:RESETCTL.CLK_LOSS_EN in order to enable Clock Loss Detection, it is not enabled after boot.

7.7.1.2 Software-Initiated System Reset

Writing to the AON_PMCTL:RESETCTL.SYSRESET register results in a system reset. After recovery, the AON_PMCTL:RESETCTL.RESET_SRC register shows SYSRESET as the source of reset.

7.7.1.3 Warm Reset Converted to System Reset

Reset requests from the MCU system is by default set to result in a system reset when any warm reset source is triggered. After recovery, the AON_PMCTL:RESETCTL.RESET_SRC register shows WARMRESET as the source of reset.

7.7.2 Reset of the MCU_VD Power Domains and Modules

Reset of logic in power domains are hardware controlled. A module without retention is reset when the encapsulating power domain is power cycled. A module with retention resets when MCU_VD is power cycled or reset.

7.7.3 Reset of AON_VD

AON_VD is reset by a system reset. For details, see Section 7.7.1.

7.7.4 Always On Watchdog Timer (AON_WDT)

The Always On Watchdog Timer (AON_WDT) is used to regain control when the system has failed due to a software error or to the failure of an external device to respond in the expected way specifically during standby. This WDT generates a reset when a time-out value is reached. Writing to AON_PMCTL:WDTLOAD starts the counter which starts counting down from the written value on every SCLK_LF tick.

The device has another watchdog timer outside the AON domain. See Chapter 19 for details.

If the AON_PMCTL:WDTLOAD register is written with a new value while the WDT counter is counting, then the counter is loaded with the new value and continues counting. If configured, using AON_PMCTL:WDTTEST, the counter can stall when the microcontroller asserts the CPU Halt flag during debug.

To prevent the WDT configuration from being inadvertently altered by software, the write access to the watchdog registers can be locked by writing the AON_PMCTL:WDTLOCK register to any value. To unlock the WDT, write the AON_PMCTL:WDTLOCK register to the value 0x1ACCE551. The WDT is running off the SCLK_LF clock

The AONWDT can be configured using the following sequence:

- Unlock the AONWDT module using the AON_PMCTL:WDTLOCK register.
- Load the AON_PMCTL:WDTLOAD register with the desired timer load value.
- Lock the WDT module using the AON_PMCTL:WDTLOCK register.

Note

When this counter is running, there is no way to stop this counter other than device reset.

7.8 PRCM Registers

7.8.1 PRCM Registers

Table 7-8 lists the memory-mapped registers for the PRCM registers. All register offset addresses not listed in Table 7-8 should be considered as reserved locations and the register contents should not be modified.

Table 7-8. PRCM Registers

Offset	Acronym	Register Name	Section
0h	INFRCLKDIVR	Infrastructure Clock Division Factor For Run Mode	Section 7.8.1.1
4h	INFRCLKDIVS	Infrastructure Clock Division Factor For Sleep Mode	Section 7.8.1.2
8h	INFRCLKDIVDS	Infrastructure Clock Division Factor For DeepSleep Mode	Section 7.8.1.3
Ch	VDCTL	MCU Voltage Domain Control	Section 7.8.1.4
28h	CLKLOADCTL	Load PRCM Settings To CLKCTRL Power Domain	Section 7.8.1.5
2Ch	RFCCLKG	RFC Clock Gate	Section 7.8.1.6
30h	VIMSCLKG	VIMS Clock Gate	Section 7.8.1.7
3Ch	SECDMACLKGR	SEC (PKA And TRNG And CRYPTO) And UDMA Clock Gate For Run And All Modes	Section 7.8.1.8
40h	SECDMACLKGS	SEC (PKA And TRNG And CRYPTO) And UDMA Clock Gate For Sleep Mode	Section 7.8.1.9
44h	SECDMACLKGDS	SEC (PKA And TRNG and CRYPTO) And UDMA Clock Gate For Deep Sleep Mode	Section 7.8.1.10
48h	GPIOCLKGR	GPIO Clock Gate For Run And All Modes	Section 7.8.1.11
4Ch	GPIOCLKGS	GPIO Clock Gate For Sleep Mode	Section 7.8.1.12
50h	GPIOCLKGDS	GPIO Clock Gate For Deep Sleep Mode	Section 7.8.1.13
54h	GPTCLKGR	GPT Clock Gate For Run And All Modes	Section 7.8.1.14
58h	GPTCLKGS	GPT Clock Gate For Sleep Mode	Section 7.8.1.15
5Ch	GPTCLKGDS	GPT Clock Gate For Deep Sleep Mode	Section 7.8.1.16
60h	I2CCLKGR	I2C Clock Gate For Run And All Modes	Section 7.8.1.17
64h	I2CCLKGS	I2C Clock Gate For Sleep Mode	Section 7.8.1.18
68h	I2CCLKGDS	I2C Clock Gate For Deep Sleep Mode	Section 7.8.1.19
6Ch	UARTCLKGR	UART Clock Gate For Run And All Modes	Section 7.8.1.20
70h	UARTCLKGS	UART Clock Gate For Sleep Mode	Section 7.8.1.21
74h	UARTCLKGDS	UART Clock Gate For Deep Sleep Mode	Section 7.8.1.22
78h	SPICLKGR	SPI Clock Gate For Run And All Modes	Section 7.8.1.23
7Ch	SPICLKGS	SPI Clock Gate For Sleep Mode	Section 7.8.1.24
80h	SPICLKGDS	SPI Clock Gate For Deep Sleep Mode	Section 7.8.1.25
84h	I2SCLKGR	I2S Clock Gate For Run And All Modes	Section 7.8.1.26
88h	I2SCLKGS	I2S Clock Gate For Sleep Mode	Section 7.8.1.27
8Ch	I2SCLKGDS	I2S Clock Gate For Deep Sleep Mode	Section 7.8.1.28
B4h	SYSBUSCLKDIV	Internal	Section 7.8.1.29
B8h	CPUCLKDIV	Internal	Section 7.8.1.30
BCh	PERBUSCPUCLKDIV	Internal	Section 7.8.1.31
C4h	PERDMACLKDIV	Internal	Section 7.8.1.32
C8h	I2SBCLKSEL	I2S Clock Control	Section 7.8.1.33
CCh	GPTCLKDIV	GPT Scalar	Section 7.8.1.34
D0h	I2SCLKCTL	I2S Clock Control	Section 7.8.1.35
D4h	I2SMCLKDIV	MCLK Division Ratio	Section 7.8.1.36
D8h	I2SBCLKDIV	BCLK Division Ratio	Section 7.8.1.37
DCh	I2SWCLKDIV	WCLK Division Ratio	Section 7.8.1.38
F0h	RESETSECDMA	RESET For SEC (PKA And TRNG And CRYPTO) And UDMA	Section 7.8.1.39

Table 7-8. PRCM Registers (continued)

Offset	Acronym	Register Name	Section
F4h	RESETGPIO	RESET For GPIO IPs	Section 7.8.1.40
F8h	RESETGPT	RESET For GPT Ips	Section 7.8.1.41
FCh	RESETI2C	RESET For I2C IPs	Section 7.8.1.42
100h	RESETUART	RESET For UART IPs	Section 7.8.1.43
104h	RESETSPI	RESET For SPI IPs	Section 7.8.1.44
108h	RESETI2S	RESET For I2S IP	Section 7.8.1.45
12Ch	PDCTL0	Power Domain Control	Section 7.8.1.46
130h	PDCTL0RFC	RFC Power Domain Control	Section 7.8.1.47
134h	PDCTL0SERIAL	SERIAL Power Domain Control	Section 7.8.1.48
138h	PDCTL0PERIPH	PERIPH Power Domain Control	Section 7.8.1.49
140h	PDSTAT0	Power Domain Status	Section 7.8.1.50
144h	PDSTAT0RFC	RFC Power Domain Status	Section 7.8.1.51
148h	PDSTAT0SERIAL	SERIAL Power Domain Status	Section 7.8.1.52
14Ch	PDSTAT0PERIPH	PERIPH Power Domain Status	Section 7.8.1.53
17Ch	PDCTL1	Power Domain Control	Section 7.8.1.54
184h	PDCTL1CPU	CPU Power Domain Direct Control	Section 7.8.1.55
188h	PDCTL1RFC	RFC Power Domain Direct Control	Section 7.8.1.56
18Ch	PDCTL1VIMS	VIMS Mode Direct Control	Section 7.8.1.57
194h	PDSTAT1	Power Manager Status	Section 7.8.1.58
198h	PDSTAT1BUS	BUS Power Domain Direct Read Status	Section 7.8.1.59
19Ch	PDSTAT1RFC	RFC Power Domain Direct Read Status	Section 7.8.1.60
1A0h	PDSTAT1CPU	CPU Power Domain Direct Read Status	Section 7.8.1.61
1A4h	PDSTAT1VIMS	VIMS Mode Direct Read Status	Section 7.8.1.62
1CCh	RFCBITS	Control To RFC	Section 7.8.1.63
1D0h	RFCMODESEL	Selected RFC Mode	Section 7.8.1.64
1D4h	RFCMODEHWOPT	Allowed RFC Modes	Section 7.8.1.65
1E0h	PWRPROFSTAT	Power Profiler Register	Section 7.8.1.66
21Ch	MCUSRAMCFG	MCU SRAM configuration	Section 7.8.1.67
224h	RAMRETEN	Memory Retention Control	Section 7.8.1.68
290h	OSCIMSC	Oscillator Interrupt Mask Control	Section 7.8.1.69
294h	OSCRIS	Oscillator Raw Interrupt Status	Section 7.8.1.70
298h	OSCICR	Oscillator Raw Interrupt Clear	Section 7.8.1.71
2B0h	NVMNSCADDR	NVM Non-Secure Callable boundary Address	Section 7.8.1.72
2B4h	NVMNSADDR	NVM Non-Secure boundary Address	Section 7.8.1.73
2B8h	SRAMNSCADDR	SRAM Non-Secure Callable boundary Address	Section 7.8.1.74
2BCh	SRAMNSADDR	SRAM Non-Secure Callable boundary Address	Section 7.8.1.75
2C0h	BUSSECCFG	BUS Security Configuration Register	Section 7.8.1.76
2C4h	CPULOCK	CPU Lock Register	Section 7.8.1.77

Complex bit access types are encoded to fit into small table cells. Table 7-9 shows the codes that are used for access types in this section.

Table 7-9. PRCM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

7.8.1.1 INFRCLKDIVR Register (Offset = 0h) [Reset = 00000000h]

INFRCLKDIVR is shown in Table 7-10.

Return to the Summary Table.

Infrastructure Clock Division Factor For Run Mode

Table 7-10. INFRCLKDIVR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	RATIO	R/W	0h	Division rate for clocks driving modules in the MCU_AON domain when system CPU is in run mode. Division ratio affects both infrastructure clock and perbusull clock. 0h = Divide by 1 1h = Divide by 2 2h = Divide by 8 3h = Divide by 32

7.8.1.2 INFRCLKDIVS Register (Offset = 4h) [Reset = 00000000h]

INFRCLKDIVS is shown in Table 7-11.

Return to the Summary Table.

Infrastructure Clock Division Factor For Sleep Mode

Table 7-11. INFRCLKDIVS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	RATIO	R/W	0h	Division rate for clocks driving modules in the MCU_AON domain when system CPU is in sleep mode. Division ratio affects both infrastructure clock and perbusull clock. 0h = Divide by 1 1h = Divide by 2 2h = Divide by 8 3h = Divide by 32

7.8.1.3 INFRCLKDIVDS Register (Offset = 8h) [Reset = 00000000h]

INFRCLKDIVDS is shown in Table 7-12.

Return to the Summary Table.

Infrastructure Clock Division Factor For DeepSleep Mode

Table 7-12. INFRCLKDIVDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	RATIO	R/W	0h	Division rate for clocks driving modules in the MCU_AON domain when system CPU is in deepsleep mode. Division ratio affects both infrastructure clock and perbusull clock. 0h = Divide by 1 1h = Divide by 2 2h = Divide by 8 3h = Divide by 32

7.8.1.4 VDCTL Register (Offset = Ch) [Reset = 00000000h]

VDCTL is shown in Table 7-13.

Return to the Summary Table.

MCU Voltage Domain Control

Table 7-13. VDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ULDO	R/W	0h	Request PMCTL to switch to uLDO. 0: No request 1: Assert request when possible The bit will have no effect before the following requirements are met: 1. PDCTL1.CPU_ON = 0 2. PDCTL1.VIMS_MODE = x0 3. SECDMACLKGDS.DMA_CLK_EN = 0 and S.CRYPTO_CLK_EN] = 0 and SECDMACLKGR.DMA_AM_CLK_EN = 0 (Note: Settings must be loaded with CLKLOADCTL.LOAD) 4. SECDMACLKGDS.CRYPTO_CLK_EN = 0 and SECDMACLKGR.CRYPTO_AM_CLK_EN = 0 (Note: Settings must be loaded with CLKLOADCTL.LOAD) 5. I2SCLKGDS.CLK_EN = 0 and I2SCLKGR.AM_CLK_EN = 0 (Note: Settings must be loaded with CLKLOADCTL.LOAD) 6. RFC do no request access to BUS 7. System CPU in deepsleep

7.8.1.5 CLKLOADCTL Register (Offset = 28h) [Reset = 00000002h]

CLKLOADCTL is shown in Table 7-14.

Return to the Summary Table.

Load PRCM Settings To CLKCTRL Power Domain

Table 7-14. CLKLOADCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	LOAD_DONE	R	1h	Status of LOAD. Will be cleared to 0 when any of the registers requiring a LOAD is written to, and be set to 1 when a LOAD is done. Note that writing no change to a register will result in the LOAD_DONE being cleared. 0 : One or more registers have been write accessed after last LOAD 1 : No registers are write accessed after last LOAD
0	LOAD	W	0h	0: No action 1: Load settings to CLKCTRL. Bit is HW cleared. Multiple changes to settings may be done before LOAD is written once so all changes takes place at the same time. LOAD can also be done after single setting updates. Registers that needs to be followed by LOAD before settings being applied are: - SYSBUSCLKDIV - CPUCLKDIV - PERBUSCPUCLKDIV - PERDMACLKDIV - PERBUSCPUCLKG - RFCCLKG - VIMSCLKG - SECDMACLKGR - SECDMACLKGS - SECDMACLKGDS - GPIOCLKGR - GPIOCLKGS - GPIOCLKGDS - GPTCLKGR - GPTCLKGS - GPTCLKGDS - GPTCLKDIV - I2CCLKGR - I2CCLKGS - I2CCLKGDS - SPICLKGR - SPICLKGS - SPICLKGDS - UARTCLKGR - UARTCLKGS - UARTCLKGDS - I2SCLKGR - I2SCLKGS - I2SCLKGDS - I2SBCLKSEL - I2SCLKCTL - I2SMCLKDIV - I2SBCLKDIV - I2SWCLKDIV - MCUSRAMCFG

7.8.1.6 RFCCLKG Register (Offset = 2Ch) [Reset = 00000001h]

RFCCLKG is shown in Table 7-15.

Return to the Summary Table.

RFC Clock Gate

Table 7-15. RFCCLKG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	1h	0: Disable Clock 1: Enable clock if RFC power domain is on For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.7 VIMSCLKG Register (Offset = 30h) [Reset = 00000003h]

VIMSCLKG is shown in Table 7-16.

Return to the Summary Table.

VIMS Clock Gate

Table 7-16. VIMSCLKG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	CLK_EN	R/W	3h	00: Disable clock 01: Disable clock when SYSBUS clock is disabled 11: Enable clock For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.8 SECDMACLKGR Register (Offset = 3Ch) [Reset = 00000000h]

SECDMACLKGR is shown in Table 7-17.

Return to the Summary Table.

SEC (PKA And TRNG And CRYPTO) And UDMA Clock Gate For Run And All Modes

Table 7-17. SECDMACLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DMA_AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides DMA_CLK_EN, SECDMACLKGS.DMA_CLK_EN and SECDMACLKGDS.DMA_CLK_EN when enabled. SYSBUS clock will always run when enabled For changes to take effect, CLKLOADCTL.LOAD needs to be written
23-20	RESERVED	R	0h	Reserved
19	PKA_ZERIOZE_RESET_N	R/W	0h	Zeroization logic hardware reset. 0: pka_zeroize logic inactive. 1: pka_zeroize of memory is enabled. This register must remain active until the memory are completely zeroized which requires 256 periods on systembus clock.
18	PKA_AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides PKA_CLK_EN, SECDMACLKGS.PKA_CLK_EN and SECDMACLKGDS.PKA_CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written
17	TRNG_AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides TRNG_CLK_EN, SECDMACLKGS.TRNG_CLK_EN and SECDMACLKGDS.TRNG_CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written
16	CRYPTO_AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CRYPTO_CLK_EN, SECDMACLKGS.CRYPTO_CLK_EN and SECDMACLKGDS.CRYPTO_CLK_EN when enabled. SYSBUS clock will always run when enabled For changes to take effect, CLKLOADCTL.LOAD needs to be written
15-9	RESERVED	R	0h	Reserved
8	DMA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by DMA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
7-3	RESERVED	R	0h	Reserved

Table 7-17. SECDMACLKGR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	PKA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by PKA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
1	TRNG_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by TRNG_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
0	CRYPTO_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by CRYPTO_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.9 SECDMACLKGS Register (Offset = 40h) [Reset = 00000000h]

SECDMACLKGS is shown in Table 7-18.

Return to the Summary Table.

SEC (PKA And TRNG And CRYPTO) And UDMA Clock Gate For Sleep Mode

Table 7-18. SECDMACLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	DMA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.DMA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
7-3	RESERVED	R	0h	Reserved
2	PKA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.PKA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
1	TRNG_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.TRNG_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
0	CRYPTO_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.CRYPTO_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.10 SECDMACLKGDS Register (Offset = 44h) [Reset = 00000000h]

SECDMACLKGDS is shown in Table 7-19.

Return to the Summary Table.

SEC (PKA And TRNG and CRYPTO) And UDMA Clock Gate For Deep Sleep Mode

Table 7-19. SECDMACLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	DMA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.DMA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
7-3	RESERVED	R	0h	Reserved
2	PKA_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SECDMACLKGR.PKA_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
1	TRNG_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock SYSBUS clock will always run when enabled Can be forced on by SECDMACLKGR.TRNG_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written
0	CRYPTO_CLK_EN	R/W	0h	0: Disable clock 1: Enable clock SYSBUS clock will always run when enabled Can be forced on by SECDMACLKGR.CRYPTO_AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.11 GPIOCLKGR Register (Offset = 48h) [Reset = 00000000h]

GPIOCLKGR is shown in Table 7-20.

Return to the Summary Table.

GPIO Clock Gate For Run And All Modes

Table 7-20. GPIOCLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, GPIOCLKGS.CLK_EN and GPIOCLKGDS.CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written
7-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.12 GPIOCLKGS Register (Offset = 4Ch) [Reset = 00000000h]

GPIOCLKGS is shown in Table 7-21.

Return to the Summary Table.

GPIO Clock Gate For Sleep Mode

Table 7-21. GPIOCLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by GPIOCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.13 GPIOCLKGDS Register (Offset = 50h) [Reset = 00000000h]

GPIOCLKGDS is shown in Table 7-22.

Return to the Summary Table.

GPIO Clock Gate For Deep Sleep Mode

Table 7-22. GPIOCLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by GPIOCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.14 GPTCLKGR Register (Offset = 54h) [Reset = 00000000h]

GPTCLKGR is shown in Table 7-23.

Return to the Summary Table.

GPT Clock Gate For Run And All Modes

Table 7-23. GPTCLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11-8	AM_CLK_EN	R/W	0h	Each bit below has the following meaning: 0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, GPTCLKGS.CLK_EN and GPTCLKGDS.CLK_EN when enabled. ENUMs can be combined For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for GPT0 in all modes 2h = Enable clock for GPT1 in all modes 4h = Enable clock for GPT2 in all modes 8h = Enable clock for GPT3 in all modes
7-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	Each bit below has the following meaning: 0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN ENUMs can be combined For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for GPT0 2h = Enable clock for GPT1 4h = Enable clock for GPT2 8h = Enable clock for GPT3

7.8.1.15 GPTCLKGS Register (Offset = 58h) [Reset = 00000000h]

GPTCLKGS is shown in Table 7-24.

Return to the Summary Table.

GPT Clock Gate For Sleep Mode

Table 7-24. GPTCLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	Each bit below has the following meaning: 0: Disable clock 1: Enable clock Can be forced on by GPTCLKGR.AM_CLK_EN ENUMs can be combined For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for GPT0 2h = Enable clock for GPT1 4h = Enable clock for GPT2 8h = Enable clock for GPT3

7.8.1.16 GPTCLKGDS Register (Offset = 5Ch) [Reset = 00000000h]

GPTCLKGDS is shown in Table 7-25.

Return to the Summary Table.

GPT Clock Gate For Deep Sleep Mode

Table 7-25. GPTCLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	Each bit below has the following meaning: 0: Disable clock 1: Enable clock Can be forced on by GPTCLKGR.AM_CLK_EN ENUMs can be combined For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for GPT0 2h = Enable clock for GPT1 4h = Enable clock for GPT2 8h = Enable clock for GPT3

7.8.1.17 I2CCLKGR Register (Offset = 60h) [Reset = 00000000h]

I2CCLKGR is shown in Table 7-26.

Return to the Summary Table.

I2C Clock Gate For Run And All Modes

Table 7-26. I2CCLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-8	AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, I2CCLKGS.CLK_EN and I2CCLKGDS.CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for I2C0 2h = Enable clock for I2C1
7-2	RESERVED	R	0h	Reserved
1-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for I2C0 2h = Enable clock for I2C1

7.8.1.18 I2CCLKGS Register (Offset = 64h) [Reset = 00000000h]

I2CCLKGS is shown in Table 7-27.

Return to the Summary Table.

I2C Clock Gate For Sleep Mode

Table 7-27. I2CCLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by I2CCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for I2C0 2h = Enable clock for I2C1

7.8.1.19 I2CCLKGDS Register (Offset = 68h) [Reset = 00000000h]

I2CCLKGDS is shown in Table 7-28.

Return to the Summary Table.

I2C Clock Gate For Deep Sleep Mode

Table 7-28. I2CCLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	RESERVED	R/W	0h	
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by I2CCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for I2C0 2h = Enable clock for I2C1

7.8.1.20 UARTCLKGR Register (Offset = 6Ch) [Reset = 00000000h]

UARTCLKGR is shown in Table 7-29.

Return to the Summary Table.

UART Clock Gate For Run And All Modes

Table 7-29. UARTCLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11-8	AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, UARTCLKGS.CLK_EN and UARTCLKGDS.CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for UART0 2h = Enable clock for UART1 4h = Enable clock for UART2 8h = Enable clock for UART3
7-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for UART0 2h = Enable clock for UART1 4h = Enable clock for UART2 8h = Enable clock for UART3

7.8.1.21 UARTCLKGS Register (Offset = 70h) [Reset = 00000000h]

UARTCLKGS is shown in Table 7-30.

Return to the Summary Table.

UART Clock Gate For Sleep Mode

Table 7-30. UARTCLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by UARTCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for UART0 2h = Enable clock for UART1 4h = Enable clock for UART2 8h = Enable clock for UART3

7.8.1.22 UARTCLKGDS Register (Offset = 74h) [Reset = 00000000h]

UARTCLKGDS is shown in Table 7-31.

Return to the Summary Table.

UART Clock Gate For Deep Sleep Mode

Table 7-31. UARTCLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by UARTCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for UART0 2h = Enable clock for UART1 4h = Enable clock for UART2 8h = Enable clock for UART3

7.8.1.23 SPICLKGR Register (Offset = 78h) [Reset = 00000000h]

SPICLKGR is shown in Table 7-32.

Return to the Summary Table.

SPI Clock Gate For Run And All Modes

Table 7-32. SPICLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11-8	AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, SPICLKGS.CLK_EN and SPICLKGDS.CLK_EN when enabled. For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for SPI0 2h = Enable clock for SPI1 4h = Enable clock for SPI2 8h = Enable clock for SPI3
7-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for SPI0 2h = Enable clock for SPI1 4h = Enable clock for SPI2 8h = Enable clock for SPI3

7.8.1.24 SPICLKGS Register (Offset = 7Ch) [Reset = 00000000h]

SPICLKGS is shown in Table 7-33.

Return to the Summary Table.

SPI Clock Gate For Sleep Mode

Table 7-33. SPICLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SPICLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for SPI0 2h = Enable clock for SPI1 4h = Enable clock for SPI2 8h = Enable clock for SPI3

7.8.1.25 SPICLKGDS Register (Offset = 80h) [Reset = 00000000h]

SPICLKGDS is shown in Table 7-34.

Return to the Summary Table.

SPI Clock Gate For Deep Sleep Mode

Table 7-34. SPICLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by SPICLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written 1h = Enable clock for SPI0 2h = Enable clock for SPI1 4h = Enable clock for SPI2 8h = Enable clock for SPI3

7.8.1.26 I2SCLKGR Register (Offset = 84h) [Reset = 00000000h]

I2SCLKGR is shown in Table 7-35.

Return to the Summary Table.

I2S Clock Gate For Run And All Modes

Table 7-35. I2SCLKGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	AM_CLK_EN	R/W	0h	0: No force 1: Force clock on for all modes (Run, Sleep and Deep Sleep) Overrides CLK_EN, I2SCLKGS.CLK_EN and I2SCLKGDS.CLK_EN when enabled. SYSBUS clock will always run when enabled For changes to take effect, CLKLOADCTL.LOAD needs to be written
7-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.27 I2SCLKGS Register (Offset = 88h) [Reset = 00000000h]

I2SCLKGS is shown in Table 7-36.

Return to the Summary Table.

I2S Clock Gate For Sleep Mode

Table 7-36. I2SCLKGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock Can be forced on by I2SCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.28 I2SCLKGDS Register (Offset = 8Ch) [Reset = 00000000h]

I2SCLKGDS is shown in Table 7-37.

Return to the Summary Table.

I2S Clock Gate For Deep Sleep Mode

Table 7-37. I2SCLKGDS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	CLK_EN	R/W	0h	0: Disable clock 1: Enable clock SYSBUS clock will always run when enabled Can be forced on by I2SCLKGR.AM_CLK_EN For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.29 SYSBUSCLKDIV Register (Offset = B4h) [Reset = 00000000h]

SYSBUSCLKDIV is shown in Table 7-38.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 7-38. SYSBUSCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	RATIO	R/W	0h	Internal. Only to be used through TI provided API.

7.8.1.30 CPUCLKDIV Register (Offset = B8h) [Reset = 00000000h]

CPUCLKDIV is shown in Table 7-39.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 7-39. CPUCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RATIO	R/W	0h	Internal. Only to be used through TI provided API.

7.8.1.31 PERBUSCPUCLKDIV Register (Offset = BCh) [Reset = 00000000h]

PERBUSCPUCLKDIV is shown in Table 7-40.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 7-40. PERBUSCPUCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	RATIO	R/W	0h	Internal. Only to be used through TI provided API.

7.8.1.32 PERDMACLKDIV Register (Offset = C4h) [Reset = 00000000h]

PERDMACLKDIV is shown in Table 7-41.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 7-41. PERDMACLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	RATIO	R/W	0h	Internal. Only to be used through TI provided API.

7.8.1.33 I2SBCLKSEL Register (Offset = C8h) [Reset = 00000000h]

I2SBCLKSEL is shown in Table 7-42.

Return to the Summary Table.

I2S Clock Control

Table 7-42. I2SBCLKSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SRC	R/W	0h	BCLK source selector 0: Use external BCLK 1: Use internally generated clock For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.34 GPTCLKDIV Register (Offset = CCh) [Reset = 00000000h]

GPTCLKDIV is shown in Table 7-43.

Return to the Summary Table.

GPT Scalar

Table 7-43. GPTCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	RATIO	R/W	0h	Scalar used for GPTs. The division rate will be constant and ungated for Run / Sleep / DeepSleep mode. For changes to take effect, CLKLOADCTL.LOAD needs to be written Other values are not supported. 0h = Divide by 1 1h = Divide by 2 2h = Divide by 4 3h = Divide by 8 4h = Divide by 16 5h = Divide by 32 6h = Divide by 64 7h = Divide by 128 8h = Divide by 256

7.8.1.35 I2SCLKCTL Register (Offset = D0h) [Reset = 00000000h]

I2SCLKCTL is shown in Table 7-44.

Return to the Summary Table.

I2S Clock Control

Table 7-44. I2SCLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	SMPL_ON_POSEDGE	R/W	0h	On the I2S serial interface, data and WCLK is sampled and clocked out on opposite edges of BCLK. 0 - data and WCLK are sampled on the negative edge and clocked out on the positive edge. 1 - data and WCLK are sampled on the positive edge and clocked out on the negative edge. For changes to take effect, CLKLOADCTL.LOAD needs to be written
2-1	WCLK_PHASE	R/W	0h	Decides how the WCLK division ratio is calculated and used to generate different duty cycles (See I2SWCLKDIV.WDIV). 0: Single phase 1: Dual phase 2: User Defined 3: Reserved/Undefined For changes to take effect, CLKLOADCTL.LOAD needs to be written
0	EN	R/W	0h	0: MCLK, BCLK and WCLK will be static low 1: Enables the generation of MCLK, BCLK and WCLK For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.36 I2SMCLKDIV Register (Offset = D4h) [Reset = 00000000h]

I2SMCLKDIV is shown in Table 7-45.

Return to the Summary Table.

MCLK Division Ratio

Table 7-45. I2SMCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-0	MDIV	R/W	0h	An unsigned factor of the division ratio used to generate MCLK [2-1024]: MCLK = MCUCLK/MDIV[Hz] MCUCLK is 48MHz. A value of 0 is interpreted as 1024. A value of 1 is invalid. If MDIV is odd the low phase of the clock is one MCUCLK period longer than the high phase. For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.37 I2SBCLKDIV Register (Offset = D8h) [Reset = 00000000h]

I2SBCLKDIV is shown in Table 7-46.

Return to the Summary Table.

BCLK Division Ratio

Table 7-46. I2SBCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-0	BDIV	R/W	0h	An unsigned factor of the division ratio used to generate I2S BCLK [2-1024]: BCLK = MCUCLK/BDIV[Hz] MCUCLK is 48MHz. A value of 0 is interpreted as 1024. A value of 1 is invalid. If BDIV is odd and I2SCLKCTL.SMPL_ON_POSEDGE = 0, the low phase of the clock is one MCUCLK period longer than the high phase. If BDIV is odd and I2SCLKCTL.SMPL_ON_POSEDGE = 1 , the high phase of the clock is one MCUCLK period longer than the low phase. For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.38 I2SWCLKDIV Register (Offset = DCh) [Reset = 00000000h]

I2SWCLKDIV is shown in Table 7-47.

Return to the Summary Table.

WCLK Division Ratio

Table 7-47. I2SWCLKDIV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	WDIV	R/W	0h	If I2SCLKCTL.WCLK_PHASE = 0, Single phase. WCLK is high one BCLK period and low WDIV[9:0] (unsigned, [1-1023]) BCLK periods. WCLK = MCUCLK / BDIV*(WDIV[9:0] + 1) [Hz] MCUCLK is 48MHz. If I2SCLKCTL.WCLK_PHASE = 1, Dual phase. Each phase on WCLK (50% duty cycle) is WDIV[9:0] (unsigned, [1-1023]) BCLK periods. WCLK = MCUCLK / BDIV*(2*WDIV[9:0]) [Hz] If I2SCLKCTL.WCLK_PHASE = 2, User defined. WCLK is high WDIV[7:0] (unsigned, [1-255]) BCLK periods and low WDIV[15:8] (unsigned, [1-255]) BCLK periods. WCLK = MCUCLK / (BDIV*(WDIV[7:0] + WDIV[15:8]) [Hz] For changes to take effect, CLKLOADCTL.LOAD needs to be written

7.8.1.39 RESETSECDMA Register (Offset = F0h) [Reset = 00000000h]

RESETSECDMA is shown in Table 7-48.

Return to the Summary Table.

RESET For SEC (PKA And TRNG And CRYPTO) And UDMA

Table 7-48. RESETSECDMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	DMA	W	0h	Write 1 to reset. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
7-3	RESERVED	R	0h	Reserved
2	PKA	W	0h	Write 1 to reset. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
1	TRNG	W	0h	Write 1 to reset. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
0	CRYPTO	W	0h	Write 1 to reset. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.40 RESETGPIO Register (Offset = F4h) [Reset = 00000000h]

RESETGPIO is shown in Table 7-49.

Return to the Summary Table.

RESET For GPIO IPs

Table 7-49. RESETGPIO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	GPIO	W	0h	0: No action 1: Reset GPIO. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.41 RESETGPT Register (Offset = F8h) [Reset = 00000000h]

RESETGPT is shown in Table 7-50.

Return to the Summary Table.

RESET For GPT Ips

Table 7-50. RESETGPT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	GPT	W	0h	0: No action 1: Reset all GPTs. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.42 RESETI2C Register (Offset = FCh) [Reset = 00000000h]

RESETI2C is shown in Table 7-51.

Return to the Summary Table.

RESET For I2C IPs

Table 7-51. RESETI2C Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	I2C1	W	0h	0: No action 1: Reset I2C1. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
0	I2C0	W	0h	0: No action 1: Reset I2C0. HW cleared. Access will only have effect when SERIAL power domain is on, PDSTAT0.SERIAL_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.43 RESETUART Register (Offset = 100h) [Reset = 00000000h]

RESETUART is shown in Table 7-52.

Return to the Summary Table.

RESET For UART IPs

Table 7-52. RESETUART Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	UART3	W	0h	0: No action 1: Reset UART3. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
2	UART2	W	0h	0: No action 1: Reset UART2. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
1	UART1	W	0h	0: No action 1: Reset UART1. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
0	UART0	W	0h	0: No action 1: Reset UART0. HW cleared. Access will only have effect when SERIAL power domain is on, PDSTAT0.SERIAL_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.44 RESETSPI Register (Offset = 104h) [Reset = 00000000h]

RESETSPI is shown in Table 7-53.

Return to the Summary Table.

RESET For SPI IPs

Table 7-53. RESETSPI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	SPI3	W	0h	0: No action 1: Reset SPI3. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
2	SPI2	W	0h	0: No action 1: Reset SPI2. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
1	SPI1	W	0h	0: No action 1: Reset SPI1. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.
0	SPI0	W	0h	0: No action 1: Reset SPI0. HW cleared. Access will only have effect when SERIAL power domain is on, PDSTAT0.SERIAL_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.45 RESETI2S Register (Offset = 108h) [Reset = 00000000h]

RESETI2S is shown in Table 7-54.

Return to the Summary Table.

RESET For I2S IP

Table 7-54. RESETI2S Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	I2S	W	0h	0: No action 1: Reset module. HW cleared. Access will only have effect when PERIPH power domain is on, PDSTAT0.PERIPH_ON = 1 Before writing set FLASH:CFG.DIS_READACCESS = 1 to ensure the reset is not activated while executing from flash. This means one cannot execute from flash when using the SW reset.

7.8.1.46 PDCTL0 Register (Offset = 12Ch) [Reset = 00000000h]

PDCTL0 is shown in Table 7-55.

Return to the Summary Table.

Power Domain Control

Table 7-55. PDCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	PERIPH_ON	R/W	0h	PERIPH Power domain. 0: PERIPH power domain is powered down 1: PERIPH power domain is powered up
1	SERIAL_ON	R/W	0h	SERIAL Power domain. 0: SERIAL power domain is powered down 1: SERIAL power domain is powered up
0	RFC_ON	R/W	0h	0: RFC power domain powered off if also PDCTL1.RFC_ON = 0 1: RFC power domain powered on

7.8.1.47 PDCTL0RFC Register (Offset = 130h) [Reset = 00000000h]

PDCTL0RFC is shown in Table 7-56.

Return to the Summary Table.

RFC Power Domain Control

Table 7-56. PDCTL0RFC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R/W	0h	Alias for PDCTL0.RFC_ON

7.8.1.48 PDCTL0SERIAL Register (Offset = 134h) [Reset = 00000000h]

PDCTL0SERIAL is shown in Table 7-57.

Return to the Summary Table.

SERIAL Power Domain Control

Table 7-57. PDCTL0SERIAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R/W	0h	Alias for PDCTL0.SERIAL_ON

7.8.1.49 PDCTL0PERIPH Register (Offset = 138h) [Reset = 00000000h]

PDCTL0PERIPH is shown in Table 7-58.

Return to the Summary Table.

PERIPH Power Domain Control

Table 7-58. PDCTL0PERIPH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R/W	0h	Alias for PDCTL0.PERIPH_ON

7.8.1.50 PDSTAT0 Register (Offset = 140h) [Reset = 00000000h]

PDSTAT0 is shown in Table 7-59.

Return to the Summary Table.

Power Domain Status

Table 7-59. PDSTAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	PERIPH_ON	R	0h	PERIPH Power domain. 0: Domain may be powered down 1: Domain powered up (guaranteed)
1	SERIAL_ON	R	0h	SERIAL Power domain. 0: Domain may be powered down 1: Domain powered up (guaranteed)
0	RFC_ON	R	0h	RFC Power domain 0: Domain may be powered down 1: Domain powered up (guaranteed)

7.8.1.51 PDSTAT0RFC Register (Offset = 144h) [Reset = 00000000h]

PDSTAT0RFC is shown in Table 7-60.

Return to the Summary Table.

RFC Power Domain Status

Table 7-60. PDSTAT0RFC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	0h	Alias for PDSTAT0.RFC_ON

7.8.1.52 PDSTAT0SERIAL Register (Offset = 148h) [Reset = 00000000h]

PDSTAT0SERIAL is shown in Table 7-61.

Return to the Summary Table.

SERIAL Power Domain Status

Table 7-61. PDSTAT0SERIAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	0h	Alias for PDSTAT0.SERIAL_ON

7.8.1.53 PDSTAT0PERIPH Register (Offset = 14Ch) [Reset = 00000000h]

PDSTAT0PERIPH is shown in Table 7-62.

Return to the Summary Table.

PERIPH Power Domain Status

Table 7-62. PDSTAT0PERIPH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	0h	Alias for PDSTAT0.PERIPH_ON

7.8.1.54 PDCTL1 Register (Offset = 17Ch) [Reset = 0000000Ah]

PDCTL1 is shown in Table 7-63.

Return to the Summary Table.

Power Domain Control

Table 7-63. PDCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-3	VIMS_MODE	R/W	1h	00: VIMS power domain is only powered when CPU power domain is powered. 01: VIMS power domain is powered whenever the BUS power domain is powered. 1X: Block power up of VIMS power domain at next wake up. This mode only has effect when VIMS power domain is not powered. Used for Autonomous RF Core.
2	RFC_ON	R/W	0h	0: RFC power domain powered off if also PDCTL0.RFC_ON = 0 1: RFC power domain powered on Bit shall be used by RFC in autonomous mode but there is no HW restrictions from system CPU to access the bit.
1	CPU_ON	R/W	1h	0: Causes a power down of the CPU power domain when system CPU indicates it is idle. 1: Initiates power-on of the CPU power domain. This bit is automatically set by a WIC power-on event.
0	RESERVED	R	0h	Reserved

7.8.1.55 PDCTL1CPU Register (Offset = 184h) [Reset = 00000001h]

PDCTL1CPU is shown in Table 7-64.

Return to the Summary Table.

CPU Power Domain Direct Control

Table 7-64. PDCTL1CPU Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R/W	1h	This is an alias for PDCTL1.CPU_ON

7.8.1.56 PDCTL1RFC Register (Offset = 188h) [Reset = 00000000h]

PDCTL1RFC is shown in Table 7-65.

Return to the Summary Table.

RFC Power Domain Direct Control

Table 7-65. PDCTL1RFC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R/W	0h	This is an alias for PDCTL1.RFC_ON

7.8.1.57 PDCTL1VIMS Register (Offset = 18Ch) [Reset = 00000001h]

PDCTL1VIMS is shown in Table 7-66.

Return to the Summary Table.

VIMS Mode Direct Control

Table 7-66. PDCTL1VIMS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	MODE	R/W	1h	This is an alias for PDCTL1.VIMS_MODE

7.8.1.58 PDSTAT1 Register (Offset = 194h) [Reset = 0000001Ah]

PDSTAT1 is shown in Table 7-67.

Return to the Summary Table.

Power Manager Status

Table 7-67. PDSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	BUS_ON	R	1h	0: BUS domain not accessible 1: BUS domain is currently accessible
3	VIMS_ON	R	1h	0: VIMS domain not accessible 1: VIMS domain is currently accessible
2	RFC_ON	R	0h	0: RFC domain not accessible 1: RFC domain is currently accessible
1	CPU_ON	R	1h	0: CPU and BUS domain not accessible 1: CPU and BUS domains are both currently accessible
0	RESERVED	R	0h	Reserved

7.8.1.59 PDSTAT1BUS Register (Offset = 198h) [Reset = 00000001h]

PDSTAT1BUS is shown in Table 7-68.

Return to the Summary Table.

BUS Power Domain Direct Read Status

Table 7-68. PDSTAT1BUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	1h	This is an alias for PDSTAT1.BUS_ON

7.8.1.60 PDSTAT1RFC Register (Offset = 19Ch) [Reset = 00000000h]

PDSTAT1RFC is shown in Table 7-69.

Return to the Summary Table.

RFC Power Domain Direct Read Status

Table 7-69. PDSTAT1RFC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	0h	This is an alias for PDSTAT1.RFC_ON

7.8.1.61 PDSTAT1CPU Register (Offset = 1A0h) [Reset = 00000001h]

PDSTAT1CPU is shown in Table 7-70.

Return to the Summary Table.

CPU Power Domain Direct Read Status

Table 7-70. PDSTAT1CPU Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	1h	This is an alias for PDSTAT1.CPU_ON

7.8.1.62 PDSTAT1VIMS Register (Offset = 1A4h) [Reset = 00000001h]

PDSTAT1VIMS is shown in Table 7-71.

Return to the Summary Table.

VIMS Mode Direct Read Status

Table 7-71. PDSTAT1VIMS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ON	R	1h	This is an alias for PDSTAT1.VIMS_ON

7.8.1.63 RFCBITS Register (Offset = 1CCh) [Reset = 00000000h]

RFCBITS is shown in Table 7-72.

Return to the Summary Table.

Control To RFC

Table 7-72. RFCBITS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	READ	R/W	0h	Control bits for RFC. The RF core CPE processor will automatically check this register when it boots, and it can be used to immediately instruct CPE to perform some tasks at its start-up. The supported functionality is ROM-defined and may vary. See the technical reference manual for more details.

7.8.1.64 RFCMODESEL Register (Offset = 1D0h) [Reset = 00000000h]

RFCMODESEL is shown in Table 7-73.

Return to the Summary Table.

Selected RFC Mode

Table 7-73. RFCMODESEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	CURR	R/W	0h	Selects the set of commands that the RFC will accept. Only modes permitted by RFCMODEHWOPT.AVAIL are writable. See the technical reference manual for details. 0h = Select Mode 0 1h = Select Mode 1 2h = Select Mode 2 3h = Select Mode 3 4h = Select Mode 4 5h = Select Mode 5 6h = Select Mode 6 7h = Select Mode 7

7.8.1.65 RFCMODEHWOPT Register (Offset = 1D4h) [Reset = 00000000h]

RFCMODEHWOPT is shown in Table 7-74.

Return to the Summary Table.

Allowed RFC Modes

Table 7-74. RFCMODEHWOPT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	AVAIL	R	0h	Permitted RFC modes. More than one mode can be permitted. 1h = Mode 0 permitted 2h = Mode 1 permitted 4h = Mode 2 permitted 8h = Mode 3 permitted 10h = Mode 4 permitted 20h = Mode 5 permitted 40h = Mode 6 permitted 80h = Mode 7 permitted

7.8.1.66 PWRPROFSTAT Register (Offset = 1E0h) [Reset = 00000001h]

PWRPROFSTAT is shown in Table 7-75.

Return to the Summary Table.

Power Profiler Register

Table 7-75. PWRPROFSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	VALUE	R/W	1h	SW can use these bits to timestamp the application. These bits are also available through the testtap and can thus be used by the emulator to profile in real time.

7.8.1.67 MCUSRAMCFG Register (Offset = 21Ch) [Reset = 00000020h]

MCUSRAMCFG is shown in Table 7-76.

Return to the Summary Table.

MCU SRAM configuration

Table 7-76. MCUSRAMCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	PARITY_EN	R/W	0h	Parity enable 0: Parity disabled Parity section available as GPRAM 1: Parity enabled
5	BM_OFF	R/W	1h	NOT in use. Writing any other value than the reset value may result in undefined behavior.
4	PAGE	R/W	0h	NOT in use. Writing any other value than the reset value may result in undefined behavior.
3	PGS	R/W	0h	NOT in use. Writing any other value than the reset value may result in undefined behavior.
2	BM	R/W	0h	NOT in use. Writing any other value than the reset value may result in undefined behavior.
1	PCH_F	R/W	0h	NOT in use. Writing any other value than the reset value may result in undefined behavior.
0	PCH_L	R/W	0h	NOT in use. Writing any other value than the reset value may result in undefined behavior.

7.8.1.68 RAMRETEN Register (Offset = 224h) [Reset = 0000000Bh]

RAMRETEN is shown in Table 7-77.

Return to the Summary Table.

Memory Retention Control

Table 7-77. RAMRETEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	RFCULL	R/W	1h	0: Retention for RFC ULL SRAM disabled 1: Retention for RFC ULL SRAM enabled Memories controlled: CPEULLRAM
2	RFC	R/W	0h	0: Retention for RFC SRAM disabled 1: Retention for RFC SRAM enabled Memories controlled: CPERAM MCERAM RFERAM DSBRAM
1-0	VIMS	R/W	3h	0: Memory retention disabled 1: Memory retention enabled Bit 0: VIMS_TRAM Bit 1: VIMS_CRAM Legal modes depend on settings in VIMS:CTL.MODE 00: VIMS:CTL.MODE must be OFF before DEEPSLEEP is asserted - must be set to CACHE or SPLIT mode after waking up again 01: VIMS:CTL.MODE must be GPRAM before DEEPSLEEP is asserted. Must remain in GPRAM mode after wake up, alternatively select OFF mode first and then CACHE or SPLIT mode. 10: Illegal mode 11: No restrictions

7.8.1.69 OSCIMSC Register (Offset = 290h) [Reset = 00000036h]

OSCIMSC is shown in Table 7-78.

Return to the Summary Table.

Oscillator Interrupt Mask Control

Table 7-78. OSCIMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	HFSRCPENDIM	R/W	0h	0: Disable interrupt generation when HFSRCPEND is qualified 1: Enable interrupt generation when HFSRCPEND is qualified
6	LFSRCDONEIM	R/W	0h	0: Disable interrupt generation when LFSRCDONE is qualified 1: Enable interrupt generation when LFSRCDONE is qualified
5	XOSCDLFIM	R/W	1h	0: Disable interrupt generation when XOSCDLF is qualified 1: Enable interrupt generation when XOSCDLF is qualified
4	XOSCLFIM	R/W	1h	0: Disable interrupt generation when XOSCLF is qualified 1: Enable interrupt generation when XOSCLF is qualified
3	RCOSCDLFIM	R/W	0h	0: Disable interrupt generation when RCOSCDLF is qualified 1: Enable interrupt generation when RCOSCDLF is qualified
2	RCOSCLFIM	R/W	1h	0: Disable interrupt generation when RCOSCLF is qualified 1: Enable interrupt generation when RCOSCLF is qualified
1	XOSCHFIM	R/W	1h	0: Disable interrupt generation when XOSCHF is qualified 1: Enable interrupt generation when XOSCHF is qualified
0	RCOSCHFIM	R/W	0h	0: Disable interrupt generation when RCOSCHF is qualified 1: Enable interrupt generation when RCOSCHF is qualified

7.8.1.70 OSCRIS Register (Offset = 294h) [Reset = 00000000h]

OSCRIS is shown in Table 7-79.

Return to the Summary Table.

Oscillator Raw Interrupt Status

Table 7-79. OSCRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	HFSRCPENDRIS	R	0h	SCLK_HF source switch pending interrupt. After a write to DDI_0_OSC:CTL0.SCLK_HF_SRC_SEL leads to a SCLK_HF source change request, then the requested SCLK_HF source will be enabled and qualified. When the new source is ready to be used as a clock source, then the interrupt HSSRCPENDRIS will go high. When the Flash allows SCLK_HF source switching to take place after flash memory read access is disabled. At this time the actual SCLK_HF clock source switch will be performed, and the interrupt status HSSRCPENDRIS will go low. 0: Indicates SCLK_HF source is not ready to be switched 1: Indicates SCLK_HF source is ready to be switched Interrupt is qualified regardless of OSCIMSC.HFSRCPENDIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.HFSRCPENDC
6	LFSRCDONERIS	R	0h	SCLK_LF source switch done. The DDI_0_OSC:CTL0.SCLK_LF_SRC_SEL register field is used to request that the SCLK_LF source shall be changed. After an SCLK_LF clock source change is requested, the new source may need to be enabled and qualified before switching of clock source can be done. The interrupt LFRSRCDONERIS goes high to indicate that the SCLK_LF clock source switching has been performed. LFRSRCDONERIS will go low again when the next clock source change is requested by writing to DDI_0_OSC:CTL0.SCLK_LF_SRC_SEL . 0: Indicates SCLK_LF source switch has not completed 1: Indicates SCLK_LF source switch has completed Interrupt is qualified regardless of OSCIMSC.LFSRCDONEIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.LFSRCDONEC
5	XOSCDLFRIS	R	0h	The XOSCDLFRIS interrupt indicates when the XOSC_HF oscillator is ready to be used as a derived low-frequency clock source for SCLK_LF or ACLK_REF. When XOSCDLFRIS is high, XOSC_HF will be used as source for SCLK_LF when selected. When none of the system clocks have XOSC_HF selected as clock source, the XOSC_HF source is automatically disabled and the XOSCDLFRIS interrupt status will go low. 0: XOSCDLF has not been qualified 1: XOSCDLF has been qualified Interrupt is qualified regardless of OSCIMSC.XOSCDLFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.XOSCDLFC

Table 7-79. OSCRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	XOSCLFRIS	R	0h	The XOSCLFRIS interrupt indicates when the output of the XOSC_LF oscillator has been qualified with respect to frequency. The XOSCLFRIS interrupt status goes high when the XOSC_LF oscillator is ready to be used as a clock source. After the clock qualification is successful, XOSCLFRIS interrupt status remains high, and further qualification is turned off until the XOSC_LF oscillator is disabled. XOSCLFRIS interrupt status will go low only at initial power-on, or after the XOSC_LF oscillator has been disabled when being deselected as a clock source. 0: XOSCLF has not been qualified 1: XOSCLF has been qualified Interrupt is qualified regardless of OSCIMSC.XOSCLFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.XOSCLFC
3	RCOSCDLFRIS	R	0h	The RCOSCDLFRIS interrupt indicates when the RCOSC_HF oscillator is ready to be used as a derived low-frequency clock source for SCLK_LF or ACLK_REF. When RCOSCDLFRIS is high, RCOSC_HF will be used as source for SCLK_LF when selected. When none of the system clocks have RCOSC_HF selected as clock source, the RCOSC_HF source is automatically disabled and the RCOSCDLFRIS interrupt status will go low. If the SCLK_LF or ACLK_REF source is changed from RCOSC_HF derived to XOSC_HF derived low-frequency clock and the new source has not been qualified, then the clock will remain running on the original source. The RCOSCDLFRIS interrupt will then remain high. 0: RCOSCDLF has not been qualified 1: RCOSCDLF has been qualified Interrupt is qualified regardless of OSCIMSC.RCOSCDLFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.RCOSCDLFC
2	RCOSCLFRIS	R	0h	The RCOSCLFRIS interrupt indicates when the output of the RCOSC_LF oscillator has been qualified with respect to frequency. The RCOSCLFRIS interrupt status goes high when the RCOSC_LF oscillator is ready to be used as a clock source. After the clock qualification is successful, RCOSCLFRIS interrupt status remains high, and further qualification is turned off until the RCOSC_LF oscillator is disabled. RCOSCLFRIS interrupt status will go low only at initial power-on, or after the RCOSC_LF oscillator has been disabled when being deselected as a clock source. 0: RCOSCLF has not been qualified 1: RCOSCLF has been qualified Interrupt is qualified regardless of OSCIMSC.RCOSCLFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.RCOSCLFC
1	XOSCHFRIS	R	0h	The XOSCHFRIS interrupt indicates when the XOSC_HF oscillator has been qualified for use as a clock source. XOSCHFRIS is also used in TCXO mode (when DDI_0_OSC:XOSCHFCTL.TCXO_MODE is 1). When the XOSCHFRIS interrupt is high, the oscillator is qualified and will be used as a clock source when selected. The XOSCHFRIS interrupt goes low when the oscillator is disabled after being deselected as a clock source. 0: XOSC_HF has not been qualified 1: XOSC_HF has been qualified Interrupt is qualified regardless of OSCIMSC.XOSCHFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.XOSCHFC

Table 7-79. OSCRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RCOSCHFRIS	R	0h	The RCOSCHFRIS interrupt indicates when the RCOSC_HF oscillator has been qualified for use as a clock source When the RCOSCHFRIS interrupt is high, the oscillator is qualified and will be used as a clock source when selected. The RCOSCHFRIS interrupt goes low when the oscillator is disabled after being deselected as a clock source. 0: RCOSC_HF has not been qualified 1: RCOSC_HF has been qualified Interrupt is qualified regardless of OSCIMSC.RCOSCHFIM setting. The order of qualifying raw interrupt and enable of interrupt mask is indifferent for generating an OSC Interrupt. Set by HW. Cleared by writing to OSCICR.RCOSCHFC

7.8.1.71 OSCICR Register (Offset = 298h) [Reset = 00000000h]

OSCICR is shown in Table 7-80.

Return to the Summary Table.

Oscillator Raw Interrupt Clear

Table 7-80. OSCICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	HFSRCPENDC	W	0h	Writing 1 to this field clears the HFSRCPEND raw interrupt status. Writing 0 has no effect.
6	LFSRCDONEC	W	0h	Writing 1 to this field clears the LFSRCDONE raw interrupt status. Writing 0 has no effect.
5	XOSCDLFC	W	0h	Writing 1 to this field clears the XOSCDLF raw interrupt status. Writing 0 has no effect.
4	XOSCLFC	W	0h	Writing 1 to this field clears the XOSCLF raw interrupt status. Writing 0 has no effect.
3	RCOSCDLFC	W	0h	Writing 1 to this field clears the RCOSCDLF raw interrupt status. Writing 0 has no effect.
2	RCOSCLFC	W	0h	Writing 1 to this field clears the RCOSCLF raw interrupt status. Writing 0 has no effect.
1	XOSCHFC	W	0h	Writing 1 to this field clears the XOSCHF raw interrupt status. Writing 0 has no effect.
0	RCOSCHFC	W	0h	Writing 1 to this field clears the RCOSCHF raw interrupt status. Writing 0 has no effect.

7.8.1.72 NVMNSCADDR Register (Offset = 2B0h) [Reset = 00000000h]

NVMNSCADDR is shown in Table 7-81.

Return to the Summary Table.

NVM Non-Secure Callable boundary Address

Table 7-81. NVMNSCADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PARITY	R	0h	Register parity bit.
30-20	RESERVED	R	0h	Reserved
19-10	BOUNDARY	R/W	0h	Non-Secure callable boundary address. Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
9-0	RESERVED	R	0h	Reserved

7.8.1.73 NVMNSADDR Register (Offset = 2B4h) [Reset = 80100000h]

NVMNSADDR is shown in Table 7-82.

Return to the Summary Table.

NVM Non-Secure boundary Address

Table 7-82. NVMNSADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PARITY	R	1h	Register parity bit
30-21	RESERVED	R	0h	Reserved
20	BOUNDARY_MSB	R	1h	Non-Secure boundary address MSB HW controlled.
19-13	BOUNDARY	R/W	0h	Non-Secure boundary address. Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
12-0	RESERVED	R	0h	Reserved

7.8.1.74 SRAMNSCADDR Register (Offset = 2B8h) [Reset = 00000000h]

SRAMNSCADDR is shown in Table 7-83.

Return to the Summary Table.

SRAM Non-Secure Callable boundary Address

Table 7-83. SRAMNSCADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PARITY	R	0h	Register parity bit
30-19	RESERVED	R	0h	Reserved
18-10	BOUNDARY	R/W	0h	Non-Secure callable boundary address. Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
9-0	RESERVED	R	0h	Reserved

7.8.1.75 SRAMNSADDR Register (Offset = 2BCh) [Reset = 00048000h]

SRAMNSADDR is shown in Table 7-84.

Return to the Summary Table.

SRAM Non-Secure Callable boundary Address

Table 7-84. SRAMNSADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PARITY	R	0h	Register parity bit
30-19	RESERVED	R	0h	Reserved
18-10	BOUNDARY	R/W	120h	Non-Secure boundary address. Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
9-0	RESERVED	R	0h	Reserved

7.8.1.76 BUSSECCFG Register (Offset = 2C0h) [Reset = 000000FFh]

BUSSECCFG is shown in Table 7-85.

Return to the Summary Table.

BUS Security Configuration Register

Table 7-85. BUSSECCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VALID	R/W	0h	Security configuration valid Registers that needs to be followed by VALID before settings being applied are: - NVMNSCADDR - NVMNSADDR - SRAMNSCADDR - SRAMNSADDR - BUSSECCFG - CPULOCK
30-8	RESERVED	R	0h	Reserved
7-0	BUS_CFG	R/W	FFh	Bus interconnect security and firewall configuration 0xFF : Trustzone enabled 0xF9 : Trustzone disabled Others: Reserved. Software should not rely on the value of reserved and should not write reserved settings.

7.8.1.77 CPULOCK Register (Offset = 2C4h) [Reset = 8000001Fh]

CPULOCK is shown in Table 7-86.

Return to the Summary Table.

CPU Lock Register

Table 7-86. CPULOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	PARITY	R	1h	Register parity bit
30-5	RESERVED	R	0h	Reserved
4	LOCKNSVTOR	R/W	1h	When set will lock non-secure vector table base address Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
3	LOCKSVTAIRCR	R/W	1h	When set will lock - Secure vector table base address - Secure interrupt priority - Busfault - Hardfault NMI security target Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
2	LOCKSAU	R/W	1h	When set will lock SAU regions Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
1	LOCKNSMPU	R/W	1h	When set will lock non-secure MPU Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.
0	LOCKSMPU	R/W	1h	When set will lock secure MPU Writing this field when BUSSECCFG.VALID is set may result in undefined behavior.

7.8.2 AON_PMCTL Registers

Table 7-87 lists the memory-mapped registers for the AON_PMCTL registers. All register offset addresses not listed in Table 7-87 should be considered as reserved locations and the register contents should not be modified.

Table 7-87. AON_PMCTL Registers

Offset	Acronym	Register Name	Section
4h	AUXSCECLK	AUX SCE Clock Management	Section 7.8.2.1
8h	RAMCFG	RAM Configuration	Section 7.8.2.2
10h	PWRCTL	Power Management Control	Section 7.8.2.3
14h	PWRSTAT	AON Power and Reset Status	Section 7.8.2.4
18h	SHUTDOWN	Shutdown Control	Section 7.8.2.5
1Ch	RECHARGECFG	Recharge Controller Configuration	Section 7.8.2.6
20h	RECHARGESTAT	Recharge Controller Status	Section 7.8.2.7
24h	OSCCFG	Oscillator Configuration	Section 7.8.2.8
28h	RESETCTL	Reset Management	Section 7.8.2.9
2Ch	SLEEPCTL	Sleep Control	Section 7.8.2.10
34h	JTAGCFG	JTAG Configuration	Section 7.8.2.11
3Ch	JTAGUSERCODE	JTAG USERCODE	Section 7.8.2.12
C4h	WDTLOAD	Configuration	Section 7.8.2.13
C8h	WDTTEST	Test Mode	Section 7.8.2.14
D0h	WDTLOCK	Lock	Section 7.8.2.15

Complex bit access types are encoded to fit into small table cells. Table 7-88 shows the codes that are used for access types in this section.

Table 7-88. AON_PMCTL Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

7.8.2.1 AUXSCECLK Register (Offset = 4h) [Reset = 00000000h]

AUXSCECLK is shown in Table 7-89.

Return to the Summary Table.

AUX SCE Clock Management
This register contains bitfields that are relevant for setting up the clock to the AUX domain.

Table 7-89. AUXSCECLK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	PD_SRC	R/W	0h	Selects the clock source for the AUX domain when AUX is in powerdown mode. Note: Switching the clock source is guaranteed to be glitch-free 0h = No clock 1h = LF clock (SCLK_LF)
7-1	RESERVED	R	0h	Reserved
0	SRC	R/W	0h	Selects the clock source for the AUX domain when AUX is in active mode. Note: Switching the clock source is guaranteed to be glitch-free 0h = HF Clock divided by 2 (SCLK_HFDIV2) 1h = MF Clock (SCLK_MF)

7.8.2.2 RAMCFG Register (Offset = 8h) [Reset = 0001000Fh]

RAMCFG is shown in Table 7-90.

Return to the Summary Table.

RAM Configuration
This register contains power management related configuration for the SRAM in the MCU and AUX domain.

Table 7-90. RAMCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17	AUX_SRAM_PWR_OFF	R/W	0h	Internal. Only to be used through TI provided API.
16	AUX_SRAM_RET_EN	R/W	1h	Internal. Only to be used through TI provided API.
15-4	RESERVED	R	0h	Reserved
3-0	BUS_SRAM_RET_EN	R/W	Fh	MCU SRAM is partitioned into 8 banks . This register controls which of the banks that has retention during MCU Bus domain power off 0h = Retention is disabled 1h = Retention on for BANK[0]: BANK[1]: BANK[2]: BANK[3] + PARITY_BANK retained 3h = Retention on for BANK[0]: BANK[1]: BANK[2]: BANK[3]:BANK[4] + PARITY_BANK retained 7h = Retention on for BANK[0]: BANK[1]: BANK[2]: BANK[3]:BANK[4]: BANK[5] + PARITY_BANK retained Fh = Retention on for all banks BANK[0]: BANK[1]: BANK[2]: BANK[3]:BANK[4]: BANK[5]: BANK[6]: BANK[7] + PARITY_BANK retained

7.8.2.3 PWRCTL Register (Offset = 10h) [Reset = 00000000h]

PWRCTL is shown in Table 7-91.

Return to the Summary Table.

Power Management Control
This register controls bitfields for setting low level power management features such as selection of regulator for VDDR supply and control of IO ring where certain segments can be enabled / disabled.

Table 7-91. PWRCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	DCDC_ACTIVE	R/W	0h	Select to use DCDC regulator for VDDR in active mode 0: Use GLDO for regulation of VDDR in active mode. 1: Use DCDC for regulation of VDDR in active mode. DCDC_EN must also be set for DCDC to be used as regulator for VDDR in active mode
1	EXT_REG_MODE	R	0h	Status of source for VDDRsupply: 0: DCDC or GLDO are generating VDDR 1: DCDC and GLDO are bypassed and an external regulator supplies VDDR
0	DCDC_EN	R/W	0h	Select to use DCDC regulator during recharge of VDDR 0: Use GLDO for recharge of VDDR 1: Use DCDC for recharge of VDDR Note: This bitfield should be set to the same as DCDC_ACTIVE

7.8.2.4 PWRSTAT Register (Offset = 14h) [Reset = 03C00003h]

PWRSTAT is shown in Table 7-92.

Return to the Summary Table.

AON Power and Reset Status
This register is used to monitor various power management related signals in AON. All other signals than JTAG_PD_ON, AUX_BUS_RESET_DONE, and AUX_RESET_DONE are for test, calibration and debug purpose only.

Table 7-92. PWRSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	JTAG_PD_ON	R	0h	Indicates JTAG power state: 0: JTAG is powered off 1: JTAG is powered on
1	AUX_BUS_RESET_DONE	R	1h	Indicates Reset Done from AUX Bus: 0: AUX Bus is being reset 1: AUX Bus reset is released
0	AUX_RESET_DONE	R	1h	Indicates Reset Done from AUX: 0: AUX is being reset 1: AUX reset is released

7.8.2.5 SHUTDOWN Register (Offset = 18h) [Reset = 00000000h]

SHUTDOWN is shown in Table 7-93.

Return to the Summary Table.

Shutdown Control
This register contains bitfields required for entering shutdown mode

Table 7-93. SHUTDOWN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Shutdown control. 0: Do not write 0 to this bit. 1: Immediately start the process to enter shutdown mode

7.8.2.6 RECHARGECFG Register (Offset = 1Ch) [Reset = C0000000h]

RECHARGECFG is shown in Table 7-94.

Return to the Summary Table.

Recharge Controller Configuration
This register sets all relevant parameters for controlling the recharge algorithm.

Table 7-94. RECHARGECFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MODE	R/W	3h	Selects recharge algorithm for VDDR when the system is running on the uLDO 0h = Recharge disabled 1h = Static timer 2h = Adaptive timer 3h = External recharge comparator. Note that the clock to the recharge comparator must be enabled, [ANATOP_MMAP:ADI_3_REFSYS:CTL_RECHARGE_CMP0:COMP_CLK_DISABLE], before selecting this recharge algorithm.
29-24	RESERVED	R	0h	Reserved
23-20	C2	R/W	0h	Internal. Only to be used through TI provided API.
19-16	C1	R/W	0h	Internal. Only to be used through TI provided API.
15-11	MAX_PER_M	R/W	0h	Internal. Only to be used through TI provided API.
10-8	MAX_PER_E	R/W	0h	Internal. Only to be used through TI provided API.
7-3	PER_M	R/W	0h	Internal. Only to be used through TI provided API.
2-0	PER_E	R/W	0h	Internal. Only to be used through TI provided API.

7.8.2.7 RECHARGESTAT Register (Offset = 20h) [Reset = 00000000h]

RECHARGESTAT is shown in Table 7-95.

Return to the Summary Table.

Recharge Controller Status
This register controls various status registers which are updated during recharge. The register is mostly intended for test and debug.

Table 7-95. RECHARGESTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-16	VDDR_SMPLS	R	0h	The last 4 VDDR samples. For each bit: 0: VDDR was below VDDR_OK threshold when recharge started 1: VDDR was above VDDR_OK threshold when recharge started The register is updated prior to every recharge period with a shift left, and bit 0 is updated with the last VDDR sample.
15-0	MAX_USED_PER	R/W	0h	Shows the maximum number of 32kHz periods that have separated two recharge cycles and VDDR still was above VDDR_OK threshold when the latter recharge started. This register can be used as an indication of the leakage current during standby. This bitfield is cleared to 0 when writing this register.

7.8.2.8 OSCCFG Register (Offset = 24h) [Reset = 00000000h]

OSCCFG is shown in Table 7-96.

Return to the Summary Table.

Oscillator Configuration
This register sets the period for Amplitude compensation requests sent to the oscillator control system. The amplitude compensations is only applicable when XOSC_HF is running in low power mode.

Table 7-96. OSCCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-3	PER_M	R/W	0h	Internal. Only to be used through TI provided API.
2-0	PER_E	R/W	0h	Internal. Only to be used through TI provided API.

7.8.2.9 RESETCTL Register (Offset = 28h) [Reset = 000001C0h]

RESETCTL is shown in Table 7-97.

Return to the Summary Table.

Reset Management
This register contains bitfields related to system reset such as reset source and reset request and control of brown out resets.

Table 7-97. RESETCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SYSRESET	W	0h	Cold reset register. Writing 1 to this bitfield will reset the entire chip and cause boot code to run again. 0: No effect 1: Generate system reset. Appears as SYSRESET in RESET_SRC
30-26	RESERVED	R	0h	Reserved
25	BOOT_DET_1_CLR	W	0h	Internal. Only to be used through TI provided API.
24	BOOT_DET_0_CLR	W	0h	Internal. Only to be used through TI provided API.
23-18	RESERVED	R	0h	Reserved
17	BOOT_DET_1_SET	W	0h	Internal. Only to be used through TI provided API.
16	BOOT_DET_0_SET	W	0h	Internal. Only to be used through TI provided API.
15	WU_FROM_SD	R	0h	A Wakeup from SHUTDOWN on an IO event has occurred, or a wakeup from SHUTDOWN has occurred as a result of the debugger being attached.. (TCK pin being forced low) Please refer to IOC:IOCFGn.WU_CFG for configuring the IO's as wakeup sources. 0: Wakeup occurred from cold reset or brown out as seen in RESET_SRC 1: A wakeup has occurred from SHUTDOWN Note: This flag will be cleared when SLEEPCTL.IO_PAD_SLEEP_DIS is asserted.
14	GPIO_WU_FROM_SD	R	0h	A wakeup from SHUTDOWN on an IO event has occurred Please refer to IOC:IOCFGn.WU_CFG for configuring the IO's as wakeup sources. 0: The wakeup did not occur from SHUTDOWN on an IO event 1: A wakeup from SHUTDOWN occurred from an IO event The case where WU_FROM_SD is asserted but this bitfield is not asserted will only occur in a debug session. The boot code will not proceed with wakeup from SHUTDOWN procedure until this bitfield is asserted as well. Note: This flag will be cleared when SLEEPCTL.IO_PAD_SLEEP_DIS is asserted.
13	BOOT_DET_1	R	0h	Internal. Only to be used through TI provided API.
12	BOOT_DET_0	R	0h	Internal. Only to be used through TI provided API.
11-9	RESERVED	R	0h	Reserved
8	VDDS_LOSS_EN	R/W	1h	Internal. Only to be used through TI provided API.
7	VDDR_LOSS_EN	R/W	1h	Internal. Only to be used through TI provided API.
6	VDD_LOSS_EN	R/W	1h	Internal. Only to be used through TI provided API.
5	CLK_LOSS_EN	R/W	0h	Controls reset generation in case SCLK_LF, SCLK_MF or SCLK_HF is lost when clock loss detection is enabled by [ANATOP_MMAP:DDI_0_OSC:CTL0.CLK_LOSS_EN] 0: Clock loss is ignored 1: Clock loss generates system reset Note: Clock loss reset generation must be disabled when changing clock source for SCLK_LF. Failure to do so may result in a spurious system reset. Clock loss reset generation is controlled by [ANATOP_MMAP:DDI_0_OSC:CTL0.CLK_LOSS_EN]
4	MCU_WARM_RESET	R/W1C	0h	Internal. Only to be used through TI provided API.

Table 7-97. RESETCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-1	RESET_SRC	R	0h	Shows the root cause of the last system reset. More than the reported reset source can have been active during the last system reset but only the root cause is reported. The capture feature is not rearmed until all off the possible reset sources have been released and the result has been copied to AON_PMCTL. During the copy and rearm process it is one 2MHz period in which and eventual new system reset will be reported as Power on reset regardless of the root cause. 0h = Power on reset 1h = Reset pin 2h = Brown out detect on VDDS 4h = Brown out detect on VDDR 5h = SCLK_LF, SCLK_MF or SCLK_HF clock loss detect 6h = Software reset via SYSRESET or hardware power management timeout detection. Note: The hardware power management timeout circuit is always enabled. 7h = Software reset via PRCM warm reset request
0	RESERVED	R	0h	Reserved

7.8.2.10 SLEEPCTL Register (Offset = 2Ch) [Reset = 00000000h]

SLEEPCTL is shown in Table 7-98.

Return to the Summary Table.

Sleep Control
This register is used to unfreeze the IO pad ring after waking up from SHUTDOWN

Table 7-98. SLEEPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	IO_PAD_SLEEP_DIS	R/W	0h	Controls the I/O pad sleep mode. The boot code will set this bitfield automatically unless waking up from a SHUTDOWN (RESETCTL.WU_FROM_SD is set). 0: I/O pad sleep mode is enabled, meaning all outputs and pad configurations are latched. Inputs are transparent if pad is configured as input before IO_PAD_SLEEP_DIS is set to 1 1: I/O pad sleep mode is disabled Application software must reconfigure the state for all IO's before setting this bitfield upon waking up from a SHUTDOWN to avoid glitches on pins.

7.8.2.11 JTAGCFG Register (Offset = 34h) [Reset = 00000100h]

JTAGCFG is shown in Table 7-99.

Return to the Summary Table.

JTAG Configuration
This register contains control for configuration of the JTAG domain. This includes permissions for each TAP.

Table 7-99. JTAGCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	JTAG_PD_FORCE_ON	R/W	1h	Controls JTAG Power domain power state: 0: Controlled exclusively by debug subsystem. (JTAG Power domain will be powered off unless a debugger is attached) 1: JTAG Power Domain is forced on, independent of debug subsystem. Note: The reset value causes JTAG Power domain to be powered on by default. Software must clear this bit to turn off the JTAG Power domain
7-0	RESERVED	R	0h	Reserved

7.8.2.12 JTAGUSERCODE Register (Offset = 3Ch) [Reset = 0B99A02Fh]

JTAGUSERCODE is shown in Table 7-100.

Return to the Summary Table.

JTAG USERCODE
Boot code copies the JTAG USERCODE to this register from where it is forwarded to the debug subsystem.

Table 7-100. JTAGUSERCODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	USER_CODE	R/W	0B99A02Fh	32-bit JTAG USERCODE register feeding main JTAG TAP Note: This field can be locked by LOCKCFG.LOCK

7.8.2.13 WDTLOAD Register (Offset = C4h) [Reset = 00000000h]

WDTLOAD is shown in Table 7-101.

Return to the Summary Table.

Configuration
Load Value register

Table 7-101. WDTLOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LOAD	R/W	0h	This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter is restarted to count down from the new value. If this register is loaded with 0x0000.0000, a reset is immediately generated. Read from this register will return the current value of the counter

7.8.2.14 WDTTEST Register (Offset = C8h) [Reset = 00000000h]

WDTTEST is shown in Table 7-102.

Return to the Summary Table.

Test Mode

Table 7-102. WDTTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STALLEN	R/W	0h	WDT Stall Enable 0: The WDT timer continues counting if the CPU is stopped with a debugger. 1: If the CPU is stopped with a debugger, the WDT stops counting. Once the CPU is restarted, the WDT resumes counting.

7.8.2.15 WDTLOCK Register (Offset = D0h) [Reset = 00000000h]

WDTLOCK is shown in Table 7-103.

Return to the Summary Table.

Lock

Table 7-103. WDTLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LOCK	R/W	0h	WDT Lock: A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates A read of this register returns the following values: 0x0000.0000: Unlocked 0x0000.0001: Locked

7.8.3 DDI_0_OSC Registers

Table 7-104 lists the memory-mapped registers for the DDI_0_OSC registers. All register offset addresses not listed in Table 7-104 should be considered as reserved locations and the register contents should not be modified.

Table 7-104. DDI_0_OSC Registers

Offset	Acronym	Register Name	Section
0h	CTL0	Control 0	Section 7.8.3.1
4h	CTL1	Control 1	Section 7.8.3.2
8h	RADCEXTCFG	RADC External Configuration	Section 7.8.3.3
Ch	AMPCOMPCTL	Amplitude Compensation Control	Section 7.8.3.4
10h	AMPCOMPTH1	Amplitude Compensation Threshold 1	Section 7.8.3.5
14h	AMPCOMPTH2	Amplitude Compensation Threshold 2	Section 7.8.3.6
18h	ANABYPASSVAL1	Analog Bypass Values 1	Section 7.8.3.7
1Ch	ANABYPASSVAL2	Internal	Section 7.8.3.8
20h	ATESTCTL	Analog Test Control	Section 7.8.3.9
24h	ADCDOUBLERNANOAMPCTL	ADC Doubler Nanoamp Control	Section 7.8.3.10
28h	XOSCHFCTL	XOSCHF Control	Section 7.8.3.11
2Ch	LFOSCCTL	Low Frequency Oscillator Control	Section 7.8.3.12
30h	RCOSCHFCTL	RCOSCHF Control	Section 7.8.3.13
34h	RCOSCMFCTL	RCOSC_MF Control	Section 7.8.3.14
3Ch	STAT0	Status 0	Section 7.8.3.15
40h	STAT1	Status 1	Section 7.8.3.16
44h	STAT2	Status 2	Section 7.8.3.17

Complex bit access types are encoded to fit into small table cells. Table 7-105 shows the codes that are used for access types in this section.

Table 7-105. DDI_0_OSC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

7.8.3.1 CTL0 Register (Offset = 0h) [Reset = 00000000h]

CTL0 is shown in Table 7-106.

Return to the Summary Table.

Control 0
Controls clock source selects

Table 7-106. CTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	XTAL_IS_24M	R/W	0h	Set based on the accurate high frequency XTAL.
30	RESERVED	R	0h	Reserved
29	BYPASS_XOSC_LF_CLK_QUAL	R/W	0h	Internal. Only to be used through TI provided API.
28	BYPASS_RCOSC_LF_CLK_QUAL	R/W	0h	Internal. Only to be used through TI provided API.
27-26	DOUBLER_START_DURATION	R/W	0h	Internal. Only to be used through TI provided API.
25	DOUBLER_RESET_DURATION	R/W	0h	Internal. Only to be used through TI provided API.
24	CLK_DCDC_SRC_SEL	R/W	0h	Select DCDC clock source. 0: CLK_DCDC is 48 MHz clock from RCOSC or XOSC / HPOSC 1: CLK_DCDC is always 48 MHz clock from RCOSC
23-15	RESERVED	R	0h	Reserved
14	HPOSC_MODE_EN	R/W	0h	0: HPOSC mode is not enabled. The 48 MHz crystal is required for radio operation. 1: Enables HPOSC mode. The internal HPOSC can be used as HF system clock and for radio operation.
13	RESERVED	R	0h	Reserved
12	RCOSC_LF_TRIMMED	R/W	0h	Internal. Only to be used through TI provided API.
11	XOSC_HF_POWER_MODE	R/W	0h	Internal. Only to be used through TI provided API.
10	XOSC_LF_DIG_BYPASS	R/W	0h	Bypass XOSC_LF and use the digital input clock from AON for the xosc_lf clock. 0: Use 32kHz XOSC as xosc_lf clock source 1: Use digital input (from AON) as xosc_lf clock source. This bit will only have effect when SCLK_LF_SRC_SEL is selecting the xosc_lf as the sclk_lf source. The muxing performed by this bit is not glitch free. The following procedure must be followed when changing this field to avoid glitches on sclk_lf. 1) Set SCLK_LF_SRC_SEL to select any source other than the xosc_lf clock source. 2) Set or clear this bit to bypass or not bypass the xosc_lf. 3) Set SCLK_LF_SRC_SEL to use xosc_lf. It is recommended that either the rcosc_hf or xosc_hf (whichever is currently active) be selected as the source in step 1 above. This provides a faster clock change.
9	CLK_LOSS_EN	R/W	0h	Enable clock loss detection and hence the indicators to the system controller. Checks both SCLK_HF, SCLK_MF and SCLK_LF clock loss indicators. 0: Disable 1: Enable Clock loss detection must be disabled when changing the sclk_lf source. STAT0.SCLK_LF_SRC can be polled to determine when a change to a new sclk_lf source has completed.

Table 7-106. CTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8-7	ACLK_TDC_SRC_SEL	R/W	0h	Source select for aclk_tdc. 00: RCOSC_HF (48MHz) 01: RCOSC_HF (24MHz) 10: XOSC_HF (24MHz) 11: Not used
6-4	ACLK_REF_SRC_SEL	R/W	0h	Source select for aclk_ref 000: RCOSC_HF derived (31.25kHz) 001: XOSC_HF derived (31.25kHz) 010: RCOSC_LF (32kHz) 011: XOSC_LF (32.768kHz) 100: RCOSC_MF (2MHz) 101-111: Not used
3-2	SCLK_LF_SRC_SEL	R/W	0h	Source select for sclk_lf 0h = Low frequency clock derived from High Frequency RCOSC 1h = Low frequency clock derived from High Frequency XOSC or HPOSC clk (use HPOSC when HPOSC_MODE_EN = 1) 2h = Low frequency RCOSC 3h = Low frequency XOSC
1	RESERVED	R	0h	Reserved
0	SCLK_HF_SRC_SEL	R/W	0h	Source select for sclk_hf. 0h = High frequency RCOSC clock 1h = High frequency XOSC or HPOSC clk (use HPOSC when HPOSC_MODE_EN = 1

7.8.3.2 CTL1 Register (Offset = 4h) [Reset = 00000000h]

CTL1 is shown in Table 7-107.

Return to the Summary Table.

Control 1
This register contains OSC_DIG configuration

Table 7-107. CTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22-18	RCOSCHFCTRIMFRACT	R/W	0h	Internal. Only to be used through TI provided API.
17	RCOSCHFCTRIMFRACT_EN	R/W	0h	Internal. Only to be used through TI provided API.
16-10	RESERVED	R	0h	Reserved
9	FORCE_RCOSC_LF	R/W	0h	Force rcosc_lf to be enabled 0: Disabled 1: Enabled
8	CLK_LF_LOSS_EN	R/W	0h	Enable LF clock loss detection and hence the indicators to the system controller. Checks SCLK_LF clock loss indicators. 0: Disable 1: Enable Clock loss detection must be disabled when changing the sclk_lf source. STAT0.SCLK_LF_SRC can be polled to determine when a change to a new sclk_lf source has completed.
7-2	RESERVED	R	0h	Reserved
1-0	XOSC_HF_FAST_START	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.3 RADCEXTCFG Register (Offset = 8h) [Reset = 00000000h]

RADCEXTCFG is shown in Table 7-108.

Return to the Summary Table.

RADC External Configuration

Table 7-108. RADCEXTCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	HPM_IBIAS_WAIT_CNT	R/W	0h	Internal. Only to be used through TI provided API.
21-16	LPM_IBIAS_WAIT_CNT	R/W	0h	Internal. Only to be used through TI provided API.
15-12	IDAC_STEP	R/W	0h	Internal. Only to be used through TI provided API.
11-6	RADC_DAC_TH	R/W	0h	Internal. Only to be used through TI provided API.
5	RADC_MODE_IS_SAR	R/W	0h	Internal. Only to be used through TI provided API.
4-0	RESERVED	R	0h	Reserved

7.8.3.4 AMPCOMPCTL Register (Offset = Ch) [Reset = 00000000h]

AMPCOMPCTL is shown in Table 7-109.

Return to the Summary Table.

Amplitude Compensation Control

Table 7-109. AMPCOMPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	AMPCOMP_REQ_MODE	R/W	0h	Internal. Only to be used through TI provided API.
29-28	AMPCOMP_FSM_UPDATE_RATE	R/W	0h	Internal. Only to be used through TI provided API.
27	AMPCOMP_SW_CTRL	R/W	0h	Internal. Only to be used through TI provided API.
26	AMPCOMP_SW_EN	R/W	0h	Internal. Only to be used through TI provided API.
25-24	RESERVED	R	0h	Reserved
23-20	IBIAS_OFFSET	R/W	0h	Internal. Only to be used through TI provided API.
19-16	IBIAS_INIT	R/W	0h	Internal. Only to be used through TI provided API.
15-8	LPM_IBIAS_WAIT_CNT_FINAL	R/W	0h	Internal. Only to be used through TI provided API.
7-4	CAP_STEP	R/W	0h	Internal. Only to be used through TI provided API.
3-0	IBIASCAP_HPTOLP_OL_CNT	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.5 AMPCOMPTH1 Register (Offset = 10h) [Reset = 00000000h]

AMPCOMPTH1 is shown in Table 7-110.

Return to the Summary Table.

Amplitude Compensation Threshold 1
This register contains threshold values for amplitude compensation algorithm

Table 7-110. AMPCOMPTH1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-18	HPMRAMP3_LTH	R/W	0h	Internal. Only to be used through TI provided API.
17-16	RESERVED	R	0h	Reserved
15-10	HPMRAMP3_HTH	R/W	0h	Internal. Only to be used through TI provided API.
9-6	IBIASCAP_LPTOHP_OL_CNT	R/W	0h	Internal. Only to be used through TI provided API.
5-0	HPMRAMP1_TH	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.6 AMPCOMPTH2 Register (Offset = 14h) [Reset = 00000000h]

AMPCOMPTH2 is shown in Table 7-111.

Return to the Summary Table.

Amplitude Compensation Threshold 2
This register contains threshold values for amplitude compensation algorithm.

Table 7-111. AMPCOMPTH2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	LPMUPDATE_LTH	R/W	0h	Internal. Only to be used through TI provided API.
25-24	RESERVED	R	0h	Reserved
23-18	LPMUPDATE_HTH	R/W	0h	Internal. Only to be used through TI provided API.
17-16	RESERVED	R	0h	Reserved
15-10	ADC_COMP_AMPTH_LPM	R/W	0h	Internal. Only to be used through TI provided API.
9-8	RESERVED	R	0h	Reserved
7-2	ADC_COMP_AMPTH_HPM	R/W	0h	Internal. Only to be used through TI provided API.
1-0	RESERVED	R	0h	Reserved

7.8.3.7 ANABYPASSVAL1 Register (Offset = 18h) [Reset = 00000000h]

ANABYPASSVAL1 is shown in Table 7-112.

Return to the Summary Table.

Analog Bypass Values 1

Table 7-112. ANABYPASSVAL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-16	XOSC_HF_ROW_Q12	R/W	0h	Internal. Only to be used through TI provided API.
15-0	XOSC_HF_COLUMN_Q12	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.8 ANABYPASSVAL2 Register (Offset = 1Ch) [Reset = 00000000h]

ANABYPASSVAL2 is shown in Table 7-113.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 7-113. ANABYPASSVAL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13-0	XOSC_HF_IBIASTHERM	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.9 ATESTCTL Register (Offset = 20h) [Reset = 00000000h]

ATESTCTL is shown in Table 7-114.

Return to the Summary Table.

Analog Test Control

Table 7-114. ATESTCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SCLK_LF_AUX_EN	R/W	0h	Enable 32 kHz clock to AUX_COMPB.
30-16	RESERVED	R	0h	Reserved
15-14	TEST_RCOSCMF	R/W	0h	Test mode control for RCOSC_MF 0x0: test modes disabled 0x1: boosted bias current into self biased inverter 0x2: clock qualification disabled 0x3: boosted bias current into self biased inverter + clock qualification disabled
13-12	ATEST_RCOSCMF	R/W	0h	ATEST control for RCOSC_MF 0x0: ATEST disabled 0x1: ATEST enabled, VDD_LOCAL connected, ATEST internal to **RCOSC_MF* enabled to send out 2MHz clock. 0x2: ATEST disabled 0x3: ATEST enabled, bias current connected, ATEST internal to **RCOSC_MF* enabled to send out 2MHz clock.
11-0	RESERVED	R	0h	Reserved

7.8.3.10 ADCDOUBLERNANOAMPCTL Register (Offset = 24h) [Reset = 00000000h]

ADCDOUBLERNANOAMPCTL is shown in Table 7-115.

Return to the Summary Table.

ADC Doubler Nanoamp Control

Table 7-115. ADCDOUBLERNANOAMPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	NANOAMP_BIAS_ENABLE	R/W	0h	Internal. Only to be used through TI provided API.
23	SPARE23	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior
22-6	RESERVED	R	0h	Reserved
5	ADC_SH_MODE_EN	R/W	0h	Internal. Only to be used through TI provided API.
4	ADC_SH_VBUF_EN	R/W	0h	Internal. Only to be used through TI provided API.
3-2	RESERVED	R	0h	Reserved
1-0	ADC_IREF_CTRL	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.11 XOSCHFCTL Register (Offset = 28h) [Reset = 00000000h]

XOSCHFCTL is shown in Table 7-116.

Return to the Summary Table.

XOSCHF Control

Table 7-116. XOSCHFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	TCXO_MODE_XOSC_HF_EN	R/W	0h	If this register is 1 when TCXO_MODE is 1, then the XOSC_HF is enabled, turning on the XOSC_HF bias current allowing a DC bias point to be provided to the clipped-sine wave clock signal on external input.
12	TCXO_MODE	R/W	0h	If this register is 1 when BYPASS is 1, this will enable clock qualification on the TCXO clock on external input. This register has no effect when BYPASS is 0.
11-10	RESERVED	R	0h	Reserved
9-8	PEAK_DET_ITRIM	R/W	0h	Internal. Only to be used through TI provided API.
7	RESERVED	R	0h	Reserved
6	BYPASS	R/W	0h	Internal. Only to be used through TI provided API.
5	RESERVED	R	0h	Reserved
4-2	HP_BUF_ITRIM	R/W	0h	Internal. Only to be used through TI provided API.
1-0	LP_BUF_ITRIM	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.12 LFOSCCTL Register (Offset = 2Ch) [Reset = 00000000h]

LFOSCCTL is shown in Table 7-117.

Return to the Summary Table.

Low Frequency Oscillator Control

Table 7-117. LFOSCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-22	XOSCLF_REGULATOR_TRIM	R/W	0h	Internal. Only to be used through TI provided API.
21-18	XOSCLF_CMIRRWR_RATIO	R/W	0h	Internal. Only to be used through TI provided API.
17-10	RESERVED	R	0h	Reserved
9-8	RCOSCLF_RTUNE_TRIM	R/W	0h	Internal. Only to be used through TI provided API.
7-0	RCOSCLF_CTUNE_TRIM	R/W	0h	Internal. Only to be used through TI provided API.

7.8.3.13 RCOSCHFCTL Register (Offset = 30h) [Reset = 00000000h]

RCOSCHFCTL is shown in Table 7-118.

Return to the Summary Table.

RCOSCHF Control

Table 7-118. RCOSCHFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RCOSCHF_CTRIM	R/W	0h	Internal. Only to be used through TI provided API.
7-0	RESERVED	R	0h	Reserved

7.8.3.14 RCOSCMFCTL Register (Offset = 34h) [Reset = 00000000h]

RCOSCMFCTL is shown in Table 7-119.

Return to the Summary Table.

RCOSC_MF Control

Table 7-119. RCOSCMFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-9	RCOSC_MF_CAP_ARRAY	R/W	0h	Adjust RCOSC_MF capacitor array. 0x0: nominal frequency, 0.625pF 0x40: highest frequency, 0.125pF 0x3F: lowest frequency, 1.125pF
8	RCOSC_MF_REG_SEL	R/W	0h	Choose regulator type. 0: default 1: alternate
7-6	RCOSC_MF_RES_COARSE	R/W	0h	Select coarse resistor for frequency adjustment. 0x0: 400kΩs, default 0x1: 300kΩs, min 0x2: 600kΩs, max 0x3: 500kΩs
5-4	RCOSC_MF_RES_FINE	R/W	0h	Select fine resistor for frequency adjustment. 0x0: 11kΩs, minimum resistance, max freq 0x1: 13kΩs 0x2: 16kΩs 0x3: 20kΩs, max resistance, min freq
3-0	RCOSC_MF_BIAS_ADJ	R/W	0h	Adjusts bias current to RCOSC_MF. The trim has binary encoding with MSB inverted. 0x5 Minimum current 0x6 Default -10 0x7 Default -9 . . . 0xF Default -1 0x0 Default current 0x1 Default +1 0x2 Default +2 0x3 Default +3 0x4 Maximum current

7.8.3.15 STAT0 Register (Offset = 3Ch) [Reset = 00000000h]

STAT0 is shown in Table 7-120.

Return to the Summary Table.

Status 0
This register contains status signals from OSC_DIG

Table 7-120. STAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RCOSC_LF_GOOD	R	0h	RCOSC_LF_GOOD
30-29	SCLK_LF_SRC	R	0h	Indicates source for the sclk_lf 0h = Low frequency clock derived from High Frequency RCOSC 1h = Low frequency clock derived from High Frequency XOSC 2h = Low frequency RCOSC 3h = Low frequency XOSC
28	SCLK_HF_SRC	R	0h	Indicates source for the sclk_hf 0h = High frequency RCOSC clock 1h = High frequency XOSC
27-23	RESERVED	R	0h	Reserved
22	RCOSC_HF_EN	R	0h	RCOSC_HF_EN
21	RCOSC_LF_EN	R	0h	RCOSC_LF_EN
20	XOSC_LF_EN	R	0h	XOSC_LF_EN
19	CLK_DCDC_RDY	R	0h	CLK_DCDC_RDY
18	CLK_DCDC_RDY_ACK	R	0h	CLK_DCDC_RDY_ACK
17	SCLK_HF_LOSS	R	0h	Indicates sclk_hf is lost
16	SCLK_LF_LOSS	R	0h	Indicates sclk_lf is lost
15	XOSC_HF_EN	R	0h	Indicates that XOSC_HF is enabled.
14	RESERVED	R	0h	Reserved
13	XB_48M_CLK_EN	R	0h	Indicates that the 48MHz clock from the DOUBLER is enabled. It will be enabled if 24 or 48 MHz crystal is used (enabled in doubler bypass for the 48MHz crystal).
12	RESERVED	R	0h	Reserved
11	XOSC_HF_LP_BUF_EN	R	0h	XOSC_HF_LP_BUF_EN
10	XOSC_HF_HP_BUF_EN	R	0h	XOSC_HF_HP_BUF_EN
9	RESERVED	R	0h	Reserved
8	ADC_THMET	R	0h	ADC_THMET
7	ADC_DATA_READY	R	0h	indicates when adc_data is ready.
6-1	ADC_DATA	R	0h	adc_data
0	PENDINGSCLKHFSWITCHING	R	0h	Indicates when SCLK_HF clock source is ready to be switched

7.8.3.16 STAT1 Register (Offset = 40h) [Reset = 00000000h]

STAT1 is shown in Table 7-121.

Return to the Summary Table.

Status 1
This register contains status signals from OSC_DIG

Table 7-121. STAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RAMPSTATE	R	0h	AMPCOMP FSM State 0h = RESET 1h = INITIALIZATION 2h = HPM_RAMP1 3h = HPM_RAMP2 4h = HPM_RAMP3 5h = HPM_UPDATE 6h = IDAC_INCREMENT 7h = IBIAS_CAP_UPDATE 8h = IBIAS_DECREMENT_WITH_MEASURE 9h = LPM_UPDATE Ah = IBIAS_INCREMENT Bh = IDAC_DECREMENT_WITH_MEASURE Ch = DUMMY_TO_INIT_1 Dh = FAST_START Eh = FAST_START_SETTLE
27-22	HPM_UPDATE_AMP	R	0h	XOSC_HF amplitude during HPM_UPDATE state. When amplitude compensation of XOSC_HF is enabled in high performance mode, this value is the amplitude of the crystal oscillations measured by the on-chip oscillator ADC, divided by 15 mV. For example, a value of 0x20 would indicate that the amplitude of the crystal is approximately 480 mV. To enable amplitude compensation, AON_WUC OSCCFG must be set to a non-zero value.
21-16	LPM_UPDATE_AMP	R	0h	XOSC_HF amplitude during LPM_UPDATE state When amplitude compensation of XOSC_HF is enabled in low power mode, this value is the amplitude of the crystal oscillations measured by the on-chip oscillator ADC, divided by 15 mV. For example, a value of 0x20 would indicate that the amplitude of the crystal is approximately 480 mV. To enable amplitude compensation, AON_WUC OSCCFG must be set to a non-zero value.
15	FORCE_RCOSC_HF	R	0h	force_rcosc_hf
14	SCLK_HF_EN	R	0h	SCLK_HF_EN
13	SCLK_MF_EN	R	0h	SCLK_MF_EN
12	ACLK_ADC_EN	R	0h	ACLK_ADC_EN
11	ACLK_TDC_EN	R	0h	ACLK_TDC_EN
10	ACLK_REF_EN	R	0h	ACLK_REF_EN
9	CLK_CHP_EN	R	0h	CLK_CHP_EN
8	CLK_DCDC_EN	R	0h	CLK_DCDC_EN
7	SCLK_HF_GOOD	R	0h	SCLK_HF_GOOD
6	SCLK_MF_GOOD	R	0h	SCLK_MF_GOOD
5	SCLK_LF_GOOD	R	0h	SCLK_LF_GOOD
4	ACLK_ADC_GOOD	R	0h	ACLK_ADC_GOOD
3	ACLK_TDC_GOOD	R	0h	ACLK_TDC_GOOD
2	ACLK_REF_GOOD	R	0h	ACLK_REF_GOOD.
1	CLK_CHP_GOOD	R	0h	CLK_CHP_GOOD
0	CLK_DCDC_GOOD	R	0h	CLK_DCDC_GOOD

7.8.3.17 STAT2 Register (Offset = 44h) [Reset = 00000000h]

STAT2 is shown in Table 7-122.

Return to the Summary Table.

Status 2
This register contains status signals from AMPCOMP FSM

Table 7-122. STAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	ADC_DCBIAS	R	0h	DC Bias read by RADC during SAR mode The value is an unsigned integer. It is used for debug only.
25	HPM_RAMP1_THMET	R	0h	Indication of threshold is met for hpm_ramp1
24	HPM_RAMP2_THMET	R	0h	Indication of threshold is met for hpm_ramp2
23	HPM_RAMP3_THMET	R	0h	Indication of threshold is met for hpm_ramp3
22-16	RESERVED	R	0h	Reserved
15-12	RAMPSTATE	R	0h	xosc_hf amplitude compensation FSM This is identical to STAT1.RAMPSTATE. See that description for encoding.
11-4	RESERVED	R	0h	Reserved
3	AMPCOMP_REQ	R	0h	ampcomp_req
2	XOSC_HF_AMPGOOD	R	0h	amplitude of xosc_hf is within the required threshold (set by DDI). Not used for anything just for debug/status
1	XOSC_HF_FREQGOOD	R	0h	frequency of xosc_hf is good to use for the digital clocks
0	XOSC_HF_RF_FREQGOOD	R	0h	frequency of xosc_hf is within +/- 20 ppm and xosc_hf is good for radio operations. Used for SW to start synthesizer.

Chapter 8

Versatile Instruction Memory System (VIMS)

This chapter discusses the Versatile Instruction Memory System (VIMS) of the CC13x4x10 and CC26x4x10 device platform.

8.1 Introduction 616
8.2 VIMS Configurations 617
8.3 VIMS Software Remarks 619
8.4 FLASH 621
8.5 ROM Functions 623
8.6 VIMS Registers 623

8.1 Introduction

The main instruction memories are encapsulated in a versatile instruction memory system (VIMS) module, which includes the following memories:

- 1 MB of FLASH
- 8 KB of RAM Cache or general-purpose RAM (GPRAM)
- 40 KB of Boot ROM

Figure 8-1 shows an overview of the VIMS module.

The VIMS module forwards CPU accesses (icode/dcode) and system bus accesses to the addressed memories. The VIMS module also arbitrates access between the CPU and the system bus.

The VIMS module runs on the 48 MHz system clock.

The FLASH memory is programmable from user software, from the debug interface, and from the ROM bootloader. The RAM block can be used as a cache for the FLASH block or as GPRAM.

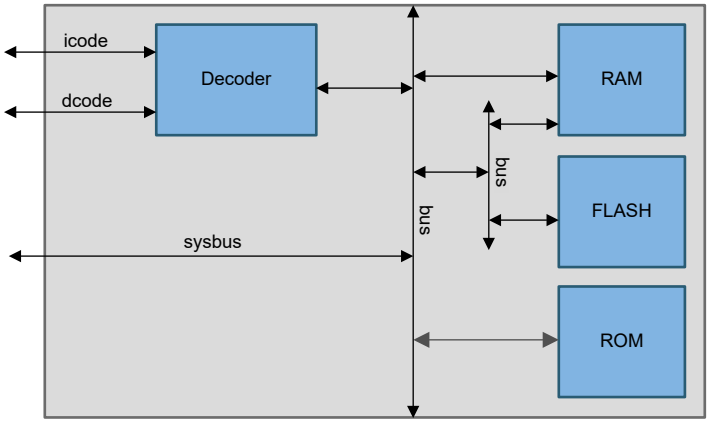


Figure 8-1. VIMS Overview

8.2 VIMS Configurations

8.2.1 VIMS Modes

The RAM block operates as follows:

- GPRAM
- CACHE
- OFF

The current mode is shown in the VIMS:STAT.MODE register, and mode switching is controlled through the VIMS:CTL.MODE register. Figure 8-2 shows the mode transitions. All mode changes are software initiated. The invalidating state is a transition state controlled by hardware. Invalidation initializes the entire content of the RAM block and takes 1029 clock periods to perform.

Once a mode change is initiated, shown in the VIMS:STATUS.MODE_CHANGING register, the mode change must complete before another mode change can be initiated. The VIMS:CTL.MODE register is blocked for updates during a mode change.

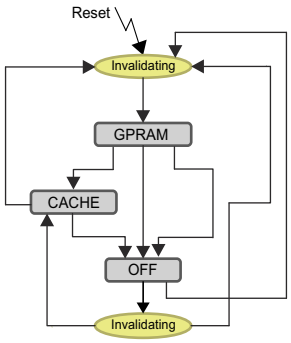


Figure 8-2. VIMS Mode Switching Flowchart

8.2.1.1 GPRAM Mode

In GPRAM mode, the RAM block functions as a general-purpose RAM (see Figure 8-3). The FLASH block has no cache support, and all accesses to the FLASH are routed directly to the FLASH block.

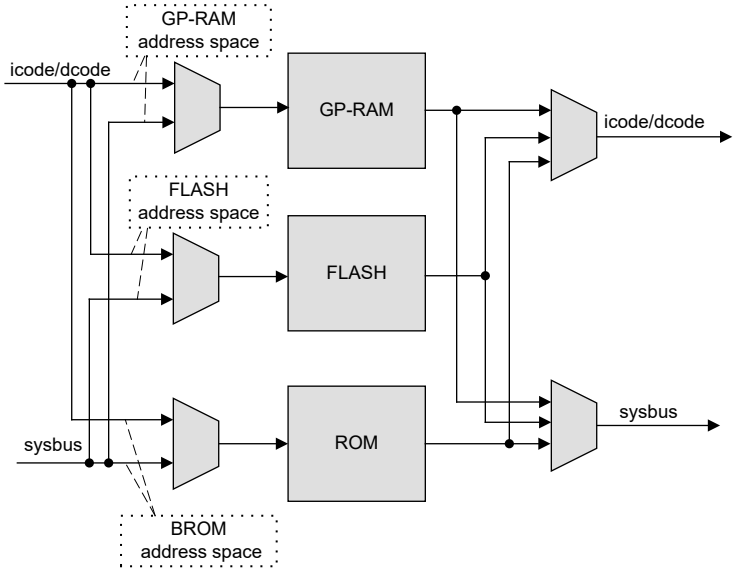


Figure 8-3. VIMS Module in GPRAM Mode

8.2.1.2 Off Mode

In off mode, the RAM block is disabled and cannot be accessed by the CPU or by the system bus (see Figure 8-4). The GPRAM space is not available in off mode.

The FLASH block has no cache support, and all accesses to the FLASH are routed directly to the FLASH block.

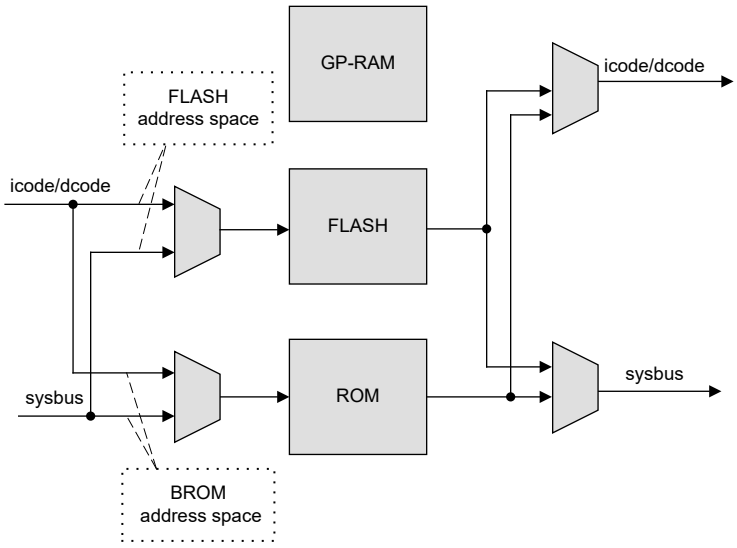


Figure 8-4. VIMS Module in Off Mode

8.2.1.3 Cache Mode

In cache mode, the RAM block functions as an 8K 4-way random replacement cache for the FLASH block (see Figure 8-5). The GPRAM space is not available in cache mode.

The cache support is only available for CPU accesses to the FLASH block. System bus accesses to the FLASH are routed directly to the FLASH block.

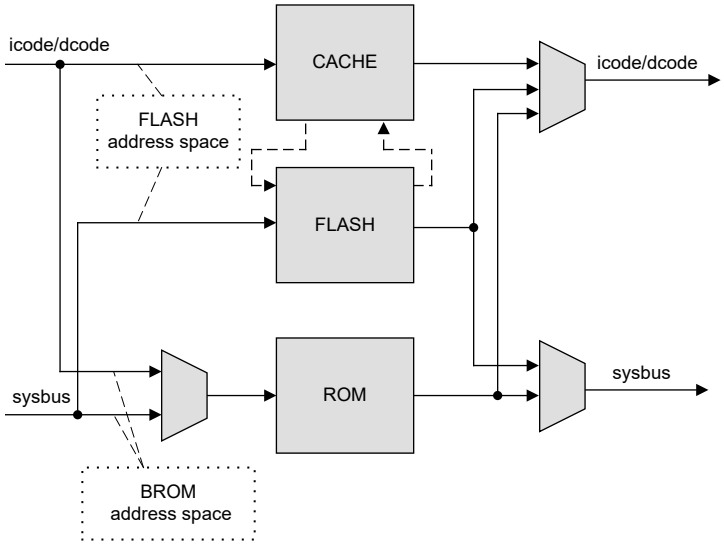


Figure 8-5. VIMS Module in Cache Mode

In cache mode, all CPU accesses to the FLASH block are directed to the cache first. The cache looks up the input address in the internal tag RAM to determine whether the access is a cache hit or a cache miss.

In the case of a cache miss, the access is forwarded to the FLASH block. The response from the FLASH block is routed back to the cache, then the cache is updated.

In the case of a cache hit, the data is fetched directly from the cache RAM.

The cache also contains a line buffer because the cache RAM word size is 64 bits. The objective of the line buffer is to prevent refetching the 32-bit part of the data that has already been fetched (but not used) in the previous access. The line buffer prevents both TAG and CACHE lookup if the data is already in the line buffer.

The cache line buffer is cleared as a part of the invalidation scheme.

8.2.2 VIMS FLASH Line Buffers

The VIMS module contains two FLASH line buffers because the FLASH word size is 128 bits.

- A line buffer is placed in the FLASH CPU bus path that is controlled by the VIMS:CTL.IDCODE_LB_DIS register.
- A line buffer is placed in the FLASH system bus path that is controlled by the VIMS:CTL.SYSBUS_LB_DIS register.

The objectives of the buffers are to prevent refetching the 32-bit part of the data that has already been fetched (but not used) in a previous cycle. The status of the line buffers can be found in the VIMS:STATUS.IDCODE_LB_DIS register and the VIMS:STATUS.SYSBUS_LB_DIS register.

8.2.3 VIMS Arbitration

The VIMS provides arbitration between the CPU and the system bus. The arbitration is configurable between *round-robin* and *static*, through the VIMS:CTL.ARB_CFG register. The static arbitration is enabled by default and gives the CPU priority over the system bus.

The system arbiter allows accesses to occur simultaneously, provided that the CPU and the system bus have different target memories. If, for example, a CPU access causes a cache hit, a system bus access can access the FLASH simultaneously.

8.2.4 VIMS Cache TAG Prefetch

The cache contains a TAG prefetch system that automatically prefetches the TAG data for the next 64-bit address. This feature is controlled through the VIMS:CTL.PREF_EN register, and is only enabled if the VIMS mode is set to cache mode. Any access using a prefetched TAG saves one CLK cycle in the access because tag lookup can be skipped. A prefetch hit is defined as an access using prefetched TAG data and data that is available in the cache.

TAG prefetch is mainly intended for performance optimization when the CPU is running at full speed. If the CPU is not running at full speed, there is no performance optimization; therefore the TAG prefetch system should be disabled to minimize power consumption.

8.3 VIMS Software Remarks

When the FLASH is programmed or updated, or when the VIMS domain is entering power down special care must be taken from the software side.

The following remarks are automatically taken care of when using in-built ROM functions and the standard API functions. However, custom code must take the following remarks into account.

8.3.1 FLASH Program or Update

Do not program or update the FLASH when the VIMS is in cache mode.

Before you program or update the FLASH, complete the following tasks:

- Set VIMS in off mode if the current mode is cache mode.
- Disable both of the VIMS FLASH line buffers.

These actions prevent old data or instructions to be fetched from the cache or the line buffers after a FLASH program or update.

After the FLASH program or update, the VIMS can be set back to cache mode. Hardware will invalidate the cache when VIMS is set back to cache mode.

8.3.2 VIMS Retention

The VIMS domain can be kept in retention, if needed, when the domain is entering power down. The retention control has the option to specify which memories (internal TAG RAM or cache RAM) are kept in retention together with VIMS logic.

Table 8-1 specifies the valid retention combination for VIMS memory.

Table 8-1. Valid Retention Combination for VIMS Memory

Mode	Retention Enabled			Comments
	TAG-RAM	CACHE-RAM	VIMS Logic	
1	No	No	Yes	Software must compensate for loss of data in RAMs
2	No	Yes	Yes	Works in GPRAM mode without software intervention
3	Yes	Yes	Yes	

8.3.2.1 Mode 1

Mode 1 is intended for use when VIMS is in off mode or cache mode.

If VIMS is in cache mode, change the VIMS mode to off mode before powering down the VIMS domain. When the system is taken out of power down, you can set the VIMS mode back to cache mode, which invalidates (initializes) the cache memories (see Figure 8-6).

Mode 1 can also be used when the system is in GPRAM mode, but all data in the GPRAM is lost when the VIMS domain is set in power down.

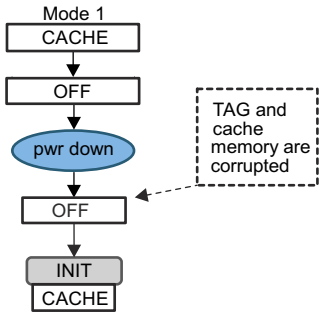


Figure 8-6. Software Precautions with No RAM Retention

8.3.2.2 Mode 2

Mode 2 is intended for systems where cache is in GPRAM mode. VIMS is retained with retention power to the GPRAM.

Note

If software tries to put VIMS into cache mode after retention, the system fails because the TAG memory is corrupted.

The correct procedure is to put VIMS in off mode *before* VIMS is put in cache mode. For more details, see Figure 8-7.

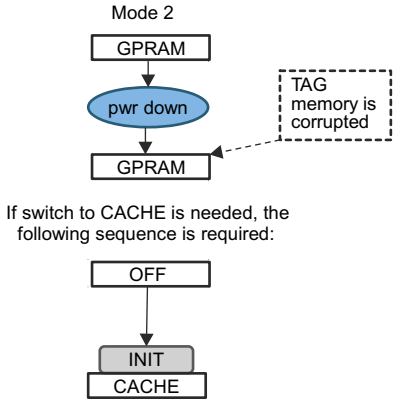


Figure 8-7. GPRAM Retention

8.3.2.3 Mode 3

Mode 3 is intended for use when the VIMS is in cache mode. In mode 3, VIMS can be in cache mode when the VIMS domain is powered off.

8.4 FLASH

The FLASH memory consists of a large MAIN region and several smaller regions, including FCFG, CCFG, TRIM and ENGR. The FCFG and CCFG regions can also be referred to as NONMAIN regions.

Each region of the FLASH memory is organized as a set of 2 KB blocks that can be individually erased. Programming the FLASH by changing bits from 1 to 0 can be done in the following increments:

- One 128-bit word
- Two 128-bit words
- Four 128-bit words

Erasing a block causes the entire contents of the block to be reset to all 1s. The 2 KB blocks are paired with sets of other 2 KB blocks that can be individually protected by being marked as read-only. Read-only blocks cannot be erased or programmed, which protects the contents of those blocks from being modified. There are separate lock bits for the MAIN, FCFG and CCFG, and these bits a located in FLASH and NVMNW.

The TRIM and ENGR regions of the flash are locked and are consequently inaccessible for program and erase.

There is a restriction on how many write operations are allowed to a FLASH row between erases. A row is comprised of 2048 bits (or 256 bytes). The FLASH memory is divided evenly into physical rows. One may perform a maximum of 83 write operations within a row between erases. If more than 83 write operations are performed before re-erasure, one may see unwritten bits in the row that are erased (in a logic 1 state) become programmed (change to a logic 0 state). User software must take care of this restriction, there is no hardware that checks and informs if this restriction is violated.

The FLASH block is mainly clocked by the 48 MHz system clock.

8.4.1 Flash Memory Protection

The FLASH memory can be write/erase protected in 2-KB sectors, or in groups of eight 2-KB sectors, depending on the location of the sectors in the FLASH memory. Protection is configured using either of two sets of lock bits. For the MAIN region, one configuration allows for individual protection for the first 32 2-KB sectors of the FLASH. Another configuration allow protection for groups of eight 2-KB sectors of the FLASH, beginning from sector 32 to the end of the FLASH memory. Protection in the CCFG and FCFG regions can be set per 2-KB sector.

Table 8-2. CC13x4x10 and CC26x4x10 Memory Write/Erase Protection

Memory Area CC13x4x10 and CC26x4x10 State(1) (2) (3)	FCFG0 (Efuse)	FCFG1 (Non-Main – Bank 0)	CCFG (Non-Main – Bank 1)	TI Locked Sector	Customer Locked	Customer Free
Unpacked die	Write ones (no way back)	Free	Free	None	None	All
Packed die	Locked	Free	Free	None	None	All
Engineering sample	Locked	Free	Free	None	None	All
Customer development	Locked	Locked	Free	Fixed	None	Except TI locked sectors
Customer delivery case 1	Locked	Locked	Writable (Not erasable)(4)	Fixed	Can add locked sectors(4)	Can be reduced(4)
Customer delivery case 2	Locked	Locked	Locked(4)	Fixed	Fixed(4)	Fixed(4)

(1) Locked: Not writable and not erasable
(2) Free: Writable and erasable
(3) Fixed: The number of this type is fixed
(4) The Chip Erase function erases all sectors locked by Texas Instruments

8.4.2 Flash Memory Programming

Memory programming is done using TI provided API. When calling the API functions, all interrupts that would trigger an access to the FLASH memory bank being written/erased should be disabled.

During a FLASH memory write or erase operation, the FLASH memory bank being written/erased should not be read. If instruction execution is required during a FLASH memory operation, the executing code must be placed in the other FLASH Memory bank not undergoing write/erase or in SRAM (and executed from SRAM) while the FLASH write/erase operation is in progress.

8.5 ROM Functions

Below is an overview of the main ROM functionalities. For more details on ROM functions available, see the DriverLib documentation in the SimpleLink™ CC13xx and CC26xx software development kit (SDK).

- Life cycle transition support used in TI production
- Device trimming and configuration during boot based on FCFG0 (eFuse), FCFG1, FCFG2 and CCFG contents
- Serial bootloader accessible on UART/SPI
- API functions for flash erase and program operations

The ROM is preprogrammed with a serial bootloader (SPI or UART). For applications that require in-field programmability, the royalty-free bootloader acts as an application loader and supports in-field firmware updates. The bootloader either executes automatically if no valid image has been written to the flash, or the bootloader can be started through a configurable GPIO. The bootloader cannot be called from application code.

8.6 VIMS Registers

8.6.1 FLASH Registers

Table 8-3 lists the memory-mapped registers for the FLASH registers. All register offset addresses not listed in Table 8-3 should be considered as reserved locations and the register contents should not be modified.

Table 8-3. FLASH Registers

Offset	Acronym	Register Name	Section
0h	WEPROT_B0_31_0_BY1	Internal	Section 8.6.1.1
4h	WEPROT_AUX_BY1	Internal	Section 8.6.1.2
1Ch	STAT	NW and Efuse Status	Section 8.6.1.3
24h	CFG	Internal	Section 8.6.1.4
2Ch	FLASH_SIZE	Internal	Section 8.6.1.5
3Ch	FWLOCK	Internal	Section 8.6.1.6
40h	FWFLAG	Internal	Section 8.6.1.7
50h	BANK0_TRIM_CFG_3	Internal	Section 8.6.1.8
54h	BANK0_TRIM_CFG_2	Internal	Section 8.6.1.9
58h	BANK0_TRIM_CFG_1	Internal	Section 8.6.1.10
5Ch	BANK0_TRIM_CFG_0	Internal	Section 8.6.1.11
60h	BANK1_TRIM_CFG_3	Internal	Section 8.6.1.12
64h	BANK1_TRIM_CFG_2	Internal	Section 8.6.1.13
68h	BANK1_TRIM_CFG_1	Internal	Section 8.6.1.14
6Ch	BANK1_TRIM_CFG_0	Internal	Section 8.6.1.15
70h	PUMP_TRIM_CFG_2	Internal	Section 8.6.1.16
74h	PUMP_TRIM_CFG_1	Internal	Section 8.6.1.17
78h	PUMP_TRIM_CFG_0	Internal	Section 8.6.1.18
1000h	EFUSE	Internal	Section 8.6.1.19
1004h	EFUSEADDR	Internal	Section 8.6.1.20
1008h	DATAUPPER	Internal	Section 8.6.1.21
100Ch	DATALOWER	Internal	Section 8.6.1.22
1010h	EFUSECFG	Internal	Section 8.6.1.23
1014h	EFUSESTAT	Internal	Section 8.6.1.24
1018h	ACC	Internal	Section 8.6.1.25
101Ch	BOUNDARY	Internal	Section 8.6.1.26
1020h	EFUSEFLAG	Internal	Section 8.6.1.27
1024h	EFUSEKEY	Internal	Section 8.6.1.28

Table 8-3. FLASH Registers (continued)

Offset	Acronym	Register Name	Section
1028h	EFUSERELEASE	Internal	Section 8.6.1.29
102Ch	EFUSEPINS	Internal	Section 8.6.1.30
1030h	EFUSECRA	Internal	Section 8.6.1.31
1034h	EFUSEREAD	Internal	Section 8.6.1.32
1038h	EFUSEPROGRAM	Internal	Section 8.6.1.33
103Ch	EFUSEERROR	Internal	Section 8.6.1.34
1040h	SINGLEBIT	Internal	Section 8.6.1.35
1044h	TWOBIT	Internal	Section 8.6.1.36
1048h	SELFTESTCYC	Internal	Section 8.6.1.37
104Ch	SELFTESTSIGN	Internal	Section 8.6.1.38

Complex bit access types are encoded to fit into small table cells. Table 8-4 shows the codes that are used for access types in this section.

Table 8-4. FLASH Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

8.6.1.1 WEPROT_B0_31_0_BY1 Register (Offset = 0h) [Reset = FFFFFFFFh]

WEPROT_B0_31_0_BY1 is shown in Table 8-5.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-5. WEPROT_B0_31_0_BY1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WEPROT_B0_31_0_BY1	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.1.2 WEPROT_AUX_BY1 Register (Offset = 4h) [Reset = 0000003Fh]

WEPROT_AUX_BY1 is shown in Table 8-6.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-6. WEPROT_AUX_BY1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	WEPROT_B1_ENGR_BY1	R/W	1h	Internal. Only to be used through TI provided API.
4	WEPROT_B0_ENGR_BY1	R/W	1h	Internal. Only to be used through TI provided API.
3	WEPROT_B1_TRIM_BY1	R/W	1h	Internal. Only to be used through TI provided API.
2	WEPROT_B0_TRIM_BY1	R/W	1h	Internal. Only to be used through TI provided API.
1	WEPROT_B1_FCFG_BY1	R/W	1h	Internal. Only to be used through TI provided API.
0	WEPROT_B0_CCFG_BY1	R/W	1h	Internal. Only to be used through TI provided API.

8.6.1.3 STAT Register (Offset = 1Ch) [Reset = 00000000h]

STAT is shown in Table 8-7.

Return to the Summary Table.

NW and Efuse Status

Table 8-7. STAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	STALLSTAT	R/W	0h	An ocp1 or ocp3 read stall has occurred. 0 : No stall or stall acknowledged by writing a 1 1 : Stall condition occurred/occurring This is a read/write-clear status bit. It will reset to 0. It will be set when either an ocp1 or ocp3 read occurs to a bank that is presently undergoing a program or write operation. An ocp2 write of 1 to this bit will clear the bit. The ocp2 write will take highest priority in the event an ocp1/ocp3 read is occurring concurrently to the ocp2 write. Clearing the bit should be done only after the ongoing program/erase operation is complete indicating that both banks are free. If clearing occurs while the stall condition persists, the field may get set back to one.
15	EFUSE_BLANK	R	0h	Efuse scanning detected if fuse ROM is blank: 0 : Not blank 1 : Blank
14	EFUSE_TIMEOUT	R	0h	Efuse scanning resulted in timeout error. 0 : No Timeout error 1 : Timeout Error
13	SPRS_BYTE_NOT_OK	R	0h	Efuse scanning resulted in scan chain Sparse byte error. 0 : No Sparse error 1 : Sparse Error
12-8	EFUSE_ERRCODE	R	0h	Same as EFUSEERROR.CODE
7-6	RESERVED	R	0h	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation
5-4	BUSY	R	0h	NW FW_SMSTAT.CMD_IN_PROGRESS bit. This flag is valid immediately after the operation setting it 0 : Not busy 1 : Busy Bit 4 is for the busy state for Bank0 which is at logical address 0x0 Bit 5 for Bank1.
3	READY1T	R	0h	1T access readiness status indicator from NW. Comes later than 2T readiness. 1: FLASH banks are ready for 1T accesses 0: FLASH banks are not ready for 1T accesses
2	READY2T	R	0h	2T access readiness status indicator from NW 1: FLASH banks are ready for 2T accesses 0: FLASH banks are not ready for 2T accesses
1-0	POWER_MODE	R	0h	Power state of each of the 2 flash arbiter FSM instances in the flash sub-system. For Thor, these bits should mostly be in the same state since both banks are in the same power mode. 0 : Active 1 : Ready for Low power (The 2T readiness has gone low or the flash_off_req has been set=1, and flash_off_ack is ready to be asserted). Bit 0 is for the power state for Bank0 which is at logical address 0x0 Bit 1 for Bank1

8.6.1.4 CFG Register (Offset = 24h) [Reset = 00000000h]

CFG is shown in Table 8-8.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-8. CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	DIS_FWTEST	R/W	0h	Internal. Only to be used through TI provided API.
29-12	RESERVED	R	0h	Reserved
11	MAIN_STICKY_EN	R/W	0h	Internal. Only to be used through TI provided API.
10	CCFG_STICKY_EN	R/W	0h	Internal. Only to be used through TI provided API.
9	FCFG_STICKY_EN	R/W	0h	Internal. Only to be used through TI provided API.
8	ENGR_TRIM_STICKY_EN	R/W	0h	Internal. Only to be used through TI provided API.
7-6	RESERVED	R	0h	Reserved
5	DIS_EFUSECLK	R/W	0h	Internal. Only to be used through TI provided API.
4	DIS_READACCESS	R/W	0h	Internal. Only to be used through TI provided API.
3-1	RESERVED	R	0h	Reserved
0	BP_TRIMCFG_EN	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.5 FLASH_SIZE Register (Offset = 2Ch) [Reset = 00000200h]

FLASH_SIZE is shown in Table 8-9.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-9. FLASH_SIZE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-7	SECTORS	R/W	4h	Internal. Only to be used through TI provided API.
6-0	RESERVED	R	0h	Internal. Only to be used through TI provided API.

8.6.1.6 FWLOCK Register (Offset = 3Ch) [Reset = 00000000h]

FWLOCK is shown in Table 8-10.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-10. FWLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	FWLOCK	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.7 FWFLAG Register (Offset = 40h) [Reset = 00000000h]

FWFLAG is shown in Table 8-11.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-11. FWFLAG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	FWFLAG	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.8 BANK0_TRIM_CFG_3 Register (Offset = 50h) [Reset = 00000000h]

BANK0_TRIM_CFG_3 is shown in Table 8-12.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-12. BANK0_TRIM_CFG_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

8.6.1.9 BANK0_TRIM_CFG_2 Register (Offset = 54h) [Reset = 00000000h]

BANK0_TRIM_CFG_2 is shown in Table 8-13.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-13. BANK0_TRIM_CFG_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

8.6.1.10 BANK0_TRIM_CFG_1 Register (Offset = 58h) [Reset = 00000000h]

BANK0_TRIM_CFG_1 is shown in Table 8-14.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-14. BANK0_TRIM_CFG_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-22	REDSWSELW3	R/W	0h	Internal. Only to be used through TI provided API.
21-16	REDSWSELW2	R/W	0h	Internal. Only to be used through TI provided API.
15-10	REDSWSELW1	R/W	0h	Internal. Only to be used through TI provided API.
9-4	REDSWSELW0	R/W	0h	Internal. Only to be used through TI provided API.
3	REDSWENW3	R/W	0h	Internal. Only to be used through TI provided API.
2	REDSWENW2	R/W	0h	Internal. Only to be used through TI provided API.
1	REDSWENW1	R/W	0h	Internal. Only to be used through TI provided API.
0	REDSWENW0	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.11 BANK0_TRIM_CFG_0 Register (Offset = 5Ch) [Reset = 00000000h]

BANK0_TRIM_CFG_0 is shown in Table 8-15.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-15. BANK0_TRIM_CFG_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BANK0_TRIM_CFG_0	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.12 BANK1_TRIM_CFG_3 Register (Offset = 60h) [Reset = 00000000h]

BANK1_TRIM_CFG_3 is shown in Table 8-16.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-16. BANK1_TRIM_CFG_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

8.6.1.13 BANK1_TRIM_CFG_2 Register (Offset = 64h) [Reset = 00000000h]

BANK1_TRIM_CFG_2 is shown in Table 8-17.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-17. BANK1_TRIM_CFG_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

8.6.1.14 BANK1_TRIM_CFG_1 Register (Offset = 68h) [Reset = 00000000h]

BANK1_TRIM_CFG_1 is shown in Table 8-18.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-18. BANK1_TRIM_CFG_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-22	REDSWSELW3	R/W	0h	Internal. Only to be used through TI provided API.
21-16	REDSWSELW2	R/W	0h	Internal. Only to be used through TI provided API.
15-10	REDSWSELW1	R/W	0h	Internal. Only to be used through TI provided API.
9-4	REDSWSELW0	R/W	0h	Internal. Only to be used through TI provided API.
3	REDSWENW3	R/W	0h	Internal. Only to be used through TI provided API.
2	REDSWENW2	R/W	0h	Internal. Only to be used through TI provided API.
1	REDSWENW1	R/W	0h	Internal. Only to be used through TI provided API.
0	REDSWENW0	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.15 BANK1_TRIM_CFG_0 Register (Offset = 6Ch) [Reset = 00000000h]

BANK1_TRIM_CFG_0 is shown in Table 8-19.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-19. BANK1_TRIM_CFG_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BANK1_TRIM_CFG_0	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.16 PUMP_TRIM_CFG_2 Register (Offset = 70h) [Reset = 00559400h]

PUMP_TRIM_CFG_2 is shown in Table 8-20.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-20. PUMP_TRIM_CFG_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25-20	VWLCT	R/W	5h	Internal. Only to be used through TI provided API.
19-14	VSLCT	R/W	16h	Internal. Only to be used through TI provided API.
13-9	VREADCT	R/W	Ah	Internal. Only to be used through TI provided API.
8-4	VINLOWCCORCT	R/W	0h	Internal. Only to be used through TI provided API.
3-0	VINHICCORCT	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.17 PUMP_TRIM_CFG_1 Register (Offset = 74h) [Reset = 19200000h]

PUMP_TRIM_CFG_1 is shown in Table 8-21.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-21. PUMP_TRIM_CFG_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	VINHICCORCTLSB	R/W	0h	Internal. Only to be used through TI provided API.
30-25	VINHCT	R/W	Ch	Internal. Only to be used through TI provided API.
24-20	VCGCT	R/W	12h	Internal. Only to be used through TI provided API.
19-15	IREFVRDCT	R/W	0h	Internal. Only to be used through TI provided API.
14-10	IREFTCCT	R/W	0h	Internal. Only to be used through TI provided API.
9-6	IREFCT	R/W	0h	Internal. Only to be used through TI provided API.
5-0	FOSCCT	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.18 PUMP_TRIM_CFG_0 Register (Offset = 78h) [Reset = 00B4C53Ah]

PUMP_TRIM_CFG_0 is shown in Table 8-22.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-22. PUMP_TRIM_CFG_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	VHVCT_PV	R/W	Bh	Internal. Only to be used through TI provided API.
19-10	VHVCT_PGM	R/W	131h	Internal. Only to be used through TI provided API.
9-0	VHVCT_ERS	R/W	13Ah	Internal. Only to be used through TI provided API.

8.6.1.19 EFUSE Register (Offset = 1000h) [Reset = 00000000h]

EFUSE is shown in Table 8-23.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-23. EFUSE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	Reserved
28-24	INSTRUCTION	R/W	0h	Internal. Only to be used through TI provided API.
23-16	RESERVED	R	0h	Reserved
15-0	DUMPWORD	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.20 EFUSEADDR Register (Offset = 1004h) [Reset = 00000000h]

EFUSEADDR is shown in Table 8-24.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-24. EFUSEADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-11	BLOCK	R/W	0h	Internal. Only to be used through TI provided API.
10-0	ROW	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.21 DATAUPPER Register (Offset = 1008h) [Reset = 00000000h]

DATAUPPER is shown in Table 8-25.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-25. DATAUPPER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-3	SPARE	R/W	0h	Internal. Only to be used through TI provided API.
2	P	R/W	0h	Internal. Only to be used through TI provided API.
1	R	R/W	0h	Internal. Only to be used through TI provided API.
0	EEN	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.22 DATALOWER Register (Offset = 100Ch) [Reset = 00000000h]

DATALOWER is shown in Table 8-26.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-26. DATALOWER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.23 EFUSECFG Register (Offset = 1010h) [Reset = 00000001h]

EFUSECFG is shown in Table 8-27.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-27. EFUSECFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	IDLEGATING	R/W	0h	Internal. Only to be used through TI provided API.
7-5	RESERVED	R	0h	Reserved
4-3	SLAVEPOWER	R/W	0h	Internal. Only to be used through TI provided API.
2-1	RESERVED	R	0h	Reserved
0	GATING	R/W	1h	Internal. Only to be used through TI provided API.

8.6.1.24 EFUSESTAT Register (Offset = 1014h) [Reset = 00000001h]

EFUSESTAT is shown in Table 8-28.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-28. EFUSESTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RESETDONE	R	1h	Internal. Only to be used through TI provided API.

8.6.1.25 ACC Register (Offset = 1018h) [Reset = 00000000h]

ACC is shown in Table 8-29.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-29. ACC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	ACCUMULATOR	R	0h	Internal. Only to be used through TI provided API.

8.6.1.26 BOUNDARY Register (Offset = 101Ch) [Reset = 00000000h]

BOUNDARY is shown in Table 8-30.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-30. BOUNDARY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23	DISROW0	R/W	0h	Internal. Only to be used through TI provided API.
22	SPARE	R/W	0h	Internal. Only to be used through TI provided API.
21	EFC_SELF_TEST_ERROR	R/W	0h	Internal. Only to be used through TI provided API.
20	EFC_INSTRUCTION_INFO	R/W	0h	Internal. Only to be used through TI provided API.
19	EFC_INSTRUCTION_ERROR	R/W	0h	Internal. Only to be used through TI provided API.
18	EFC_AUTOLOAD_ERROR	R/W	0h	Internal. Only to be used through TI provided API.
17-14	OUTPUTENABLE	R/W	0h	Internal. Only to be used through TI provided API.
13	SYS_ECC_SELF_TEST_EN	R/W	0h	Internal. Only to be used through TI provided API.
12	SYS_ECC_OVERRIDE_EN	R/W	0h	Internal. Only to be used through TI provided API.
11	EFC_FDI	R/W	0h	Internal. Only to be used through TI provided API.
10	SYS_DIEID_AUTOLOAD_EN	R/W	0h	Internal. Only to be used through TI provided API.
9-8	SYS_REPAIR_EN	R/W	0h	Internal. Only to be used through TI provided API.
7-4	SYS_WS_READ_STATES	R/W	0h	Internal. Only to be used through TI provided API.
3-0	INPUTENABLE	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.27 EFUSEFLAG Register (Offset = 1020h) [Reset = 00000000h]

EFUSEFLAG is shown in Table 8-31.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-31. EFUSEFLAG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	KEY	R	0h	Internal. Only to be used through TI provided API.

8.6.1.28 EFUSEKEY Register (Offset = 1024h) [Reset = 00000000h]

EFUSEKEY is shown in Table 8-32.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-32. EFUSEKEY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CODE	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.29 EFUSERELEASE Register (Offset = 1028h) [Reset = 00000000h]

EFUSERELEASE is shown in Table 8-33.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-33. EFUSERELEASE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	ODPYEAR	R	X	Internal. Only to be used through TI provided API.
24-21	ODPMONTH	R	X	Internal. Only to be used through TI provided API.
20-16	ODPDAY	R	X	Internal. Only to be used through TI provided API.
15-9	EFUSEYEAR	R	X	Internal. Only to be used through TI provided API.
8-5	EFUSEMONTH	R	X	Internal. Only to be used through TI provided API.
4-0	EFUSEDAY	R	X	Internal. Only to be used through TI provided API.

8.6.1.30 EFUSEPINS Register (Offset = 102Ch) [Reset = 00000000h]

EFUSEPINS is shown in Table 8-34.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-34. EFUSEPINS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Internal. Only to be used through TI provided API.
15	EFC_SELF_TEST_DONE	R	X	Internal. Only to be used through TI provided API.
14	EFC_SELF_TEST_ERROR	R	X	Internal. Only to be used through TI provided API.
13	SYS_ECC_SELF_TEST_EN	R	X	Internal. Only to be used through TI provided API.
12	EFC_INSTRUCTION_INFO	R	X	Internal. Only to be used through TI provided API.
11	EFC_INSTRUCTION_ERROR	R	X	Internal. Only to be used through TI provided API.
10	EFC_AUTOLOAD_ERROR	R	X	Internal. Only to be used through TI provided API.
9	SYS_ECC_OVERRIDE_EN	R	X	Internal. Only to be used through TI provided API.
8	EFC_READY	R	X	Internal. Only to be used through TI provided API.
7	EFC_FCLRZ	R	X	Internal. Only to be used through TI provided API.
6	SYS_DIEID_AUTOLOAD_EN	R	X	Internal. Only to be used through TI provided API.
5-4	SYS_REPAIR_EN	R	X	Internal. Only to be used through TI provided API.
3-0	SYS_WS_READ_STATES	R	X	Internal. Only to be used through TI provided API.

8.6.1.31 EFUSECRA Register (Offset = 1030h) [Reset = 00000000h]

EFUSECRA is shown in Table 8-35.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-35. EFUSECRA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	DATA	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.32 EFUSEREAD Register (Offset = 1034h) [Reset = 00000000h]

EFUSEREAD is shown in Table 8-36.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-36. EFUSEREAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-8	DATABIT	R/W	0h	Internal. Only to be used through TI provided API.
7-4	READCLOCK	R/W	0h	Internal. Only to be used through TI provided API.
3	DEBUG	R/W	0h	Internal. Only to be used through TI provided API.
2	SPARE	R/W	0h	Internal. Only to be used through TI provided API.
1-0	MARGIN	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.33 EFUSEPROGRAM Register (Offset = 1038h) [Reset = 00000000h]

EFUSEPROGRAM is shown in Table 8-37.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-37. EFUSEPROGRAM Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	COMPAREDISABLE	R/W	0h	Internal. Only to be used through TI provided API.
29-14	CLOCKSTALL	R/W	0h	Internal. Only to be used through TI provided API.
13	VPPTOVDD	R/W	0h	Internal. Only to be used through TI provided API.
12-9	ITERATIONS	R/W	0h	Internal. Only to be used through TI provided API.
8-0	WRITECLOCK	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.34 EFUSEERROR Register (Offset = 103Ch) [Reset = 00000000h]

EFUSEERROR is shown in Table 8-38.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-38. EFUSEERROR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	DONE	R/W	0h	Internal. Only to be used through TI provided API.
4-0	CODE	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.35 SINGLEBIT Register (Offset = 1040h) [Reset = 00000000h]

SINGLEBIT is shown in Table 8-39.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-39. SINGLEBIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	FROMN	R	0h	Internal. Only to be used through TI provided API.
0	FROM0	R	0h	Internal. Only to be used through TI provided API.

8.6.1.36 TWOBIT Register (Offset = 1044h) [Reset = 00000000h]

TWOBIT is shown in Table 8-40.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-40. TWOBIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	FROMN	R	0h	Internal. Only to be used through TI provided API.
0	FROM0	R	0h	Internal. Only to be used through TI provided API.

8.6.1.37 SELFTESTCYC Register (Offset = 1048h) [Reset = 00000000h]

SELFTESTCYC is shown in Table 8-41.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-41. SELFTESTCYC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CYCLES	R/W	0h	Internal. Only to be used through TI provided API.

8.6.1.38 SELFTESTSIGN Register (Offset = 104Ch) [Reset = 00000000h]

SELFTESTSIGN is shown in Table 8-42.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-42. SELFTESTSIGN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	Internal. Only to be used through TI provided API.

8.6.2 VIMS Registers

Table 8-43 lists the memory-mapped registers for the VIMS registers. All register offset addresses not listed in Table 8-43 should be considered as reserved locations and the register contents should not be modified.

Table 8-43. VIMS Registers

Offset	Acronym	Register Name	Section
0h	STAT	Status	Section 8.6.2.1
4h	CTL	Control	Section 8.6.2.2

Complex bit access types are encoded to fit into small table cells. Table 8-44 shows the codes that are used for access types in this section.

Table 8-44. VIMS Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

8.6.2.1 STAT Register (Offset = 0h) [Reset = 00000000h]

STAT is shown in Table 8-45.

Return to the Summary Table.

Status
Displays current VIMS mode and line buffer status

Table 8-45. STAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	IDCODE_LB_DIS	R	0h	Icode/Dcode flash line buffer status 0: Enabled or in transition to disabled 1: Disabled and flushed
4	SYSBUS_LB_DIS	R	0h	Sysbus flash line buffer control 0: Enabled or in transition to disabled 1: Disabled and flushed
3	MODE_CHANGING	R	0h	VIMS mode change status 0: VIMS is in the mode defined by MODE 1: VIMS is in the process of changing to the mode given in CTL.MODE
2	INV	R	0h	This bit is set when invalidation of the cache memory is active / ongoing
1-0	MODE	R	0h	Current VIMS mode 0h = GPRAM : VIMS GPRAM mode 1h = CACHE : VIMS Cache mode 3h = VIMS Off mode

8.6.2.2 CTL Register (Offset = 4h) [Reset = 00000000h]

CTL is shown in Table 8-46.

Return to the Summary Table.

Control
Configure VIMS mode and line buffer settings

Table 8-46. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	STATS_CLR	R/W	0h	Set this bit to clear statistic counters.
30	STATS_EN	R/W	0h	Set this bit to enable statistic counters.
29	DYN_CG_EN	R/W	0h	0: The in-built clock gate functionality is bypassed. 1: The in-built clock gate functionality is enabled, automatically gating the clock when not needed.
28-6	RESERVED	R	0h	Reserved
5	IDCODE_LB_DIS	R/W	0h	Icode/Dcode flash line buffer control 0: Enable 1: Disable
4	SYSBUS_LB_DIS	R/W	0h	Sysbus flash line buffer control 0: Enable 1: Disable
3	ARB_CFG	R/W	0h	Icode/Dcode and sysbus arbitation scheme 0: Static arbitration (icode/docde > sysbus) 1: Round-robin arbitration
2	PREF_EN	R/W	0h	Tag prefetch control 0: Disabled 1: Enabled
1-0	MODE	R/W	0h	VIMS mode request. Write accesses to this field will be blocked while STAT.MODE_CHANGING is set to 1. 0h = GPRAM : VIMS GPRAM mode 1h = CACHE : VIMS Cache mode 3h = VIMS Off mode

8.6.3 NVMNW Registers

Table 8-47 lists the memory-mapped registers for the NVMNW registers. All register offset addresses not listed in Table 8-47 should be considered as reserved locations and the register contents should not be modified.

Table 8-47. NVMNW Registers

Offset	Acronym	Register Name	Section
20h	IIDX	Internal	Section 8.6.3.1
28h	IMASK	Internal	Section 8.6.3.2
30h	RIS	Internal	Section 8.6.3.3
38h	MIS	Internal	Section 8.6.3.4
40h	ISET	Internal	Section 8.6.3.5
48h	ICLR	Internal	Section 8.6.3.6
E0h	EVT_MODE	Internal	Section 8.6.3.7
FCh	DESC	Internal	Section 8.6.3.8
100h	CMDEXEC	Internal	Section 8.6.3.9
104h	CMDTYPE	Internal	Section 8.6.3.10
108h	CMDCTL	Internal	Section 8.6.3.11
120h	CMDADDR	Internal	Section 8.6.3.12
124h	CMDBYTEN	Internal	Section 8.6.3.13
12Ch	CMDDATAINDEX	Internal	Section 8.6.3.14
130h	CMDDATA0	Internal	Section 8.6.3.15
134h	CMDDATA1	Internal	Section 8.6.3.16
138h	CMDDATA2	Internal	Section 8.6.3.17
13Ch	CMDDATA3	Internal	Section 8.6.3.18
140h	CMDDATA4	Internal	Section 8.6.3.19
144h	CMDDATA5	Internal	Section 8.6.3.20
148h	CMDDATA6	Internal	Section 8.6.3.21
14Ch	CMDDATA7	Internal	Section 8.6.3.22
150h	CMDDATA8	Internal	Section 8.6.3.23
154h	CMDDATA9	Internal	Section 8.6.3.24
158h	CMDDATA10	Internal	Section 8.6.3.25
15Ch	CMDDATA11	Internal	Section 8.6.3.26
160h	CMDDATA12	Internal	Section 8.6.3.27
164h	CMDDATA13	Internal	Section 8.6.3.28
168h	CMDDATA14	Internal	Section 8.6.3.29
16Ch	CMDDATA15	Internal	Section 8.6.3.30
1D0h	CMDWEPROTA	Internal	Section 8.6.3.31
1D4h	CMDWEPROTB	Internal	Section 8.6.3.32
210h	CMDWEPROTNM	Internal	Section 8.6.3.33
214h	CMDWEPROTTR	Internal	Section 8.6.3.34
218h	CMDWEPROTEN	Internal	Section 8.6.3.35
3B0h	CFGCMD	Internal	Section 8.6.3.36
3B4h	CFGPCNT	Internal	Section 8.6.3.37
3D0h	STATCMD	Internal	Section 8.6.3.38
3D4h	STATADDR	Internal	Section 8.6.3.39
3D8h	STATPCNT	Internal	Section 8.6.3.40
3DCh	STATMODE	Internal	Section 8.6.3.41
3F0h	GBLINFO0	Internal	Section 8.6.3.42

Table 8-47. NVMNW Registers (continued)

Offset	Acronym	Register Name	Section
3F4h	GBLINFO1	Internal	Section 8.6.3.43
3F8h	GBLINFO2	Internal	Section 8.6.3.44
400h	BANK0INFO0	Internal	Section 8.6.3.45
404h	BANK0INFO1	Internal	Section 8.6.3.46
410h	BANK1INFO0	Internal	Section 8.6.3.47
414h	BANK1INFO1	Internal	Section 8.6.3.48

Complex bit access types are encoded to fit into small table cells. Table 8-48 shows the codes that are used for access types in this section.

Table 8-48. NVMNW Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

8.6.3.1 IIDX Register (Offset = 20h) [Reset = 00000000h]

IIDX is shown in Table 8-49.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-49. IIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	STAT	R	0h	Internal. Only to be used through TI provided API.

8.6.3.2 IMASK Register (Offset = 28h) [Reset = 00000000h]

IMASK is shown in Table 8-50.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-50. IMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	DONE	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.3 RIS Register (Offset = 30h) [Reset = 00000000h]

RIS is shown in Table 8-51.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-51. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	DONE	R	0h	Internal. Only to be used through TI provided API.

8.6.3.4 MIS Register (Offset = 38h) [Reset = 00000000h]

MIS is shown in Table 8-52.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-52. MIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	DONE	R	0h	Internal. Only to be used through TI provided API.

8.6.3.5 ISET Register (Offset = 40h) [Reset = 00000000h]

ISET is shown in Table 8-53.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-53. ISET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	DONE	W	0h	Internal. Only to be used through TI provided API.

8.6.3.6 ICLR Register (Offset = 48h) [Reset = 00000000h]

ICLR is shown in Table 8-54.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-54. ICLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	DONE	W	0h	Internal. Only to be used through TI provided API.

8.6.3.7 EVT_MODE Register (Offset = E0h) [Reset = 00000001h]

EVT_MODE is shown in Table 8-55.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-55. EVT_MODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Internal. Only to be used through TI provided API.
1-0	INT0_CFG	R	1h	Internal. Only to be used through TI provided API.

8.6.3.8 DESC Register (Offset = FCh) [Reset = 0B400010h]

DESC is shown in Table 8-56.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-56. DESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MODULEID	R	B40h	Internal. Only to be used through TI provided API.
15-12	FEATUREVER	R	0h	Internal. Only to be used through TI provided API.
11-8	INSTNUM	R	0h	Internal. Only to be used through TI provided API.
7-4	MAJREV	R	1h	Internal. Only to be used through TI provided API.
3-0	MINREV	R	0h	Internal. Only to be used through TI provided API.

8.6.3.9 CMDEXEC Register (Offset = 100h) [Reset = 00000000h]

CMDEXEC is shown in Table 8-57.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-57. CMDEXEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	VAL	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.10 CMDTYPE Register (Offset = 104h) [Reset = 00h]

CMDTYPE is shown in Table 8-58.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-58. CMDTYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
6-4	SIZE	R/W	0h	Internal. Only to be used through TI provided API.
3	RESERVED	R	0h	Internal. Only to be used through TI provided API.
2-0	COMMAND	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.11 CMDCTL Register (Offset = 108h) [Reset = 0000C000h]

CMDCTL is shown in Table 8-59.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-59. CMDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Internal. Only to be used through TI provided API.
21	DATAVEREN	R/W	0h	Internal. Only to be used through TI provided API.
20	SSERASEDIS	R/W	0h	Internal. Only to be used through TI provided API.
19	ERASEMASKDIS	R/W	0h	Internal. Only to be used through TI provided API.
18	PROGMASKDIS	R/W	0h	Internal. Only to be used through TI provided API.
17	RESERVED	R	0h	Internal. Only to be used through TI provided API.
16	ADDRXLATEOVR	R/W	0h	Internal. Only to be used through TI provided API.
15	POSTVEREN	R/W	1h	Internal. Only to be used through TI provided API.
14	PREVEREN	R/W	1h	Internal. Only to be used through TI provided API.
13	RESERVED	R	0h	Internal. Only to be used through TI provided API.
12-9	REGIONSEL	R/W	0h	Internal. Only to be used through TI provided API.
8-5	RESERVED	R/W	0h	
4	BANKSEL	R/W	0h	Internal. Only to be used through TI provided API.
3-0	MODESEL	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.12 CMDADDR Register (Offset = 120h) [Reset = 00000000h]

CMDADDR is shown in Table 8-60.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-60. CMDADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.13 CMDBYTEN Register (Offset = 124h) [Reset = 0000FFFFh]

CMDBYTEN is shown in Table 8-61.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-61. CMDBYTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Internal. Only to be used through TI provided API.
15-0	VAL	R/W	FFFFh	Internal. Only to be used through TI provided API.

8.6.3.14 CMDDATAINDEX Register (Offset = 12Ch) [Reset = 00000000h]

CMDDATAINDEX is shown in Table 8-62.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-62. CMDDATAINDEX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Internal. Only to be used through TI provided API.
1-0	VAL	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.15 CMDDATA0 Register (Offset = 130h) [Reset = FFFFFFFFh]

CMDDATA0 is shown in Table 8-63.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-63. CMDDATA0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.16 CMDDATA1 Register (Offset = 134h) [Reset = FFFFFFFFh]

CMDDATA1 is shown in Table 8-64.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-64. CMDDATA1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.17 CMDDATA2 Register (Offset = 138h) [Reset = FFFFFFFFh]

CMDDATA2 is shown in Table 8-65.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-65. CMDDATA2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.18 CMDDATA3 Register (Offset = 13Ch) [Reset = FFFFFFFFh]

CMDDATA3 is shown in Table 8-66.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-66. CMDDATA3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.19 CMDDATA4 Register (Offset = 140h) [Reset = FFFFFFFFh]

CMDDATA4 is shown in Table 8-67.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-67. CMDDATA4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.20 CMDDATA5 Register (Offset = 144h) [Reset = FFFFFFFFh]

CMDDATA5 is shown in Table 8-68.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-68. CMDDATA5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.21 CMDDATA6 Register (Offset = 148h) [Reset = FFFFFFFFh]

CMDDATA6 is shown in Table 8-69.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-69. CMDDATA6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.22 CMDDATA7 Register (Offset = 14Ch) [Reset = FFFFFFFFh]

CMDDATA7 is shown in Table 8-70.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-70. CMDDATA7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.23 CMDDATA8 Register (Offset = 150h) [Reset = FFFFFFFFh]

CMDDATA8 is shown in Table 8-71.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-71. CMDDATA8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.24 CMDDATA9 Register (Offset = 154h) [Reset = FFFFFFFFh]

CMDDATA9 is shown in Table 8-72.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-72. CMDDATA9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.25 CMDDATA10 Register (Offset = 158h) [Reset = FFFFFFFFh]

CMDDATA10 is shown in Table 8-73.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-73. CMDDATA10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.26 CMDDATA11 Register (Offset = 15Ch) [Reset = FFFFFFFFh]

CMDDATA11 is shown in Table 8-74.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-74. CMDDATA11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.27 CMDDATA12 Register (Offset = 160h) [Reset = FFFFFFFFh]

CMDDATA12 is shown in Table 8-75.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-75. CMDDATA12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.28 CMDDATA13 Register (Offset = 164h) [Reset = FFFFFFFFh]

CMDDATA13 is shown in Table 8-76.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-76. CMDDATA13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.29 CMDDATA14 Register (Offset = 168h) [Reset = FFFFFFFFh]

CMDDATA14 is shown in Table 8-77.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-77. CMDDATA14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.30 CMDDATA15 Register (Offset = 16Ch) [Reset = FFFFFFFFh]

CMDDATA15 is shown in Table 8-78.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-78. CMDDATA15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.31 CMDWEPROTA Register (Offset = 1D0h) [Reset = FFFFFFFFh]

CMDWEPROTA is shown in Table 8-79.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-79. CMDWEPROTA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.32 CMDWEPROTB Register (Offset = 1D4h) [Reset = FFFFFFFFh]

CMDWEPROTB is shown in Table 8-80.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-80. CMDWEPROTB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	FFFFFFFFh	Internal. Only to be used through TI provided API.

8.6.3.33 CMDWEPROTNM Register (Offset = 210h) [Reset = 00000001h]

CMDWEPROTNM is shown in Table 8-81.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-81. CMDWEPROTNM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	VAL	R/W	1h	Internal. Only to be used through TI provided API.

8.6.3.34 CMDWEPROTTR Register (Offset = 214h) [Reset = 00000001h]

CMDWEPROTTR is shown in Table 8-82.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-82. CMDWEPROTTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	VAL	R/W	1h	Internal. Only to be used through TI provided API.

8.6.3.35 CMDWEPROTEN Register (Offset = 218h) [Reset = 00000001h]

CMDWEPROTEN is shown in Table 8-83.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-83. CMDWEPROTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	VAL	R/W	1h	Internal. Only to be used through TI provided API.

8.6.3.36 CFGCMD Register (Offset = 3B0h) [Reset = 000000002h]

CFGCMD is shown in Table 8-84.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-84. CFGCMD Register Field Descriptions

Bit	Field	Type	Reset	Description
32-4	RESERVED	R	0h	Internal. Only to be used through TI provided API.
3-0	WAITSTATE	R/W	2h	Internal. Only to be used through TI provided API.

8.6.3.37 CFGPCNT Register (Offset = 3B4h) [Reset = 000000000h]

CFGPCNT is shown in Table 8-85.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-85. CFGPCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
32-12	RESERVED	R	0h	Internal. Only to be used through TI provided API.
11-4	MAXPCNTVAL	R/W	0h	Internal. Only to be used through TI provided API.
3-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	MAXPCNTOVR	R/W	0h	Internal. Only to be used through TI provided API.

8.6.3.38 STATCMD Register (Offset = 3D0h) [Reset = 00000000h]

STATCMD is shown in Table 8-86.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-86. STATCMD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	Internal. Only to be used through TI provided API.
12	FAILMISC	R	0h	Internal. Only to be used through TI provided API.
11-9	RESERVED	R	0h	
8	FAILINVDATA	R	0h	Internal. Only to be used through TI provided API.
7	FAILMODE	R	0h	Internal. Only to be used through TI provided API.
6	FAILILLADDR	R	0h	Internal. Only to be used through TI provided API.
5	FAILVERIFY	R	0h	Internal. Only to be used through TI provided API.
4	FAILWEPROT	R	0h	Internal. Only to be used through TI provided API.
3	RESERVED	R	0h	Internal. Only to be used through TI provided API.
2	CMDINPROGRESS	R	0h	Internal. Only to be used through TI provided API.
1	CMDPASS	R	0h	Internal. Only to be used through TI provided API.
0	CMDDONE	R	0h	Internal. Only to be used through TI provided API.

8.6.3.39 STATADDR Register (Offset = 3D4h) [Reset = 00200000h]

STATADDR is shown in Table 8-87.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-87. STATADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Internal. Only to be used through TI provided API.
25-21	BANKID	R	1h	Internal. Only to be used through TI provided API.
20-16	REGIONID	R	0h	Internal. Only to be used through TI provided API.
15-0	BANKADDR	R	0h	Internal. Only to be used through TI provided API.

8.6.3.40 STATPCNT Register (Offset = 3D8h) [Reset = 00000000h]

STATPCNT is shown in Table 8-88.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-88. STATPCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Internal. Only to be used through TI provided API.
11-0	PULSECNT	R	0h	Internal. Only to be used through TI provided API.

8.6.3.41 STATMODE Register (Offset = 3DCh) [Reset = 00000h]

STATMODE is shown in Table 8-89.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-89. STATMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
17	BANK1TRDY	R	0h	Internal. Only to be used through TI provided API.
16	BANK2TRDY	R	0h	Internal. Only to be used through TI provided API.
15-12	RESERVED	R	0h	
11-8	BANKMODE	R	0h	Internal. Only to be used through TI provided API.
7-1	RESERVED	R	0h	Internal. Only to be used through TI provided API.
0	BANKNOTINRD	R	0h	Internal. Only to be used through TI provided API.

8.6.3.42 GBLINFO0 Register (Offset = 3F0h) [Reset = 20800h]

GBLINFO0 is shown in Table 8-90.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-90. GBLINFO0 Register Field Descriptions

Bit	Field	Type	Reset	Description
18-16	NUMBANKS	R	2h	Internal. Only to be used through TI provided API.
15-0	SECTORSIZE	R	800h	Internal. Only to be used through TI provided API.

8.6.3.43 GBLINFO1 Register (Offset = 3F4h) [Reset = 00040080h]

GBLINFO1 is shown in Table 8-91.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-91. GBLINFO1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Internal. Only to be used through TI provided API.
18-16	REDWIDTH	R	4h	Internal. Only to be used through TI provided API.
15-13	RESERVED	R	0h	
12-8	ECCWIDTH	R	0h	Internal. Only to be used through TI provided API.
7-0	DATAWIDTH	R	80h	Internal. Only to be used through TI provided API.

8.6.3.44 GBLINFO2 Register (Offset = 3F8h) [Reset = 00000004h]

GBLINFO2 is shown in Table 8-92.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-92. GBLINFO2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Internal. Only to be used through TI provided API.
3-0	DATAREGISTERS	R	4h	Internal. Only to be used through TI provided API.

8.6.3.45 BANK0INFO0 Register (Offset = 400h) [Reset = 00000100h]

BANK0INFO0 is shown in Table 8-93.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-93. BANK0INFO0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Internal. Only to be used through TI provided API.
11-0	MAINSIZE	R	100h	Internal. Only to be used through TI provided API.

8.6.3.46 BANK0INFO1 Register (Offset = 404h) [Reset = 00010101h]

BANK0INFO1 is shown in Table 8-94.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-94. BANK0INFO1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Internal. Only to be used through TI provided API.
23-16	ENGRSIZE	R	1h	Internal. Only to be used through TI provided API.
15-8	TRIMSIZE	R	1h	Internal. Only to be used through TI provided API.
7-0	NONMAINSIZE	R	1h	Internal. Only to be used through TI provided API.

8.6.3.47 BANK1INFO0 Register (Offset = 410h) [Reset = 00000100h]

BANK1INFO0 is shown in Table 8-95.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-95. BANK1INFO0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Internal. Only to be used through TI provided API.
11-0	MAINSIZE	R	100h	Internal. Only to be used through TI provided API.

8.6.3.48 BANK1INFO1 Register (Offset = 414h) [Reset = 00010101h]

BANK1INFO1 is shown in Table 8-96.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 8-96. BANK1INFO1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Internal. Only to be used through TI provided API.
23-16	ENGRSIZE	R	1h	Internal. Only to be used through TI provided API.
15-8	TRIMSIZE	R	1h	Internal. Only to be used through TI provided API.
7-0	NONMAINSIZE	R	1h	Internal. Only to be used through TI provided API.

This page intentionally left blank.

Chapter 9

SRAM

This chapter discusses the RAM system of the CC13x4x10 and CC26x4x10 device platform.

9.1 Introduction 718
9.2 Main Features 718
9.3 Data Retention 718
9.4 Parity and SRAM Error Support 718
9.5 SRAM Auto-Initialization 718
9.6 Parity Debug Behavior 718
9.7 SRAM Registers 719

9.1 Introduction

The CC13x4x10 and CC26x4x10 device platform provides 256 KB of system RAM consisting of single-cycle on-chip SRAM. It is extendable up to 288 KB when parity is disabled. The SRAM also supports full data retention in standby power mode.

Bit-banding is supported in order to reduce the execution time for read-modify-write (RMW) operations to memory. With bit-banding, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in one atomic operation.

9.2 Main Features

The main features of the SRAM are:

- Configurable data retention in standby power mode
- Parity error detection
- Auto-initialization function

9.3 Data Retention

Data stored in SRAM is retained during standby power mode. SRAM retention can be configured in banks by the AON_PMCTL:RAMCFG register (see Section 7.8.2). All of the memory is retained by default.

9.4 Parity and SRAM Error Support

During an SRAM write, a parity bit is calculated and stored for each byte that is written. Parity error detection is done on a byte-wide basis during an SRAM read operation. This means that a parity error on any byte in a memory read operation causes a memory data error to be detected. A parity error causes a processor bus fault exception (see Section 5.1.2).

The SRAM address that was read during parity error detection is captured in the SRAM_MMR:PER_CHK register. This parity error address is stored as an offset from the base address of SRAM memory.

9.4.1 SRAM Extension Mode

When parity is disabled, the usable SRAM space for the application is extended to 288 KB.

9.5 SRAM Auto-Initialization

The SRAM can be initialized by dedicated auto-initialization hardware. All memory locations are initialized to zero and the parity information is initialized as required. The memory initialization ensures that parity errors are not generated due to reads from locations that have not been initialized by software. Auto-initialization is performed using the SRAM_MMR:MEM_CTL register.

SRAM_MMR:MEM_CTL also allows selective or partial initialization of SRAM space with granularity of 32 KB. The sub-regions can be selected using the SRAM_MMR:MEM_CTL.MEM_SEL field, where each bit corresponds to one 32 KB block. SRAM_MMR:MEM_CTL.MEM_SEL[16] selects initialization of parity bits.

Note

The SRAM is auto-initialized during the system CPU boot sequence after a system reset.

9.6 Parity Debug Behavior

The following features are provided to allow debugging and testing of the handling of parity error detection and bus fault exceptions that are due to parity errors.

- An SRAM parity error can be forced by using the SRAM_MMR:PER_CTL register. The generated parity error will be observed after reading from the address offset that has been written to the SRAM_MMR:PER_DBG register.
- Updating of the status in the SRAM_MMR:PER_CHK register can be disabled by setting the SRAM_MMR:PER_CTL.PER_DISABLE bit. This can be used by debugger software in situations where it

is known that parity errors will be generated by the debugger software reading from uninitialized memory locations.

9.7 SRAM Registers

9.7.1 SRAM_MMR Registers

Table 9-1 lists the memory-mapped registers for the SRAM_MMR registers. All register offset addresses not listed in Table 9-1 should be considered as reserved locations and the register contents should not be modified.

Table 9-1. SRAM_MMR Registers

Offset	Acronym	Register Name	Section
0h	PER_CTL	Parity Error Control	Section 9.7.1.1
4h	PER_CHK	Parity Error Check	Section 9.7.1.2
8h	PER_DBG	Parity Error Debug	Section 9.7.1.3
Ch	MEM_CTL	Memory Control	Section 9.7.1.4
10h	MEM_STA	Memory Status	Section 9.7.1.5

Complex bit access types are encoded to fit into small table cells. Table 9-2 shows the codes that are used for access types in this section.

Table 9-2. SRAM_MMR Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

9.7.1.1 PER_CTL Register (Offset = 0h) [Reset = 00000000h]

PER_CTL is shown in Table 9-3.

Return to the Summary Table.

Parity Error Control
Parity error check controls

Table 9-3. PER_CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	PER_DISABLE	R/W	0h	Parity Status Disable 0: A parity error will update PER_CHK.PER_ADDR field 1: Parity error does not update PER_CHK.PER_ADDR field
7-1	RESERVED	R/W	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
0	PER_DEBUG_ENABLE	R/W	0h	Parity Error Debug Enable 0: Normal operation 1: An address offset can be written to PER_DBG.PER_DEBUG_ADDR and parity errors will be generated on reads from within this offset

9.7.1.2 PER_CHK Register (Offset = 4h) [Reset = 00000000h]

PER_CHK is shown in Table 9-4.

Return to the Summary Table.

Parity Error Check
Parity error check results

Table 9-4. PER_CHK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	PER_ADDR	R	0h	Parity Error Address Offset Returns the last address offset which resulted in a parity error during an SRAM read. The address offset returned is always the word-aligned address that contains the location with the parity error. For parity faults on non word-aligned accesses, CPU_SCS:BFAR.ADDRESS will hold the address of the location that resulted in parity error.

9.7.1.3 PER_DBG Register (Offset = 8h) [Reset = 00000000h]

PER_DBG is shown in Table 9-5.

Return to the Summary Table.

Parity Error Debug
Parity error check debug address setting

Table 9-5. PER_DBG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	PER_DEBUG_ADDR	R/W	0h	Debug Parity Error Address Offset When PER_CTL.PER_DEBUG is 1, this field is used to set a parity debug address offset. The address offset must be a word-aligned address. Writes within this address offset will force incorrect parity bits to be stored together with the data written. The following reads within this same address offset will thus result in parity errors to be generated.

9.7.1.4 MEM_CTL Register (Offset = Ch) [Reset = 00000000h]

MEM_CTL is shown in Table 9-6.

Return to the Summary Table.

Memory Control
Controls memory initialization

Table 9-6. MEM_CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	MEM_SEL	R/W	0h	Memory Instance Select This field is used to enable/disable initialization of each SRAM instance when triggered using MEM_CTL.MEM_CLR_EN. Each bit corresponds to the respective SRAM instance. bit[x]: 0: Initialization of instance x is disabled 1: Initialization of instance x is enabled
7-2	RESERVED	R	0h	Reserved
1	MEM_BUSY	R	0h	Memory Busy status 0: Memory accepts transfers 1: Memory controller is busy during initialization. Read and write transfers are not performed.
0	MEM_CLR_EN	R/W	0h	Memory Contents Initialization enable Writing 1 to MEM_CLR_EN will start memory initialization. The contents of all byte locations will be initialized to 0x00. MEM_BUSY will be 1 until memory initialization has completed.

9.7.1.5 MEM_STA Register (Offset = 10h) [Reset = 00000000h]

MEM_STA is shown in Table 9-7.

Return to the Summary Table.

Memory Status
Controls memory initialization

Table 9-7. MEM_STA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	MEM_STA	R	0h	Memory Instance Status This field gives the current status of each SRAM instance. When an instance is being initialized the corresponding bit is set to 1, 0 otherwise. bit[x]: 0 : Instance x is in normal mode 1 : Instance x is getting initialized
7-0	RESERVED	R	0h	Reserved

9.7.2 SRAM Registers

Table 9-8 lists the memory-mapped registers for the SRAM registers. All register offset addresses not listed in Table 9-8 should be considered as reserved locations and the register contents should not be modified.

Table 9-8. SRAM Registers

Offset	Acronym	Register Name	Section
0h + formula	BANK0_y	32k SRAM	Section 9.7.2.1
8000h + formula	BANK1_y	32k SRAM	Section 9.7.2.2
00010000h + formula	BANK2_y	32k SRAM	Section 9.7.2.3
00018000h + formula	BANK3_y	32k SRAM	Section 9.7.2.4
00020000h + formula	BANK4_y	32k SRAM	Section 9.7.2.5
00028000h + formula	BANK5_y	32k SRAM	Section 9.7.2.6
00030000h + formula	BANK6_y	32k SRAM	Section 9.7.2.7
00038000h + formula	BANK7_y	32k SRAM	Section 9.7.2.8

Complex bit access types are encoded to fit into small table cells. Table 9-9 shows the codes that are used for access types in this section.

Table 9-9. SRAM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

9.7.2.1 BANK0_y Register (Offset = 0h + formula) [Reset = 00000000h]

BANK0_y is shown in Table 9-10.

Return to the Summary Table.

32k SRAM

Offset = 0h + (y * 4h); where y = 0h to 1FFFh

Table 9-10. BANK0_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.2 BANK1_y Register (Offset = 8000h + formula) [Reset = 00000000h]

BANK1_y is shown in Table 9-11.

Return to the Summary Table.

32k SRAM

Offset = 8000h + (y * 4h); where y = 0h to 1FFFh

Table 9-11. BANK1_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.3 BANK2_y Register (Offset = 00010000h + formula) [Reset = 00000000h]

BANK2_y is shown in Table 9-12.

Return to the Summary Table.

32k SRAM

Offset = 00010000h + (y * 4h); where y = 0h to 1FFFh

Table 9-12. BANK2_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.4 BANK3_y Register (Offset = 00018000h + formula) [Reset = 00000000h]

BANK3_y is shown in Table 9-13.

Return to the Summary Table.

32k SRAM

Offset = 00018000h + (y * 4h); where y = 0h to 1FFFh

Table 9-13. BANK3_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.5 BANK4_y Register (Offset = 00020000h + formula) [Reset = 00000000h]

BANK4_y is shown in Table 9-14.

Return to the Summary Table.

32k SRAM

Offset = 00020000h + (y * 4h); where y = 0h to 1FFFh

Table 9-14. BANK4_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.6 BANK5_y Register (Offset = 00028000h + formula) [Reset = 00000000h]

BANK5_y is shown in Table 9-15.

Return to the Summary Table.

32k SRAM

Offset = 00028000h + (y * 4h); where y = 0h to 1FFFh

Table 9-15. BANK5_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.7 BANK6_y Register (Offset = 00030000h + formula) [Reset = 00000000h]

BANK6_y is shown in Table 9-16.

Return to the Summary Table.

32k SRAM

Offset = 00030000h + (y * 4h); where y = 0h to 1FFFh

Table 9-16. BANK6_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

9.7.2.8 BANK7_y Register (Offset = 00038000h + formula) [Reset = 00000000h]

BANK7_y is shown in Table 9-17.

Return to the Summary Table.

32k SRAM

Offset = 00038000h + (y * 4h); where y = 0h to 1FFFh

Table 9-17. BANK7_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	X	SRAM data

This page intentionally left blank.

Chapter 10

Bootloader

This section describes the bootloader included in the CC13x4x10 and CC26x4x10 device platform.

10.1 Bootloader Functionality 736
10.2 Bootloader Interfaces 736

10.1 Bootloader Functionality

The CC13x4x10 and CC26x4x10 device platform includes a simple ROM-based bootloader that can communicate with an external device over the serial interfaces on the UART0 and SPI0 peripherals. The same communication protocol is used on both serial interfaces. The UART is IP from Arm®.

The main purpose of the ROM bootloader is to support functionality for downloading a Flash image.

10.1.1 Bootloader Disabling

The ROM Bootloader supports commands that can read the Flash image. Due to this read capability and its potential to compromise the device's code to a third party, a secure measure for disabling the bootloader has been implemented. If the bootloader is disabled using the CCFG BOOTLOADER_ENABLE parameter, the bootloader is unable to execute any commands. This prevents attackers from using the bootloader if the program counter (PC) of the Arm® Cortex®-M33F processor is forced to execute from bootloader code. In TI distributed software (SimpleLink™ CC13xx and CC26xx software development kit (SDK)), the CCFG parameters are set at compile time and linked into the CCFG region (0x50000000). The CCFG BOOTLOADER_ENABLE parameter is configured by the value of the SET_CCFG_BL_CONFIG_BOOTLOADER_ENABLE define. In supported projects, this value can be updated in the .syscfg file of the project.

Note

Even if the bootloader is disabled, it can still execute the CMD_GET_STATUS command. This makes it possible to verify that a CMD_DOWNLOAD_CRC command has executed correctly (even if the downloaded Flash image contains CCFG data that disables the bootloader). If any command other than CMD_GET_STATUS is sent to the device while the bootloader is disabled, the bootloader will ignore it.

10.1.2 Bootloader Backdoor

To enter the ROM bootloader even when a valid image is in Flash, a bootloader backdoor is implemented. The CCFG parameter BL_ENABLE can enable this backdoor. The backdoor functionality uses a configurable I/O pin (CCFG parameter BL_PIN_NUMBER) and a configurable I/O pin level (CCFG parameter BL_LEVEL).

If backdoor functionality is enabled, externally applying the configurable signal level on the configurable I/O pin can force ROM bootloader entry upon reset. If the backdoor is enabled and a valid Flash image is present, start-up code checks the level of the I/O pin. If the configured I/O pin level matches the configured signal level, the ROM bootloader does not transfer control to the Flash image.

If the backdoor pin configuration matches one of the UART0 or SPI0 pins, the external user must de-assert the backdoor signal before transmitting on the UART0 or SPI0 interface.

In TI distributed software, the CCFG parameters called BL_ENABLE, BL_PIN_NUMBER and BL_LEVEL are configured by the values of the following defines of the CCFG region.

- SET_CCFG_BL_CONFIG_BL_ENABLE
- SET_CCFG_BL_CONFIG_BL_PIN_NUMBER
- SET_CCFG_BL_CONFIG_BL_LEVEL

Section 11.2 shows the BL_CONFIG parameter layout in CCFG.

Note

When using the bootloader backdoor functionality, the pin configured as backdoor (BL_PIN_NUMBER) will be configured to enable a pull level opposite to the configured I/O pin level while checking the backdoor level.

10.2 Bootloader Interfaces

The bootloader communicates with an external device over a 2-pin UART or 4-pin SPI interface. the communication protocol and transport layers are described in the following subsections.

10.2.1 Packet Handling

The bootloader uses well-defined packets to ensure reliable communications with the external host (a computer or an external processor). All communications use these well-defined packets, with the exception of the UART automatic baud (see Section 10.2.2.1). The packets are always acknowledged or rejected by the communicating devices with defined acknowledge (ACK) and not-acknowledge (NACK) bytes.

The packets use the same format for receiving and sending packets. This format includes the method to acknowledge successful or unsuccessful reception of a packet.

While the actual signaling on the serial ports is different, the packet format remains the same for supported UART and SPI interfaces.

Packet send and receive must adhere to the simple protocol shown in Figure 10-1.

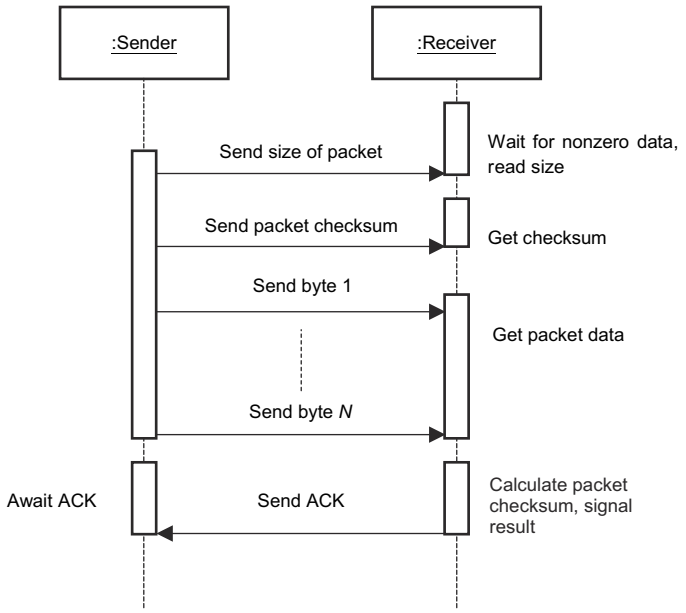


Figure 10-1. Sequence Diagram for Send and Receive Protocol

Perform the following steps to successfully send a packet:

1. Send the size of the packet to be transferred to the device. The size is always the size of the data + 2 with truncation to 8 bits. This implies there is a maximum packet size of 256 bytes.
2. Send the checksum of the data buffer to ensure proper transmission of the command. The checksum algorithm is a sum of the data bytes again truncated to 8 bits.
3. Send the actual data bytes
4. Wait for a single-byte ACK from the device that the data was properly received or that a transmission error was detected.

Perform the following steps to successfully receive a packet:

1. Wait for nonzero data to be returned from the device. This is important as the device may send zero bytes between the sent and received data packet. The first nonzero byte received is the size of the packet that is being received.
2. Read the next byte, which is the checksum of for the packet.
3. Read the data bytes from the device. During the data transfer phase, packet size minus 2 bytes is sent. For example, if the packet size was 3, then there is only 1 byte of data to be received.
4. Calculate the checksum of the data bytes and verify it matches the checksum received in the packet.
5. Send an ACK byte or a NACK byte to the device to indicate the successful or unsuccessful reception of the packet.

ACK bytes are sent out whenever a packet is successfully received and verified by the receiver. A NACK byte is sent out whenever a sent packet is detected to have an error, usually as the result of a checksum error or just malformed data in the packet, which allows the sender to retransmit the previous packet.

To illustrate packet handling, the basic packet format is shown in Figure 10-2.

In Figure 10-2, the top line shows the device that is transmitting data; the bottom line is the response from the other device.

In this case, a 6-byte packet is sent with the data shown in Figure 10-2. This data results in a checksum of 0x48+0x6F+0x6C+0x61 which, when truncated to 8 bits, is 0x84. The first byte transmitted holds the size of the packet in number of bytes. Then the checksum byte is transmitted. The next bytes to go out are the 4 data bytes in this packet. The transmitter is allowed to send zeros until a nonzero response is received, that is necessary to SPI and is allowed by the UART interface. The receiver is allowed to return zeros until it is ready to ACK or NACK the packet that is being sent. Neither device transfers a nonzero byte until it has received a response after transmitting a packet.

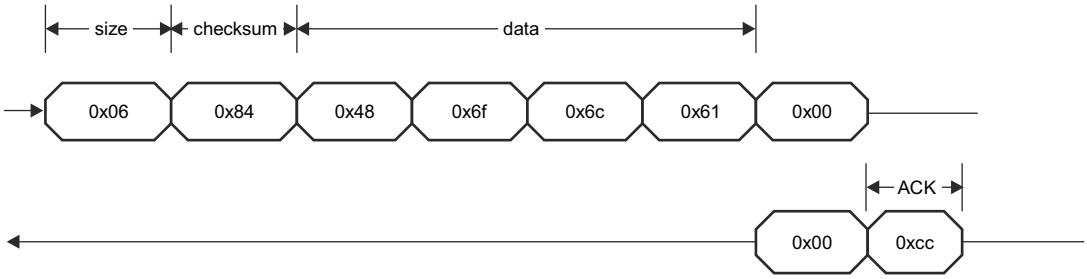


Figure 10-2. Serial Bus Packet Format

10.2.1.1 Packet Acknowledge and Not-Acknowledge Bytes

Table 10-1 shows the defined values for ACK and NACK bytes.

Table 10-1. Protocol Acknowledge and Not-Acknowledge Bytes

Protocol Byte	Value
ACK	0xCC
NACK	0x33

10.2.2 Transport Layer

The bootloader supports the UART0 and SPI0 ports, which are available on the CC13x4x10 and CC26x4x10 device platform. The SPI0 port has the advantage of supporting higher and more flexible data rates, but it also requires more connections to the CC13x4x10 and CC26x4x10 device platform. The UART0 has the disadvantage of having slightly lower and possibly less flexible rates. However, the UART0 requires fewer pins and can be easily implemented with any standard UART connection.

Table 10-2 specifies which serial interface signals are configured to specific DIOs. These pins are fixed and cannot be reconfigured.

Table 10-2. DIO Configuration of Serial Interfaces

Signal	CC26x4x10	CC1314R10	CC1354x10
UART0 RX	12	2	12
UART0 TX	13	3	13
SPI0_MISO	8	8	8
SPI0_MOSI	9	9	9
SPI0_CLK	10	10	10
SPI0_CS	11	11	11

The bootloader initially configures only the input pins on the two serial interfaces. By default, all I/O pins have their input buffers disabled, so the bootloader configures the required pins to be input pins so that the bootloader interface is not accessible from the host before this point in time. For this initial configuration of input pins, the firmware configures the IOC to route the input signals listed in Table 10-2 to their corresponding peripheral signals.

The bootloader selects the interface that is the first to be accessed by the external device. Once selected, the output pin for the selected interface (TX for UART0 or MOSI for SPI0) is configured and the other interface is disabled. To switch to the other interface, the CC13x4x10 and CC26x4x10 device platform must be reset. The delayed configuration of the TX pin imposes special consideration on the SPI0 host device regarding the transfer of the first byte of the first packet (see Section 10.2.2.2).

10.2.2.1 UART Transport

The connections required to use the UART port are the following two pins:

- UART0 TX
- UART0 RX

The device communicating with the bootloader drives the UART0 RX pin on the CC13x4x10 and CC26x4x10 device platform, while the CC13x4x10 and CC26x4x10 device platform drives the UART0 TX pin.

While the baud rate is flexible, the UART serial format is fixed at 8 data bits, no parity, and 1 stop bit. The bootloader automatically detects the baud rate for communication (see Section 10.2.2.1.1).

10.2.2.1.1 UART Baud Rate Automatic Detection

The bootloader provides a method to automatically detect the UART baud rate being used to communicate with it. To synchronize with the host, the bootloader must receive 2 bytes with the value of 0x55. If the synchronization succeeds, the bootloader returns an ACK consisting of 2 bytes with the values of 0x00 and 0xCC.

If synchronization fails, the bootloader waits for the next synchronization attempt. In the automatic detection function, the UART0 RX pin is monitored for edges using GPIO interrupts. When enough edges are detected, the bootloader determines the ratio of baud rate and frequency needed to program the UART.

The UART module system clock must be at least 16 times the baud rate; thus the maximum theoretical baud rate is 3 Mbaud (48 MHz divided by 16), but is limited to 1.2 Mbaud by the firmware that performs the detection of the host transfer rate.

10.2.2.2 SPI Transport

The connections required to use the SPI port are the following four pins:

- SPI0_CLK
- SPI0_CS
- SPI0_MOSI
- SPI0_MISO

The device communicating with the bootloader drives the SPI0_CLK, SPI0_MOSI and SPI0_CS pins, while the CC13x4x10 and CC26x4x10 device platform drives the SPI0_MISO pin.

The format used for SPI communications is the Motorola format with SPH set to 1 and SPO set to 1 (see Figure 23-9 for more information on this format). The SPI interface has a hardware requirement that limits the maximum rate of SPI0_CLK to be at most 1/12 of the frequency of the SPI module clock

The host device must take special consideration (regarding the use of the SPI interface) due to the bootloader not configuring any output pins before the host device has selected a serial interface.

Note

On the first packet transferred by the host device, no data is received from the bootloader while the bootloader clocks out the bits in the first byte of the packet.

Once the bootloader detects that 1 byte has been received on the SPI interface, the bootloader finally configures the SPI0_MISO output pin.

Before transmitting the next byte in the first packet, the host device must include a small delay to ensure that the bootloader has completed the configuration of the SPI0_MISO output pin.

10.2.3 Serial Bus Commands

Table 10-3 lists the commands supported by the custom protocol on the UART0 and SPI0 bootloader interfaces.

Each command is transferred within a protocol packet. The first 2 bytes within a packet are the size byte followed by the checksum byte. The third byte holds the command value that identifies the command; the values for all the supported commands are listed in the Command ID column of Table 10-3. The remaining bytes within the packet are command parameters. See Section 10.2.3.1 through Section 10.2.3.13 for a complete description of the command byte and parameter bytes for each command.

The following subsections specify the individual bytes within the protocol packets for each command.

Table 10-3. Supported Bootloader Commands

Command	Command ID	Bytes in Packet	Description
COMMAND_PING	0x20	3	Receives an acknowledge from the bootloader indicating that communication has been established.
COMMAND_DOWNLOAD	0x21	11	Prepares Flash programming. Specifies from where to program data in Flash and how many bytes will be sent by the COMMAND_SEND_DATA commands that follow.
COMMAND_GET_STATUS	0x23	3	Returns the status of the last command that was issued. Typically, this command must be received by the bootloader after every command is sent to ensure that the previous command was successful. For defined status values, see Table 10-4. The status is returned within a protocol packet of 3 bytes.
COMMAND_SEND_DATA	0x24	4 to 255	Transfers data and programs Flash. Transferring data which is programmed into Flash following a COMMAND_DOWNLOAD command or another COMMAND_SEND_DATA command. The number of data bytes to be programmed in Flash can be 1 to 252 (maximum data load in packet). If more data are downloaded by the COMMAND_SEND_DATA commands than are specified by the COMMAND_DOWNLOAD command, an error status is generated.

Table 10-3. Supported Bootloader Commands (continued)

Command	Command ID	Bytes in Packet	Description
COMMAND_RESET	0x25	3	Performs a system reset. For details, see Section 7.7.1.2.
COMMAND_SECTOR_ERASE	0x26	7	Erases one sector within the Flash main bank. The sector to erase is specified by the sector start address. Only Flash sectors not protected by write-protect bits in FCFG1 and CCFG are erased. If the CCFG sector is selected, the content of CCFG will be reset to the same values as when the device was delivered from TI.
COMMAND_CRC32	0x27	15	Calculates CRC32 over a specified memory area. The number of reads per memory location is specified.
COMMAND_GET_CHIP_ID	0x28	3	Returns the 32-bit USER CODE from the AON_PMCTL JTAGUSERCODE register with MSB first. The ID is returned within a protocol packet.
COMMAND_MEMORY_READ	0x2A	9	Reads a specified number of elements with a specified access width (8 bits or 32 bits) from a specified memory-mapped start address. The requested amount of data must be less than the maximum size of a communication packet.
COMMAND_MEMORY_WRITE	0x2B	9 to 255	Writes the received data in accesses with a specified width (8 or 32 bits) from a specified memory-mapped start address. Data to be written must be contained in same packet as the command.
COMMAND_BANK_ERASE	0x2C	3	Performs an erase of all of the customer-accessible Flash sectors not protected by FCFG1 and CCFG write-protect bits. No erase operation is performed if the CCFG parameter BANK_ERASE_DIS is cleared. Because the CCFG sector might be erased, the content of CCFG will be reset to the same values as when the device was delivered from TI.
COMMAND_SET_CCFG	0x2D	11	Writes the CC13x4x10 and CC26x4x10-defined CCFG fields to the Flash CCFG area with the values received in the data bytes of this command. This command abstracts the user from detailed knowledge concerning which physical addresses within the Flash CCFG holding the defined CCFG fields.
COMMAND_DOWNLOAD_CRC	0x2F	15	Prepares Flash programming with the specified CRC32 value. Specifies from where to program data in the Flash main bank, how many bytes will be sent by the COMMAND_SEND_DATA commands that follow, and the 32-bits CRC value covering the total number of bytes to be programmed.

10.2.3.1 COMMAND_PING

The COMMAND_PING command receives an ACK from the bootloader, indicating that communication has been established. This command is a single byte.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[3];
ucPacket[0] = <size=3>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_PING;
```

10.2.3.2 COMMAND_DOWNLOAD

The COMMAND_DOWNLOAD command is sent to the bootloader to indicate where to store data in Flash and how many bytes will be sent by the COMMAND_SEND_DATA commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming image data into, which the second is the 32-bit size of the image data that will be sent. This command must be followed by a COMMAND_GET_STATUS command to ensure that the image address and image size are valid for the device. On the CC13x4x10 and CC26x4x10 devices, the Flash starts at address 0x00000000. The command does not perform any kind of erase operation; it only prepares for the following Flash programming performed by COMMAND_SEND_DATA commands. Required Flash erase can be done by the COMMAND_BANK_ERASE and COMMAND_SECTOR_ERASE commands.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[11];
ucPacket[0] = <size=11>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_DOWNLOAD;
ucPacket[3] = <image address [31:24]>;
ucPacket[4] = <image address [23:16]>;
ucPacket[5] = <image address [15:8]>;
ucPacket[6] = <image address [7:0]>;
ucPacket[7] = <image size [31:24]>;
ucPacket[8] = <image size [23:16]>;
ucPacket[9] = <image size [15:8]>;
ucPacket[10] = <image size [7:0]>;
```

10.2.3.3 COMMAND_GET_STATUS

The COMMAND_GET_STATUS command returns the status of the last command that was issued. Typically, this command is received after every other command is sent to ensure that the previous command was successful; or, if the command failed, to properly respond to a failure. The bootloader responds by sending a 3-byte packet with the size byte, checksum byte and 1 byte of the current-status value.

The bootloader then waits for an ACK from the host as a confirmation that the packet was received.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[3];
ucPacket[0] = <size=3>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_GET_STATUS;
```

Table 10-4 lists the definitions for the possible status values that can be returned from the bootloader when a COMMAND_GET_STATUS command is sent to the bootloader.

Table 10-4. Defined Status Values

Status Definition	Value	Description
COMMAND_RET_SUCCESS	0x40	Status for successful command
COMMAND_RET_UNKNOWN_CMD	0x41	Status for unknown command
COMMAND_RET_INVALID_CMD	0x42	Status for invalid command (in other words, incorrect packet size)

Table 10-4. Defined Status Values (continued)

Status Definition	Value	Description
COMMAND_RET_INVALID_ADR	0x43	Status for invalid input address
COMMAND_RET_FLASH_FAIL	0x44	Status for failing Flash erase or program operation

10.2.3.4 COMMAND_SEND_DATA

The COMMAND_SEND_DATA command must only follow a COMMAND_DOWNLOAD or another COMMAND_SEND_DATA command, if more data is needed. Consecutive COMMAND_SEND_DATA commands automatically increment the address and continue programming from the previous location.

The command terminates programming when the number of bytes indicated by the COMMAND_DOWNLOAD command are received.

The bootloader sends the ACK in response to the command after the actual programming is complete. Each time this function is called, enter a COMMAND_GET_STATUS command to ensure that the data was successfully programmed into the Flash. If the bootloader sends a NACK response to this command, the bootloader does not increment the current address, which allows for retransmission of the previous data.

Remember the size byte of the packet is an 8-bit value. This provides the limit for the packet length of 256 bytes.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[4-255];
ucPacket[0] = <size>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_SEND_DATA;
ucPacket[3] = <image data[0]>;
ucPacket[4] = <image data[1]>;
ucPacket[5] = <image data[2]>;
ucPacket[6] = <image data[3]>;
ucPacket[7] = <image data[4]>;
...
...
ucPacket[<size-1>] = <image data[<size-3>];
```

10.2.3.5 COMMAND_RESET

The COMMAND_RESET command tells the bootloader to perform a system reset. Use this command after downloading a new Flash image to the CC13x4x10 and CC26x4x10 devices to cause the new application to start from a reset. The normal boot sequence occurs and the Flash image runs similarly as if a hardware reset had occurred. Also, use this command to reset the bootloader if a critical error occurs and the host device wants to restart communication with the bootloader.

The bootloader responds with an ACK signal to the host device before actually executing the system reset. This ACK signal informs the host that the command was received successfully, and the CC13x4x10 and CC26x4x10 devices are then reset.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[3];
ucPacket[0] = <size=3>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_RESET;
```

10.2.3.6 COMMAND_SECTOR_ERASE

The COMMAND_SECTOR_ERASE command erases a specified Flash sector.

The command consists of one 32-bit value that is transferred MSB first. The 32-bit value is the start address of the Flash sector to be erased.

The bootloader responds with an ACK signal to the host device after the actual erase operation is performed.

On the CC13x4x10 and CC26x4x10 device platform, the Flash starts at address 0x00000000 and it has sectors of 2 KB each.

Note

Sectors protected by write-protect bits in FCFG1 and CCFG are not erased.

If the sector address of the CCFG sector is specified, the actual erase is followed by a set of default CCFG values being written into the CCFG so that the bootloader is still enabled upon reset.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[7];
ucPacket[0] = <size=7>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_SECTOR_ERASE;
ucPacket[3] = <Sector Address [31:24]>;
ucPacket[4] = <Sector Address [23:16]>;
ucPacket[5] = <Sector Address [15:8]>;
ucPacket[6] = <Sector Address [7:0]>;
```

10.2.3.7 COMMAND_CRC32

The COMMAND_CRC32 command checks a Flash area using CRC32. The command consists of three 32-bit values that are all transferred MSB first. The first 32-bit value is the address in memory from where the CRC32 calculation starts. The second 32-bit value is the number of bytes comprised by the CRC32 calculation and the third 32-bit value is the number of read repeats for each data location. A read repeat count of 0x00000000 causes the checksum to be generated by a read of all data locations only once. The command sends the ACK signal in response to the command after the actual CRC32 calculation. The result is finally returned as 4 bytes (MSB first) in a 6-byte packet. The bootloader then waits for an ACK signal from the host as a confirmation that the packet was received. The second parameter that holds the number of bytes must be higher than eight. If not, the returned checksum is 0xFFFFFFFF.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[15];
ucPacket[0] = <size=15>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_CRC32;
ucPacket[3] = <Data Address [31:24]>;
ucPacket[4] = <Data Address [23:16]>;
ucPacket[5] = <Data Address [15:8]>;
ucPacket[6] = <Data Address [7:0]>;
ucPacket[7] = <Data Size [31:24]>;
ucPacket[8] = <Data Size [23:16]>;
ucPacket[9] = <Data Size [15:8]>;
ucPacket[10] = <Data Size [7:0]>;
ucPacket[11] = <Read Repeat Count [31:24]>;
ucPacket[12] = <Read Repeat Count [23:16]>;
ucPacket[13] = <Read Repeat Count [15:8]>;
ucPacket[14] = <Read Repeat Count [7:0]>;
```

10.2.3.8 COMMAND_GET_CHIP_ID

The COMMAND_GET_CHIP_ID command makes the bootloader return the value of the 32-bit user ID from the AON_PMCTL:JTAGUSERCODE register. The bootloader first responds by sending the ACK signal in response to the command; then the bootloader sends a packet of 6 bytes comprised of the size byte, the checksum byte, and the 4 bytes (MSB first) holding the user ID.

The bootloader then waits for an ACK signal from the host as a confirmation that the packet was received.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[7];
ucPacket[0] = <size=7>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_GET_CHIP_ID;
```

10.2.3.9 COMMAND_MEMORY_READ

The COMMAND_MEMORY_READ command requests the bootloader to read one or multiple 8- or 32-bit memory elements. The command sends the number of elements, the access type (8 or 32 bits), the start address and returns the read data in the subsequent communications packets. The requested amount of data must be less than the maximum size of a communication packet. The specified Access Type must be either a 0 or 1. The value of 0 forces 8-bit read accesses. The value of 1 forces 32-bit read accesses. The specified Number of Accesses gives the number of 8- or 32-bit read accesses. Maximum value of Number of Accesses is 253 for Access Type = 0. Maximum value for Number of Accesses is 63 for Access Type = 1. The format of the packet including the command is as follows:

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[9];
ucPacket[0] = <size=9>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_MEMORY_READ;
ucPacket[3] = <Memory Mapped Address [31:24]>;
ucPacket[4] = <Memory Mapped Address [23:16]>;
ucPacket[5] = <Memory Mapped Address [15:8]>;
ucPacket[6] = <Memory Mapped Address [7:0]>;
ucPacket[7] = <Access Type [7:0]>;
ucPacket[8] = <Number of Accesses [7:0]>;
```

10.2.3.10 COMMAND_MEMORY_WRITE

The COMMAND_MEMORY_WRITE command requests the bootloader to write 8- or 32-bit data to a specified memory address. Data to be written must be contained in same packet as the command. The access width is given by the specified Access Type. The specified Access Type must be either a 0 or 1. The value of 0 forces 8-bit read accesses. The value of 1 forces 32-bit read accesses. The specified Number of Accesses gives the number of 8- or 32-bit read accesses. The number of data bytes received is given by the packet size byte. Maximum number of data bytes for access width 0 is 247 and 244 for access width 1.

Specific memory mapped areas must not be written to using the COMMAND_MEMORY_WRITE command.

Memory writes to the following memory mapped areas can cause the serial bootloader to end up in a nonfunctional state as these areas are used by the bootloader. This is valid for the following memory mapped areas:

- The lower 4 KB of SRAM (0x20000000 to 0x20000FFF)
- Any hardware registers controlling the functionality of the serial interface (UART or SPI) currently being used by the serial bootloader

Note

The COMMAND_MEMORY_WRITE command cannot be used to write to Flash memory.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[(from 9 to 255)];
ucPacket[0] = <size=(from 9 to 255)>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_MEMORY_WRITE;
ucPacket[3] = <Memory Mapped Address [31:24]>;
ucPacket[4] = <Memory Mapped Address [23:16]>;
ucPacket[5] = <Memory Mapped Address [15:8]>;
ucPacket[6] = <Memory Mapped Address [7:0]>;
ucPacket[7] = <Access Type [7:0]>;
ucPacket[8] = <Data [0]>;
...
...
ucPacket[9 + (size-9)] = <Data [size-8]>;
```

10.2.3.11 COMMAND_BANK_ERASE

The COMMAND_BANK_ERASE command does not perform any erase operation if the CCFG parameter BANK_ERASE_DIS is cleared. When COMMAND_BANK_ERASE is not cleared, this command erases all main bank Flash sectors including CCFG not protected by write-protect bits in FCFG1 and CCFG.

The command sends the ACK in response to the command after the actual erase operation is performed. The actual erase is followed by CCFG values being programmed to the factory default values.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[3];
ucPacket[0] = <size=3>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_BANK_ERASE;
```

Note

The BANK_ERASE command will not erase the CCFG region , due to the CCFG not being in the main Flash region. In order to erase the CCFG, the SECTOR_ERASE command must be sent with the sector address of the CCFG.

10.2.3.12 COMMAND_SET_CCFG

The COMMAND_SET_CCFG command is sent to the bootloader to configure the defined fields in the Flash CCFG area that are read by the ROM boot firmware. The command sends the ACK signal in response to the command after the actual Flash program operation is performed. This command does not execute any erase operation before the write operation.

The command consists of two 32-bit values that are all transferred MSB first. The first 32-bit value is the CCFG Field ID, which identifies the CCFG parameter to be written. The second 32-bit value is the Field Value to be programmed. The command handler masks out Field Value bits not corresponding to the CCFG parameter size.

Note

The COMMAND_SET_CCFG command can only change CCFG parameter value bits from 1 to 0. Attempting to change any bit from 0 to 1 results in an error status that can be observed by a following COMMAND_GET_STATUS command. Only the CCFG fields controlling device configuration during ROM boot, can be written by this command. (fields sequential from BL_CONFIG to until end).

The only way to change CCFG parameter value bits from 0 to 1 is by erasing the complete CCFG Flash sector. The command sends the ACK signal in response to the command after the actual Flash programming has terminated.

The programming operation fails if the CCFG area is write-protected by the protect bit in FCFG1 or in CCFG.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[11];
ucPacket[0] = <size=11>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_SET_CCFG;
ucPacket[3] = <Field Id [31:24]>;
ucPacket[4] = <Field Id [23:16]>;
ucPacket[5] = <Field Id [15:8]>;
ucPacket[6] = <Field Id [7:0]>;
ucPacket[7] = <Field Value [31:24]>;
ucPacket[8] = <Field Value [23:16]>;
ucPacket[9] = <Field Value [15:8]>;
ucPacket[10] = <Field Value [7:0]>;
```

Defined CCFG field IDs with corresponding field values are described in Table 10-5.

Table 10-5. Defined CCFG Field IDs and Field Values

Field ID	Field Value	Description
0: ID_SECTOR_PROT	Bit[31:0]: The number of the Flash sector to be protected from being program and erase	The ID_SECTOR_PROT field is used to update the CCFG:WEPROT_31_O_BY2K field. This allows the first 32 sectors to be write erase protected. This command also sets the sticky sector protect bit in the Flash wrapper registers upon the next boot. A single sector can be protected per ID_SECTOR_PROT update. If a 0xF is passed for instance, the CCFG will be updated to write erase protect the 15th sector.
1: ID_IMAGE_VALID	Bit[31:0]: 0x00000000	For the boot sequence to transfer execution control to a Flash image, this Field Value must be set to the start address of the Flash image vector table
2: ID_TEST_TAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TAP Unlocked	Any other value than 0xC5 forces a locked TAP after a following boot sequence
3: ID_PWRPROF_TAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TAP Unlocked	Any other value than 0xC5 forces a locked TAP after a following boot sequence
4: ID_CPU_DAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = DAP Unlocked	Any other value than 0xC5 forces a locked DAP after a following boot sequence.
5: ID_AON_TAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TAP Unlocked	Any other value than 0xC5 forces a locked TAP after a following boot sequence.
6: ID_PBIST1_TAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TAP Unlocked	Any other value than 0xC5 forces a locked TAP after a following boot sequence
7: ID_PBIST2_TAP_LCK	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TAP unlocked	Any other value than 0xC5 forces a locked TAP after a following boot sequence.
8: ID_BANK_ERASE_DIS	Bit[31:1]: Don't Care Bit[0]: 0 = Bank Erase Disable	If set to 0, the COMMAND_BANK_ERASE bootloader command does not force any erase operation.
9: ID_CHIP_ERASE_DIS	Bit[31:1]: Don't Care Bit[0]: 0 = Chip erase disable	If set to 0, the start-up sequence does not perform any chip erase operation regardless of any chip erase request.
10: ID_TI_FA_ENABLE	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = TI FA enable	Any value other than 0xC5 disables the TIFA enable functionality in a following boot.
11: ID_BL_BACKDOOR_EN	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = Bootloader Backdoor Enable	Any other value than 0xC5 forces the bootloader backdoor to be disabled in a following boot sequence.
12: ID_BL_BACKDOOR_PIN	Bit[31:8]: Don't Care Bit[7:0]: Bootloader Backdoor I/O Pin Number	If the pin number exceeds number of I/O pins on the device, the highest I/O pin number on the device is selected.
13: ID_BL_BACKDOOR_LEVEL	Bit[31:1]: Don't Care Bit[0]: Bootloader backdoor pin active level	0 = Active low 1 = Active high
14: ID_BL_ENABLE	Bit[31:8]: Don't Care Bit[7:0]: Bootloader Enable	Any value other than 0xC5 forces the bootloader to ignore any received command.
15: ID_CCFG_PROT	Bit[31:1]: Don't Care Bit[0]: Write/Erase Protect the CCFG	If set to 0, the CCFG is protected against Write and Erase operations
16: ID_TI_IDAU_CFG_ENABLE	Bit[31:8]: Don't Care Bit[7:0]: 0xC5 = IDAU Configuration Disabled	Any other value than 0xC5 will enable IDAU configuration.

Table 10-5. Defined CCFG Field IDs and Field Values (continued)

Field ID	Field Value	Description
17: ID_TZ_FLASH_NS_BOUND_HIGH	Bit[31:1]: Don't Care Bit[0]: Boundary Address [7]	Flash Non-secure boundary address This address will be written to PRCM:NVMNSADDR.BOUNDARY by ROM Boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5
18: ID_TZ_FLASH_NS_BOUND_LOW	Bit[31:16]: Don't Care Bit[15:10]: Boundary Address [6:0] Bit[9:0]: Don't Care	
19: ID_TZ_FLASH_NSC_BOUND_HIGH	Bit[31:2]: Don't Care Bit[1]: Boundary Address [9] Bit[0]: Boundary Address [8]	Flash Non-secure callable boundary address This address will be written to PRCM:NVMNSCADDR.BOUNDARY by ROM Boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5
20: ID_TZ_FLASH_NSC_BOUND_LOW	Bit[31:8]: Don't Care Bit[7:0]: Boundary Address [7:0]	
21: ID_TZ_SRAM_NS_BOUND_HIGH	Bit[31:2]: Don't Care Bit[1]: Boundary Address [8] Bit[0]: Boundary Address [7]	SRAM Non-secure boundary address This address will be written to PRCM:SRAMNSADDR.BOUNDARY by ROM Boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5
22: ID_TZ_SRAM_NS_BOUND_LOW	Bit[31:2]: Don't Care Bit[7:1]: Boundary Address [6:0] Bit[0]: Don't Care	
23: ID_TZ_SRAM_NSC_BOUND_HIGH	Bit[31:1]: Don't Care Bit[0]: Boundary Address [8]	SRAM Non-secure callable boundary address This address will be written to PRCM:SRAMNSCADDR.BOUNDARY by ROM Boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5
24: ID_TZ_SRAM_NSC_BOUND_LOW	Bit[31:2]: Don't Care Bit[7:0]: Boundary Address [7:0]	
25: ID_CPU_LOCK_LOCKNSVTOR	Bit[31:5]: Don't Care Bit[4]: Lock/Unlock Bit[3:0]: Don't Care	0 = Locked 1 = Unlocked Lock/Unlock the Non-secure vector table base address
26: ID_CPU_LOCK_LOCKSVTAIRCR	Bit[31:4]: Don't Care Bit[3]: Lock/Unlock Bit[2:0]: Don't Care	0 = Locked 1 = Unlocked Lock/Unlock the following regions:. • Secure vector table base address • Secure interrupt priority • Busfault • Hardfault NMI security target
27: ID_CPU_LOCK_LOCKSAU	Bit[31:3]: Don't Care Bit[2]: Lock/Unlock Bit[1:0]: Don't Care	0 = Locked 1 = Unlocked Lock/Unlock the SAU regions
28: ID_CPU_LOCK_LOCKNSMPU	Bit[31:2]: Don't Care Bit[1]: Lock/Unlock Bit[0]: Don't Care	0 = Locked 1 = Unlocked Lock/Unlock the Non-secure MPU
29: ID_CPU_LOCK_LOCKSMPU	Bit[31:1]: Don't Care Bit[0]: Lock/Unlock	0 = Locked 1 = Unlocked Lock/Unlock the Secure MPU
30: ID_DEB_AUTH_INTSPNIDEN	Bit[31:4]: Don't Care Bit[3]: Debug Enable/Disable Bit[2:0]: Don't Care	Internal Secure non-invasive debug enable. Overrides the external Secure non-invasive debug authentication interfaces. This value will be written to CPU_DCB:DAUTHCTRL.INTSPNIDEN by ROM boot FW

Table 10-5. Defined CCFG Field IDs and Field Values (continued)

Field ID	Field Value	Description
31: ID_DEB_AUTH_SPNIDENSEL	Bit[31:3]: Don't Care Bit[2]: Debug Select/De-select Bit[1:0]: Don't Care	Secure non-invasive debug enable select. Selects between DAUTHCTL and the eternal authentication interface for control of Secure non-invasive debug. This value will be written to CPU_DCB:DAUTHCTRL.SPNIDENSEL by ROM boot FW.
32: ID_DEB_AUTH_INTSPIDEN	Bit[31:2]: Don't Care Bit[1]: Debug Enable/Disable Bit[0]: Don't Care	Internal Secure invasive debug enable. Overrides the external Secure invasive debug authentication interfaces. This value will be written to CPU_DCB:DAUTHCTRL.INTSPIDEN by ROM boot FW
33: ID_DEB_AUTH_SPIDENSEL	Bit[31:1]: Don't Care Bit[0]: Debug Select/De-select	Secure invasive debug enable select. Selects between DAUTHCTL and the eternal authentication interface for control of Secure invasive debug. This value will be written to CPU_DCB:DAUTHCTRL.SPIDENSEL by ROM boot FW.
34: ID_BUS_CFG	Bit[31:8]: Don't Care Bit[7:0]: Bus CFG	Bus interconnect security and firewall configuration. This value will be written to PRCM:BUSSECCFG.BUS_CFG by ROM boot FW.

10.2.3.13 COMMAND_DOWNLOAD_CRC

The COMMAND_DOWNLOAD_CRC command is sent to the bootloader to indicate where to program data in Flash, how many bytes will be sent by the COMMAND_SEND_DATA commands that follow, and the expected 32-bit CRC value covering the total number of data bytes to be stored in Flash. The command consists of three 32-bit values that are all transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent, and the third is the 32-bit CRC value covering the total number of bytes to program in Flash.

This command should be followed by a COMMAND_GET_STATUS to ensure that the program address and program size were valid for the device.

The Flash starts at address 0x00000000.

The COMMAND_DOWNLOAD_CRC command does not perform any kind of erase operation; it only prepares for following NVM programming performed by the COMMAND_SEND_DATA commands. Required Flash erase can be done by the COMMAND_BANK_ERASE and COMMAND_SECTOR_ERASE commands. The final COMMAND_SEND_DATA command that programs the last bytes of the total number of bytes specified by the COMMAND_DOWNLOAD_CRC command, (after the programming) will calculate the CRC32 value covering the Flash area specified by the COMMAND_DOWNLOAD_CRC command and check if the calculated CRC32 value equals the specified 32-bit CRC value. If these values are not equal, the complete Flash area that has been programmed will be erased before the COMMAND_SEND_DATA command responds with an ACK. In this scenario, a following COMMAND_GET_STATUS command will report a COMMAND_RET_FLASH_FAIL status.

Note

Due to the CRC calculation and possible Flash erase operation during execution of the last COMMAND_SEND_DATA command, the COMMAND_DOWNLOAD_CRC command will have a delayed ACK response.

The format of the packet including the command ID is as follows:

```
unsigned char ucPacket[15];
ucPacket[0] = <size=15>;
ucPacket[1] = <checksum>;
ucPacket[2] = COMMAND_DOWNLOAD_CRC;
ucPacket[3] = <image address [31:24]>;
ucPacket[4] = <image address [23:16]>;
ucPacket[5] = <image address [15:8]>;
ucPacket[6] = <image address [7:0]>;
ucPacket[7] = <image size [31:24]>;
ucPacket[8] = <image size [23:16]>;
ucPacket[9] = <image size [15:8]>;
ucPacket[10] = <image size [7:0]>;
ucPacket[11] = <image crc [31:24]>;
ucPacket[12] = <image crc [23:16]>;
ucPacket[13] = <image crc [15:8]>;
ucPacket[14] = <image crc [7:0]>;
```

Chapter 11

Device Configuration

This chapter describes the device configuration areas. The factory configuration (FCFG) and customer configuration (CCFG) areas are located in Flash. The FCFG is set by Texas Instruments during device production and contains device-specific trim values and configuration. The CCFG is set at compile time and contains configuration parameters for the ROM boot code, device hardware, and device firmware.

Note

The FCFG and CCFG might be updated, so refer to the latest SDK version for the register map (SimpleLink™ CC13xx and CC26xx software development kit (SDK)).

11.1 Customer Configuration (CCFG) 754
11.2 CCFG Registers 756
11.3 Factory Configuration (FCFG) 788
11.4 FCFG1 Registers 789

11.1 Customer Configuration (CCFG)

The following list shows some of the parameters configurable through CCFG. For a complete overview, refer to the ccfg file in the SimpleLink™ CC13xx and CC26xx software development kit (SDK).

- Image valid parameter (normally set by the programming tool)
- Failure analysis access configuration
- Custom MAC addresses
- Bootloader configuration
- TAP and DAP access configuration
- Debug authentication control
- Trustzone and interconnect security configuration
- CPU lock options
- SRAM parity configuration (only for devices with 256 KB SRAM)
- CC13x4x10 device: Configure VDDR boost mode to enable up to +14 dBm output power

In TI distributed software, the CCFG parameters are set at compile time in the ccfg.c file. The CCFG settings are set by default to allow full debugging of the device. The CCFG settings are not recommended for production.

For the CC13x4x10 device only:

To enable output power of +14 dBm, the CCFG_FORCE_VDDR_HH define must be set to 1 in ccfg.c distributed in the TI Simplelink SDK. If CCFG_FORCE_VDDR_HH is set to 0 the maximum possible output power is +12.5 dBm.

11.1.1 CCFG Recommendations for Final Production

TI recommends configuring a device for final production with the steps that follow:

1. The following defines in ccfg.c must be set to 0x00 to disallow access to Flash contents through the bootloader interface.
   - SET_CCFG_BL_CONFIG_BOOTLOADER_ENABLE
   - SET_CCFG_BL_CONFIG_BL_ENABLE
2. The SET_CCFG_CCFG_TI_OPTIONS_TI_FA_ENABLE define in ccfg.c must be set to 0x00 to disallow failure analysis access by TI.
3. The following defines in ccfg.c must be set to 0x00 to individually disallow access to the JTAG access ports:
   - SET_CCFG_CCFG_TAP_DAP_0_PWRPROF_TAP_ENABLE
   - SET_CCFG_CCFG_TAP_DAP_0_TEST_TAP_ENABLE
   - SET_CCFG_CCFG_TAP_DAP_0_CPU_DAP_ENABLE
   - SET_CCFG_CCFG_TAP_DAP_1_AON_TAP_ENABLE
   - SET_CCFG_CCFG_TAP_DAP_1_PBIST1_TAP_ENABLE
   - SET_CCFG_CCFG_TAP_DAP_1_PBIST2_TAP_ENABLE
4. To enable power profiling with EnergyTrace™ software, the SET_CCFG_CCFG_TAP_DAP_0_PWRPROF_TAP_ENABLE define in ccfg.c must be set to 0xC5.
5. The SET_CCFG_IMAGE_VALID_CONF_IMAGE_VALID define in ccfg.c must be set to the address of the vector table of the Flash image to pass control to the programmed image in Flash at boot. Most standard Flash images will have the vector table located at address 0x00000000.
6. Optionally, the SET_CCFG_ERASE_CONF_CHIP_ERASE_DIS_N define in ccfg.c can be set to 0x0 to disallow erasing of the Flash when Chip Erase is requested by JTAG.
7. Use the SET_CCFG_CCFG_PROT_n defines in ccfg.c to program and erase protect the sectors of Flash that are not designed to be updated in-system by the final product. Any bit in the define set to 0 will force that the corresponding Flash sector number is program and erase protected.
8. Set SET_CCFG_DEB_AUTH_CFG_SPIDENSEL to 1 and set SET_CCFG_DEB_AUTH_CFG_INTSPIDEN to 0 to disallow Secure debug.
9. SET_CCFG_BUS_CFG must be set to 0xFFFFFFFF.

10. The following defines should be individually set to 0x1 to allow VTOR mapping and MPU re-configuration:
    - SET_CCFG_CPU_LOCK_CFG_LOCKNSVTOR
    - SET_CCFG_CPU_LOCK_CFG_LOCKSVTAIRCR
    - SET_CCFG_CPU_LOCK_CFG_LOCKSAU
    - SET_CCFG_CPU_LOCK_CFG_LOCKNSMPU
    - SET_CCFG_CPU_LOCK_CFG_LOCKSMPU
11. If default Trustzone boundaries need to be specified, then SET_CCFG_CCFG_TI_OPTIONS_IDAU_CFG_ENABLE must be set to 0x0, and the following watermark configurations should be specified:
    - SET_CCFG_TRUSTZONE_FLASH_CFG_NSADDR_BOUNDARY
    - SET_CCFG_TRUSTZONE_FLASH_CFG_NSCADDR_BOUNDARY
    - SET_CCFG_TRUSTZONE_SRAM_CFG_NSADDR_BOUNDARY
    - SET_CCFG_TRUSTZONE_SRAM_CFG_NSCADDR_BOUNDARY
12. SRAM parity is recommended to be enabled by setting SET_CCFG_SRAM_CFG_PARITY_DIS to 0x0. Parity can be disabled only if increased soft fault rate is acceptable. Applications with run time security must have parity enabled.

Note

Enabling some of the functionalities in the ENABLE fields in CCFG are contingent on the corresponding ENABLE field in FCFG that has been set to enabled by the TI production test. This is the case for the access configuration of the TEST_TAP access port, for example. In the products where the TEST_TAP access is not enabled in FCFG, the value in the corresponding CCFG field set by the SET_CCFG_CCFG_TAP_DAP_0_TEST_TAP_ENABLE define in ccfg.c, is ignored and the functionality is disabled.

11.2 CCFG Registers

Table 11-1 lists the memory-mapped registers for the CCFG registers. All register offset addresses not listed in Table 11-1 should be considered as reserved locations and the register contents should not be modified.

Table 11-1. CCFG Registers

Offset	Acronym	Register Name	Section
0h	SIZE_AND_DIS_FLAGS	CCFG Size and Disable Flags	Section 11.2.1
4h	MODE_CONF	Mode Configuration 0	Section 11.2.2
8h	MODE_CONF_1	Mode Configuration 1	Section 11.2.3
Ch	VOLT_LOAD_0	Voltage Load 0	Section 11.2.4
10h	VOLT_LOAD_1	Voltage Load 1	Section 11.2.5
14h	EXT_LF_CLK	Extern LF clock configuration	Section 11.2.6
18h	IEEE_MAC_0	IEEE MAC Address 0	Section 11.2.7
1Ch	IEEE_MAC_1	IEEE MAC Address 1	Section 11.2.8
20h	IEEE_BLE_0	IEEE BLE Address 0	Section 11.2.9
24h	IEEE_BLE_1	IEEE BLE Address 1	Section 11.2.10
28h	BL_CONFIG	Bootloader Configuration	Section 11.2.11
2Ch	ERASE_CONF	Erase Configuration	Section 11.2.12
30h	ERASE_CONF_1	Erase Configuration 1	Section 11.2.13
34h	CCFG_TI_OPTIONS	TI Options	Section 11.2.14
38h	CCFG_TAP_DAP_0	Test Access Points Enable 0	Section 11.2.15
3Ch	CCFG_TAP_DAP_1	Test Access Points Enable 1	Section 11.2.16
40h	IMAGE_VALID_CONF	Image Valid	Section 11.2.17
44h	CCFG_WEPROT_31_0_BY2K	Protect Sectors 0-31	Section 11.2.18
48h	CCFG_WEPROT_SPARE_1	Spare register for WriteErase configuration	Section 11.2.19
4Ch	CCFG_WEPROT_SPARE_2	Spare register for WriteErase configuration	Section 11.2.20
50h	CCFG_WEPROT_SPARE_3	Spare register for WriteErase configuration	Section 11.2.21
54h	TRUSTZONE_FLASH_CFG	Trustzone configuration register for flash	Section 11.2.22
58h	TRUSTZONE_SRAM_CFG	Trustzone configuration register for MCU SRAM	Section 11.2.23
5Ch	SRAM_CFG	Configuration register for MCU SRAM	Section 11.2.24
64h	CPU_LOCK_CFG	Configuration register for MCU CPU lock options	Section 11.2.25
68h	DEB_AUTH_CFG	Configuration register for debug authentication	Section 11.2.26
6Ch	CKEY0	Customer key	Section 11.2.27
70h	CKEY1	Customer key	Section 11.2.28
74h	CKEY2	Customer key	Section 11.2.29
78h	CKEY3	Customer key	Section 11.2.30

Complex bit access types are encoded to fit into small table cells. Table 11-2 shows the codes that are used for access types in this section.

Table 11-2. CCFG Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Reset or Default Value		
-n		Value after reset or the default value

11.2.1 SIZE_AND_DIS_FLAGS Register (Offset = 0h) [Reset = FFFFFFFFh]

SIZE_AND_DIS_FLAGS is shown in Table 11-3.

Return to the Summary Table.

CCFG Size and Disable Flags

Table 11-3. SIZE_AND_DIS_FLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SIZE_OF_CCFG	R	FFFFh	Total size of CCFG in bytes.
15-4	DISABLE_FLAGS	R	FFFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/ default value may result in undefined behavior.
3	DIS_TCXO	R	1h	Deprecated. Must be set to 1.
2	DIS_GPRAM	R	1h	Disable GPRAM (or use the 8K VIMS RAM as CACHE RAM). 0: GPRAM is enabled and hence CACHE disabled. 1: GPRAM is disabled and instead CACHE is enabled (default). Notes: - Disabling CACHE will reduce CPU execution speed (up to 60%). - GPRAM is 8 K-bytes in size and located at 0x11000000-0x11001FFF if enabled. See: VIMS:CTL.MODE
1	DIS_ALT_DCDC_SETTING	R	1h	Disable alternate DC/DC settings. 0: Enable alternate DC/DC settings. 1: Disable alternate DC/DC settings. See: MODE_CONF_1.ALT_DCDC_VMIN MODE_CONF_1.ALT_DCDC_DITHER_EN MODE_CONF_1.ALT_DCDC_IPEAK NOTE! The DriverLib function SysCtrl_DCDC_VoltageConditionalControl() must be called regularly to apply this field.
0	DIS_XOSC_OVR	R	1h	Disable XOSC override functionality. 0: Enable XOSC override functionality. 1: Disable XOSC override functionality. See: MODE_CONF_1.DELTA_IBIAS_INIT MODE_CONF_1.DELTA_IBIAS_OFFSET MODE_CONF_1.XOSC_MAX_START

11.2.2 MODE_CONF Register (Offset = 4h) [Reset = FFFFFFFFh]

MODE_CONF is shown in Table 11-4.

Return to the Summary Table.

Mode Configuration 0

Table 11-4. MODE_CONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	VDDR_TRIM_SLEEP_DELTA	R	Fh	Signed delta value to apply to the VDDR_TRIM_SLEEP target, minus one. See FCFG1:VOLT_TRIM.VDDR_TRIM_SLEEP_H. 0x8 (-8) : Delta = -7 ... 0xF (-1) : Delta = 0 0x0 (0) : Delta = +1 ... 0x7 (7) : Delta = +8
27	DCDC_RECHARGE	R	1h	DC/DC during recharge in powerdown. 0: Use the DC/DC during recharge in powerdown. 1: Do not use the DC/DC during recharge in powerdown (default). NOTE! The DriverLib function SysCtrl_DCDC_VoltageConditionalControl() must be called regularly to apply this field.
26	DCDC_ACTIVE	R	1h	DC/DC in active mode. 0: Use the DC/DC during active mode. 1: Do not use the DC/DC during active mode (default). NOTE! The DriverLib function SysCtrl_DCDC_VoltageConditionalControl() must be called regularly to apply this field.
25	VDDR_EXT_LOAD	R	1h	For TI internal use only.
24	VDDS_BOD_LEVEL	R	1h	VDDS BOD level. 0: VDDS BOD level is 2.0V (necessary for external load mode, or for maximum PA output power on CC13x4x10). 1: VDDS BOD level is 1.8V (or 1.65V for external regulator mode) (default).
23-22	SCLK_LF_OPTION	R	3h	Select source for SCLK_LF. 0h = 31.25 kHz clock derived from 48 MHz XOSC or HPOSC. The RTC tick speed AON_RTC:SUBSECINC is updated to 0x8637BD, corresponding to a 31.25 kHz clock (done in the SetupTrimDevice() driverlib boot function). The device must be blocked from entering Standby mode when using this clock source. 1h = External low frequency clock on DIO defined by EXT_LF_CLK.DIO. The RTC tick speed AON_RTC:SUBSECINC is updated to EXT_LF_CLK.RTC_INCREMENT (done in the SetupTrimDevice() driverlib boot function). External clock must always be running when the chip is in standby for VDDR recharge timing. 2h = 32.768 kHz low frequency XOSC 3h = Low frequency RCOSC (default)
21	VDDR_TRIM_SLEEP_TC	R	1h	0x1: VDDR_TRIM_SLEEP_DELTA is not temperature compensated 0x0: TI's Power Manager temperature compensates VDDR_TRIM_SLEEP_DELTA every time Standby mode is entered. When temperature compensation is performed, the delta is calculated this way: Delta = max (delta, min(8, floor(62-temp)/8)) Here, delta is given by VDDR_TRIM_SLEEP_DELTA, and temp is the current temperature in degrees C.

Table 11-4. MODE_CONF Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20	RTC_COMP	R	1h	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
19-18	XOSC_FREQ	R	3h	Selects which high frequency oscillator is used (required for radio usage). 0h = External 48 MHz TCXO. Refer to MODE_CONF_1.TCXO_MAX_START and MODE_CONF_1.TCXO_TYPE bit fields for additional configuration of TCXO. 1h = Internal high precision oscillator. 2h = 48M : 48 MHz XOSC_HF 3h = 24M : 24 MHz XOSC_HF. Not supported.
17	XOSC_CAP_MOD	R	1h	Enable modification (delta) to XOSC cap-array. Value specified in XOSC_CAPARRAY_DELTA. 0: Apply cap-array delta 1: Do not apply cap-array delta (default)
16	HF_COMP	R	1h	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
15-8	XOSC_CAPARRAY_DELTA	R	FFh	Signed 8-bit value, directly modifying trimmed XOSC cap-array step value. Enabled by XOSC_CAP_MOD.
7-0	VDDR_CAP	R	FFh	Unsigned 8-bit integer, representing the minimum decoupling capacitance (worst case) on VDDR, in units of 100nF. This should take into account capacitor tolerance and voltage dependent capacitance variation. This bit affects the recharge period calculation when going into powerdown or standby. NOTE! If using the following functions this field must be configured (used by TI RTOS): SysCtrlSetRechargeBeforePowerDown() SysCtrlAdjustRechargeAfterPowerDown()

11.2.3 MODE_CONF_1 Register (Offset = 8h) [Reset = FFFFFFFFh]

MODE_CONF_1 is shown in Table 11-5.

Return to the Summary Table.

Mode Configuration 1

Table 11-5. MODE_CONF_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	TCXO_TYPE	R	1h	Selects the TCXO type. 0: CMOS type. Internal common-mode bias will not be enabled. 1: Clipped-sine type. Internal common-mode bias will be enabled when TCXO is used. Bit field value is only valid if MODE_CONF.XOSC_FREQ=0.
30-24	TCXO_MAX_START	R	7Fh	Maximum TCXO startup time in units of 100us. Bit field value is only valid if MODE_CONF.XOSC_FREQ=0.
23-20	ALT_DCDC_VMIN	R	Fh	Minimum voltage for when DC/DC should be used if alternate DC/DC setting is enabled (SIZE_AND_DIS_FLAGS.DIS_ALT_DCDC_SETTING=0). Voltage = (28 + ALT_DCDC_VMIN) / 16. 0: 1.75V 1: 1.8125V ... 14: 2.625V 15: 2.6875V NOTE! The DriverLib function SysCtrl_DCDC_VoltageConditionalControl() must be called regularly to apply this field (handled automatically if using TI RTOS!).
19	ALT_DCDC_DITHER_EN	R	1h	Enable DC/DC dithering if alternate DC/DC setting is enabled (SIZE_AND_DIS_FLAGS.DIS_ALT_DCDC_SETTING=0). 0: Dither disable 1: Dither enable
18-16	ALT_DCDC_IPEAK	R	7h	Inductor peak current if alternate DC/DC setting is enabled (SIZE_AND_DIS_FLAGS.DIS_ALT_DCDC_SETTING=0). Assuming 10uH external inductor! 0: Min 46mA ... 4: Typical 70mA ... 7: Max 87mA
15-12	DELTA_IBIAS_INIT	R	Fh	Signed delta value for IBIAS_INIT. Delta value only applies if SIZE_AND_DIS_FLAGS.DIS_XOSC_OVR=0. See FCFG1:AMPCOMP_CTRL1.IBIAS_INIT
11-8	DELTA_IBIAS_OFFSET	R	Fh	Signed delta value for IBIAS_OFFSET. Delta value only applies if SIZE_AND_DIS_FLAGS.DIS_XOSC_OVR=0. See FCFG1:AMPCOMP_CTRL1.IBIAS_OFFSET
7-0	XOSC_MAX_START	R	FFh	Unsigned value of maximum XOSC startup time (worst case) in units of 100us. Value only applies if SIZE_AND_DIS_FLAGS.DIS_XOSC_OVR=0.

11.2.4 VOLT_LOAD_0 Register (Offset = Ch) [Reset = FFFFFFFFh]

VOLT_LOAD_0 is shown in Table 11-6.

Return to the Summary Table.

Voltage Load 0
Enabled by MODE_CONF.VDDR_EXT_LOAD.

Table 11-6. VOLT_LOAD_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	VDDR_EXT_TP45	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
23-16	VDDR_EXT_TP25	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
15-8	VDDR_EXT_TP5	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
7-0	VDDR_EXT_TM15	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.

11.2.5 VOLT_LOAD_1 Register (Offset = 10h) [Reset = FFFFFFFFh]

VOLT_LOAD_1 is shown in Table 11-7.

Return to the Summary Table.

Voltage Load 1
Enabled by MODE_CONF.VDDR_EXT_LOAD.

Table 11-7. VOLT_LOAD_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	VDDR_EXT_TP125	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
23-16	VDDR_EXT_TP105	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
15-8	VDDR_EXT_TP85	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.
7-0	VDDR_EXT_TP65	R	FFh	Reserved for future use. Software should not rely on the value of a reserved. Writing any other value than the reset/default value may result in undefined behavior.

11.2.6 EXT_LF_CLK Register (Offset = 14h) [Reset = FFFFFFFFh]

EXT_LF_CLK is shown in Table 11-8.

Return to the Summary Table.

Extern LF clock configuration

Table 11-8. EXT_LF_CLK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	DIO	R	FFh	Unsigned integer, selecting the DIO to supply external 32 kHz clock as SCLK_LF when MODE_CONF.SCLK_LF_OPTION is set to EXTERNAL.
23-0	RTC_INCREMENT	R	00FFFFFFh	Unsigned integer, defining the input frequency of the external clock and is written to AON_RTC:SUBSECINC.VALUEINC. Defined as follows: EXT_LF_CLK.RTC_INCREMENT = 2^{38}/ InputClockFrequency in Hertz (e.g.: RTC_INCREMENT=0x800000 for InputClockFrequency=32768 Hz)

11.2.7 IEEE_MAC_0 Register (Offset = 18h) [Reset = FFFFFFFFh]

IEEE_MAC_0 is shown in Table 11-9.

Return to the Summary Table.

IEEE MAC Address 0

Table 11-9. IEEE_MAC_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	FFFFFFFFh	Bits[31:0] of the 64-bits custom IEEE MAC address. If different from 0xFFFFFFFF then the value of this field is applied otherwise use value from FCFG:MAC_15_4_0.

11.2.8 IEEE_MAC_1 Register (Offset = 1Ch) [Reset = FFFFFFFFh]

IEEE_MAC_1 is shown in Table 11-10.

Return to the Summary Table.

IEEE MAC Address 1

Table 11-10. IEEE_MAC_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	FFFFFFFFh	Bits[63:32] of the 64-bits custom IEEE MAC address. If different from 0xFFFFFFFF then the value of this field is applied, otherwise use value from FCFG:MAC_15_4_1.

11.2.9 IEEE_BLE_0 Register (Offset = 20h) [Reset = FFFFFFFFh]

IEEE_BLE_0 is shown in Table 11-11.

Return to the Summary Table.

IEEE BLE Address 0

Table 11-11. IEEE_BLE_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	FFFFFFFFh	Bits[31:0] of the 64-bits custom IEEE BLE address. If different from 0xFFFFFFFF then the value of this field is applied otherwise use value from FCFG:MAC_BLE_0.

11.2.10 IEEE_BLE_1 Register (Offset = 24h) [Reset = FFFFFFFFh]

IEEE_BLE_1 is shown in Table 11-12.

Return to the Summary Table.

IEEE BLE Address 1

Table 11-12. IEEE_BLE_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	FFFFFFFFh	Bits[63:32] of the 64-bits custom IEEE BLE address. If different from 0xFFFFFFFF then the value of this field is applied, otherwise use value from FCFG:MAC_BLE_1.

11.2.11 BL_CONFIG Register (Offset = 28h) [Reset = C5FFFFFFh]

BL_CONFIG is shown in Table 11-13.

Return to the Summary Table.

Bootloader Configuration
Configures the functionality of the ROM boot loader.
If both the boot loader is enabled by the BOOTLOADER_ENABLE field and the boot loader backdoor is enabled by the BL_ENABLE field it is possible to force entry of the ROM boot loader even if a valid image is present in flash.

Table 11-13. BL_CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	BOOTLOADER_ENABLE	R	C5h	Bootloader enable. Boot loader can be accessed if IMAGE_VALID_CONF.IMAGE_VALID is an invalid vector table address or BL_ENABLE is enabled (and conditions for boot loader backdoor are met). 0xC5: Boot loader is enabled. Any other value: Boot loader is disabled.
23-17	RESERVED	R	0h	Reserved
16	BL_LEVEL	R	1h	Sets the active level of the selected DIO number BL_PIN_NUMBER if boot loader backdoor is enabled by the BL_ENABLE field. 0: Active low. 1: Active high.
15-8	BL_PIN_NUMBER	R	FFh	DIO number that is level checked if the boot loader backdoor is enabled by the BL_ENABLE field.
7-0	BL_ENABLE	R	FFh	Enables the boot loader backdoor. 0xC5: Boot loader backdoor is enabled. Any other value: Boot loader backdoor is disabled. NOTE! Boot loader must be enabled (see BOOTLOADER_ENABLE) if boot loader backdoor is enabled.

11.2.12 ERASE_CONF Register (Offset = 2Ch) [Reset = FFFFFFFFh]

ERASE_CONF is shown in Table 11-14.

Return to the Summary Table.

Erase Configuration

Table 11-14. ERASE_CONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	CHIP_ERASE_DIS_N	R	1h	Chip erase. This bit controls if a chip erase requested through the JTAG WUC TAP will be ignored in a following boot caused by a reset of the MCU VD. A successful chip erase operation will force the content of the flash main bank back to the state as it was when delivered by TI. 0: Disable. Any chip erase request detected during boot will be ignored. 1: Enable. Any chip erase request detected during boot will be performed by the boot FW.
7-1	RESERVED	R	0h	Reserved
0	BANK_ERASE_DIS_N	R	1h	Bank erase. This bit controls if the ROM serial boot loader will accept a received Bank Erase command (COMMAND_BANK_ERASE). A successful Bank Erase operation will erase all main bank sectors not protected by write protect configuration bits in CCFG. 0: Disable the boot loader bank erase function. 1: Enable the boot loader bank erase function.

11.2.13 ERASE_CONF_1 Register (Offset = 30h) [Reset = FFFFFFFFh]

ERASE_CONF_1 is shown in Table 11-15.

Return to the Summary Table.

Erase Configuration 1

Table 11-15. ERASE_CONF_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	WEPROT_CCFG_N	R	1h	WriteErase protect the CCFG sector Setting this bit = 0 will set FLASH:WEPROT_AUX_BY1.WEPROT_B0_CCFG_BY1 = 1 during boot and hence WriteErase protect the CCFG

11.2.14 CCFG_TI_OPTIONS Register (Offset = 34h) [Reset = FFC5C5C5h]

CCFG_TI_OPTIONS is shown in Table 11-16.

Return to the Summary Table.

TI Options

Table 11-16. CCFG_TI_OPTIONS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	C_FA_DIS	R	C5h	Option to disable failure analysis without customer password. If C_FA_DIS != 0xC5, CKEY (CKEY0.KEY, CKEY1.KEY, CKEY2.KEY, CKEY2.KEY) must be provided to TI for failure analysis to be possible. 0xC5: Failure analysis without customer password is enabled All other values: Failure analysis without customer password is disabled
15-8	IDAU_CFG_ENABLE	R	C5h	IDAU configuration. 0xC5: Disable IDAU configuration controlled by TRUSTZONE_FLASH_CFG and TRUSTZONE_SRAM_CFG. All other values: Enable IDAU configuration controlled by TRUSTZONE_FLASH_CFG and TRUSTZONE_SRAM_CFG.
7-0	TI_FA_ENABLE	R	C5h	TI Failure Analysis. 0xC5: Enable the functionality of unlocking the TI FA (TI Failure Analysis) option with the unlock code. All other values: Disable the functionality of unlocking the TI FA option with the unlock code.

11.2.15 CCFG_TAP_DAP_0 Register (Offset = 38h) [Reset = FFC5C5C5h]

CCFG_TAP_DAP_0 is shown in Table 11-17.

Return to the Summary Table.

Test Access Points Enable 0

Table 11-17. CCFG_TAP_DAP_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	CPU_DAP_ENABLE	R	C5h	Enable CPU DAP. 0xC5: Main CPU DAP access is enabled during power-up/system-reset by ROM boot FW. Any other value: Main CPU DAP access will remain disabled out of power-up/system-reset.
15-8	PWRPROF_TAP_ENABLE	R	C5h	Enable PWRPROF TAP. 0xC5: PWRPROF TAP access is enabled during power-up/system-reset by ROM boot FW if enabled by corresponding configuration value in FCFG1 defined by TI. Any other value: PWRPROF TAP access will remain disabled out of power-up/system-reset.
7-0	TEST_TAP_ENABLE	R	C5h	Enable Test TAP. 0xC5: TEST TAP access is enabled during power-up/system-reset by ROM boot FW if enabled by corresponding configuration value in FCFG1 defined by TI. Any other value: TEST TAP access will remain disabled out of power-up/system-reset.

11.2.16 CCFG_TAP_DAP_1 Register (Offset = 3Ch) [Reset = FFC5C5C5h]

CCFG_TAP_DAP_1 is shown in Table 11-18.

Return to the Summary Table.

Test Access Points Enable 1

Table 11-18. CCFG_TAP_DAP_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	PBIST2_TAP_ENABLE	R	C5h	Enable PBIST2 TAP. 0xC5: PBIST2 TAP access is enabled during power-up/system-reset by ROM boot FW if enabled by corresponding configuration value in FCFG1 defined by TI. Any other value: PBIST2 TAP access will remain disabled out of power-up/system-reset.
15-8	PBIST1_TAP_ENABLE	R	C5h	Enable PBIST1 TAP. 0xC5: PBIST1 TAP access is enabled during power-up/system-reset by ROM boot FW if enabled by corresponding configuration value in FCFG1 defined by TI. Any other value: PBIST1 TAP access will remain disabled out of power-up/system-reset.
7-0	AON_TAP_ENABLE	R	C5h	Enable AON TAP 0xC5: AON TAP access is enabled during power-up/system-reset by ROM boot FW if enabled by corresponding configuration value in FCFG1 defined by TI. Any other value: AON TAP access will remain disabled out of power-up/system-reset.

11.2.17 IMAGE_VALID_CONF Register (Offset = 40h) [Reset = FFFFFFFFh]

IMAGE_VALID_CONF is shown in Table 11-19.

Return to the Summary Table.

Image Valid

Table 11-19. IMAGE_VALID_CONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IMAGE_VALID	R	FFFFFFFFh	This field must have the address value of the start of the flash vector table in order to enable the boot FW in ROM to transfer control to a flash image. Any illegal vector table start address value will force the boot FW in ROM to transfer control to the serial boot loader in ROM.

11.2.18 CCFG_WEPROT_31_0_BY2K Register (Offset = 44h) [Reset = FFFFFFFFh]

CCFG_WEPROT_31_0_BY2K is shown in Table 11-20.

Return to the Summary Table.

Protect Sectors 0-31
Each bit write protects one 2KB flash sector from being both programmed and erased. Bit must be set to 0 in order to enable sector WriteErase protect.

Table 11-20. CCFG_WEPROT_31_0_BY2K Register Field Descriptions

Bit	Field	Type	Reset	Description
31	WEPROT_SEC_31_N	R	1h	0: Sector protected
30	WEPROT_SEC_30_N	R	1h	0: Sector protected
29	WEPROT_SEC_29_N	R	1h	0: Sector protected
28	WEPROT_SEC_28_N	R	1h	0: Sector protected
27	WEPROT_SEC_27_N	R	1h	0: Sector protected
26	WEPROT_SEC_26_N	R	1h	0: Sector protected
25	WEPROT_SEC_25_N	R	1h	0: Sector protected
24	WEPROT_SEC_24_N	R	1h	0: Sector protected
23	WEPROT_SEC_23_N	R	1h	0: Sector protected
22	WEPROT_SEC_22_N	R	1h	0: Sector protected
21	WEPROT_SEC_21_N	R	1h	0: Sector protected
20	WEPROT_SEC_20_N	R	1h	0: Sector protected
19	WEPROT_SEC_19_N	R	1h	0: Sector protected
18	WEPROT_SEC_18_N	R	1h	0: Sector protected
17	WEPROT_SEC_17_N	R	1h	0: Sector protected
16	WEPROT_SEC_16_N	R	1h	0: Sector protected
15	WEPROT_SEC_15_N	R	1h	0: Sector protected
14	WEPROT_SEC_14_N	R	1h	0: Sector protected
13	WEPROT_SEC_13_N	R	1h	0: Sector protected
12	WEPROT_SEC_12_N	R	1h	0: Sector protected
11	WEPROT_SEC_11_N	R	1h	0: Sector protected
10	WEPROT_SEC_10_N	R	1h	0: Sector protected
9	WEPROT_SEC_9_N	R	1h	0: Sector protected
8	WEPROT_SEC_8_N	R	1h	0: Sector protected
7	WEPROT_SEC_7_N	R	1h	0: Sector protected
6	WEPROT_SEC_6_N	R	1h	0: Sector protected
5	WEPROT_SEC_5_N	R	1h	0: Sector protected
4	WEPROT_SEC_4_N	R	1h	0: Sector protected
3	WEPROT_SEC_3_N	R	1h	0: Sector protected
2	WEPROT_SEC_2_N	R	1h	0: Sector protected
1	WEPROT_SEC_1_N	R	1h	0: Sector protected
0	WEPROT_SEC_0_N	R	1h	0: Sector protected

11.2.19 CCFG_WEPROT_SPARE_1 Register (Offset = 48h) [Reset = FFFFFFFFh]

CCFG_WEPROT_SPARE_1 is shown in Table 11-21.

Return to the Summary Table.

Spare register for WriteErase configuration

Table 11-21. CCFG_WEPROT_SPARE_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	FFFFFFFFh	Reserved

11.2.20 CCFG_WEPROT_SPARE_2 Register (Offset = 4Ch) [Reset = FFFFFFFFh]

CCFG_WEPROT_SPARE_2 is shown in Table 11-22.

Return to the Summary Table.

Spare register for WriteErase configuration

Table 11-22. CCFG_WEPROT_SPARE_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	FFFFFFFFh	Reserved

11.2.21 CCFG_WEPROT_SPARE_3 Register (Offset = 50h) [Reset = FFFFFFFFh]

CCFG_WEPROT_SPARE_3 is shown in Table 11-23.

Return to the Summary Table.

Spare register for WriteErase configuration

Table 11-23. CCFG_WEPROT_SPARE_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	FFFFFFFFh	Reserved

11.2.22 TRUSTZONE_FLASH_CFG Register (Offset = 54h) [Reset = FFFFFFFFh]

TRUSTZONE_FLASH_CFG is shown in Table 11-24.

Return to the Summary Table.

Trustzone configuration register for flash

Table 11-24. TRUSTZONE_FLASH_CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-10	NSADDR_BOUNDARY	R	7Fh	Value will be written to PRCM:NVMNSADDR.BOUNDARY by ROM boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5.
9-0	NSCADDR_BOUNDARY	R	3FFh	Value will be written to PRCM:NVMNSCADDR.BOUNDARY by ROM boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5.

11.2.23 TRUSTZONE_SRAM_CFG Register (Offset = 58h) [Reset = FFFFFFFFh]

TRUSTZONE_SRAM_CFG is shown in Table 11-25.

Return to the Summary Table.

Trustzone configuration register for MCU SRAM

Table 11-25. TRUSTZONE_SRAM_CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17-9	NSADDR_BOUNDARY	R	1FFh	Value will be written to PRCM:SRAMNSADDR.BOUNDARY by ROM boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5.
8-0	NSCADDR_BOUNDARY	R	1FFh	Value will be written to PRCM:SRAMNSCADDR.BOUNDARY by ROM boot FW only if CCFG_TI_OPTIONS.IDAU_CFG_ENABLE != 0xC5.

11.2.24 SRAM_CFG Register (Offset = 5Ch) [Reset = FFFFFFFFh]

SRAM_CFG is shown in Table 11-26.

Return to the Summary Table.

Configuration register for MCU SRAM

Table 11-26. SRAM_CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	MEM_SEL	R	00FFFFFFh	Value will be written to SRAM_MMR:MEM_CTL.MEM_SEL by ROM boot FW
7-1	RESERVED	R	0h	Reserved
0	PARITY_DIS	R	1h	Value will be inverted and then written to PRCM:MCUSRAMCFG.PARITY_EN by ROM boot FW

11.2.25 CPU_LOCK_CFG Register (Offset = 64h) [Reset = FFFFFFFFh]

CPU_LOCK_CFG is shown in Table 11-27.

Return to the Summary Table.

Configuration register for MCU CPU lock options

Table 11-27. CPU_LOCK_CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	LOCKNSVTOR_N	R	1h	Value will be inverted and written to PRCM:CPULOCK.LOCKNSVTOR by ROM boot FW
3	LOCKSVTAIRCR_N	R	1h	Value will be inverted and written to PRCM:CPULOCK.LOCKSVTAIRCR by ROM boot FW
2	LOCKSAU_N	R	1h	Value will be inverted and written to PRCM:CPULOCK.LOCKSAU by ROM boot FW
1	LOCKNSMPU_N	R	1h	Value will be inverted and written to PRCM:CPULOCK.LOCKNSMPU by ROM boot FW
0	LOCKSMPU_N	R	1h	Value will be inverted and written to PRCM:CPULOCK.LOCKSMPU by ROM boot FW

11.2.26 DEB_AUTH_CFG Register (Offset = 68h) [Reset = FFFFFFFFh]

DEB_AUTH_CFG is shown in Table 11-28.

Return to the Summary Table.

Configuration register for debug authentication

Table 11-28. DEB_AUTH_CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	INTSPNIDEN	R	1h	Value will be written to CPU_DCB:DAUTHCTRL.INTSPNIDEN by ROM boot FW
2	SPNIDENSEL	R	1h	Value will be written to CPU_DCB:DAUTHCTRL.SPNIDENSEL by ROM boot FW
1	INTSPIDEN	R	1h	Value will be written to CPU_DCB:DAUTHCTRL.INTSPIDEN by ROM boot FW
0	SPIDENSEL	R	1h	Value will be written to CPU_DCB:DAUTHCTRL.SPIDENSEL by ROM boot FW

11.2.27 CKEY0 Register (Offset = 6Ch) [Reset = 0FFFFFFFh]

CKEY0 is shown in Table 11-29.

Return to the Summary Table.

Customer key

Table 11-29. CKEY0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R	0FFFFFFFh	Bit[31:0] of customer key used for XOR of TI unlock code when CCFG_TI_OPTIONS.C_FA_DIS != 0xC5.

11.2.28 CKEY1 Register (Offset = 70h) [Reset = 0FFFFFFFh]

CKEY1 is shown in Table 11-30.

Return to the Summary Table.

Customer key

Table 11-30. CKEY1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R	0FFFFFFFh	Bit[63:32] of customer key used for XOR of TI unlock code when CCFG_TI_OPTIONS.C_FA_DIS != 0xC5.

11.2.29 CKEY2 Register (Offset = 74h) [Reset = 0FFFFFFFh]

CKEY2 is shown in Table 11-31.

Return to the Summary Table.

Customer key

Table 11-31. CKEY2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R	0FFFFFFFh	Bit[95:64] of customer key used for XOR of TI unlock code when CCFG_TI_OPTIONS.C_FA_DIS != 0xC5.

11.2.30 CKEY3 Register (Offset = 78h) [Reset = 0FFFFFFFh]

CKEY3 is shown in Table 11-32.

Return to the Summary Table.

Customer key

Table 11-32. CKEY3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KEY	R	0FFFFFFFh	Bit[127:96] of customer key used for XOR of TI unlock code when CCFG_TI_OPTIONS.C_FA_DIS != 0xC5.

11.3 Factory Configuration (FCFG)

The FCFG are programmed for each device by the TI production test. The FCFG contains device-specific trim values and configuration. Most of the trim values are used by TI boot code, RF core, ROM code, or are automatically provided by TI software.

Some of the more useful fields in FCFG are:

- MAC_15_4_n fields, which give the preprogrammed IEEE address of the chipset
- MAC_BLE_n fields, which give the Bluetooth® Low Energy address of the chipset

11.4 FCFG1 Registers

Table 11-33 lists the memory-mapped registers for the FCFG1 registers. All register offset addresses not listed in Table 11-33 should be considered as reserved locations and the register contents should not be modified.

Table 11-33. FCFG1 Registers

Offset	Acronym	Register Name	Section
A0h	MISC_CONF_1	Misc configurations	Section 11.4.1
A4h	MISC_CONF_2	Internal	Section 11.4.2
B0h	HPOSC_MEAS_5	Internal	Section 11.4.3
B4h	HPOSC_MEAS_4	Internal	Section 11.4.4
B8h	HPOSC_MEAS_3	Internal	Section 11.4.5
BCh	HPOSC_MEAS_2	Internal	Section 11.4.6
C0h	HPOSC_MEAS_1	Internal	Section 11.4.7
C4h	CONFIG_CC26_FE	Internal	Section 11.4.8
C8h	CONFIG_CC13_FE	Internal	Section 11.4.9
CCh	CONFIG_RF_COMMON	Internal	Section 11.4.10
D0h	CONFIG_SYNTH_DIV2_CC26_2G4	Internal	Section 11.4.11
D4h	CONFIG_SYNTH_DIV2_CC13_2G4	Internal	Section 11.4.12
D8h	CONFIG_SYNTH_DIV2_CC26_1G	Internal	Section 11.4.13
DCh	CONFIG_SYNTH_DIV2_CC13_1G	Internal	Section 11.4.14
E0h	CONFIG_SYNTH_DIV4_CC26	Internal	Section 11.4.15
E4h	CONFIG_SYNTH_DIV4_CC13	Internal	Section 11.4.16
E8h	CONFIG_SYNTH_DIV5	Internal	Section 11.4.17
ECh	CONFIG_SYNTH_DIV6_CC26	Internal	Section 11.4.18
F0h	CONFIG_SYNTH_DIV6_CC13	Internal	Section 11.4.19
F4h	CONFIG_SYNTH_DIV10	Internal	Section 11.4.20
F8h	CONFIG_SYNTH_DIV12_CC26	Internal	Section 11.4.21
FCh	CONFIG_SYNTH_DIV12_CC13	Internal	Section 11.4.22
100h	CONFIG_SYNTH_DIV15	Internal	Section 11.4.23
104h	CONFIG_SYNTH_DIV30	Internal	Section 11.4.24
144h	IOCONF	IO Configuration	Section 11.4.25
294h	USER_ID	User Identification.	Section 11.4.26
2B0h	FLASH_OTP_DATA3	Internal	Section 11.4.27
2B4h	ANA2_TRIM	Internal	Section 11.4.28
2B8h	LDO_TRIM	Internal	Section 11.4.29
2E8h	MAC_BLE_0	MAC BLE Address 0	Section 11.4.30
2ECh	MAC_BLE_1	MAC BLE Address 1	Section 11.4.31
2F0h	MAC_15_4_0	MAC IEEE 802.15.4 Address 0	Section 11.4.32
2F4h	MAC_15_4_1	MAC IEEE 802.15.4 Address 1	Section 11.4.33
30Ch	MISC_TRIM	Miscellaneous Trim Parameters	Section 11.4.34
310h	RCOSC_HF_TEMPCOMP	Internal	Section 11.4.35
318h	ICEPICK_DEVICE_ID	IcePick Device Identification	Section 11.4.36
31Ch	FCFG1_REVISION	Factory Configuration (FCFG1) Revision	Section 11.4.37
320h	MISC_OTP_DATA	Misc OTP Data	Section 11.4.38
34Ch	CONFIG_IF_ADC	Internal	Section 11.4.39
350h	CONFIG_OSC_TOP	Internal	Section 11.4.40
35Ch	SOC_ADC_ABS_GAIN	AUX_ADC Gain in Absolute Reference Mode	Section 11.4.41

Table 11-33. FCFG1 Registers (continued)

Offset	Acronym	Register Name	Section
360h	SOC_ADC_REL_GAIN	AUX_ADC Gain in Relative Reference Mode	Section 11.4.42
368h	SOC_ADC_OFFSET_INT	AUX_ADC Temperature Offsets in Absolute Reference Mode	Section 11.4.43
36Ch	SOC_ADC_REF_TRIM_AND_OFFSET_EXT	Internal	Section 11.4.44
370h	AMPCOMP_TH1	Internal	Section 11.4.45
374h	AMPCOMP_TH2	Internal	Section 11.4.46
378h	AMPCOMP_CTRL1	Internal	Section 11.4.47
37Ch	ANABYPASS_VALUE2	Internal	Section 11.4.48
388h	VOLT_TRIM	Internal	Section 11.4.49
38Ch	OSC_CONF	OSC Configuration	Section 11.4.50
390h	FREQ_OFFSET	Internal	Section 11.4.51
398h	MISC_OTP_DATA_1	Internal	Section 11.4.52
3D0h	SHDW_DIE_ID_0	Shadow of EFUSE:DIE_ID_0 register	Section 11.4.53
3D4h	SHDW_DIE_ID_1	Shadow of EFUSE:DIE_ID_1 register	Section 11.4.54
3D8h	SHDW_DIE_ID_2	Shadow of EFUSE:DIE_ID_2 register	Section 11.4.55
3DCh	SHDW_DIE_ID_3	Shadow of EFUSE:DIE_ID_3 register	Section 11.4.56
3F8h	SHDW_SCAN_MCU3_SEC	Internal	Section 11.4.57
3FCh	SHDW_SCAN_DATA1_CRC	Internal	Section 11.4.58
404h	SHDW_ANA_TRIM	Internal	Section 11.4.59
408h	OSC_CONF1	Oscillator configuration	Section 11.4.60
40Ch	DAC_BIAS_CNF	Internal	Section 11.4.61
418h	TFW_PROBE	Internal	Section 11.4.62
41Ch	TFW_FT	Internal	Section 11.4.63
420h	DAC_CAL0	Internal	Section 11.4.64
424h	DAC_CAL1	Internal	Section 11.4.65
428h	DAC_CAL2	Internal	Section 11.4.66
42Ch	DAC_CAL3	Internal	Section 11.4.67

Complex bit access types are encoded to fit into small table cells. Table 11-34 shows the codes that are used for access types in this section.

Table 11-34. FCFG1 Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Reset or Default Value		
-n		Value after reset or the default value

11.4.1 MISC_CONF_1 Register (Offset = A0h) [Reset = FFFFFF00h]

MISC_CONF_1 is shown in Table 11-35.

Return to the Summary Table.

Misc configurations

Table 11-35. MISC_CONF_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	DEVICE_MINOR_REV	R	X	HW minor revision number (a value of 0xFF shall be treated equally to 0x00). Any test of this field by SW should be implemented as a 'greater or equal' comparison as signed integer. Value may change without warning. Default value holds log information from production test.

11.4.2 MISC_CONF_2 Register (Offset = A4h) [Reset = FFFFFF00h]

MISC_CONF_2 is shown in Table 11-36.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-36. MISC_CONF_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	HPOSC_COMP_P3	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.3 HPOSC_MEAS_5 Register (Offset = B0h) [Reset = 00000000h]

HPOSC_MEAS_5 is shown in Table 11-37.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-37. HPOSC_MEAS_5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_D5	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
15-8	HPOSC_T5	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
7-0	HPOSC_DT5	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.

11.4.4 HPOSC_MEAS_4 Register (Offset = B4h) [Reset = 00000000h]

HPOSC_MEAS_4 is shown in Table 11-38.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-38. HPOSC_MEAS_4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_D4	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
15-8	HPOSC_T4	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
7-0	HPOSC_DT4	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.

11.4.5 HPOSC_MEAS_3 Register (Offset = B8h) [Reset = 00000000h]

HPOSC_MEAS_3 is shown in Table 11-39.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-39. HPOSC_MEAS_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_D3	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
15-8	HPOSC_T3	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
7-0	HPOSC_DT3	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.

11.4.6 HPOSC_MEAS_2 Register (Offset = BCh) [Reset = 00000000h]

HPOSC_MEAS_2 is shown in Table 11-40.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-40. HPOSC_MEAS_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_D2	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
15-8	HPOSC_T2	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
7-0	HPOSC_DT2	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.

11.4.7 HPOSC_MEAS_1 Register (Offset = C0h) [Reset = 00000000h]

HPOSC_MEAS_1 is shown in Table 11-41.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-41. HPOSC_MEAS_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_D1	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
15-8	HPOSC_T1	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.
7-0	HPOSC_DT1	R	X	Internal. Only to be used through TI provided API. Default value holds log information from production test.

11.4.8 CONFIG_CC26_FE Register (Offset = C4h) [Reset = 70000F00h]

CONFIG_CC26_FE is shown in Table 11-42.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-42. CONFIG_CC26_FE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	IFAMP_IB	R	7h	Internal. Only to be used through TI provided API.
27-24	LNA_IB	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
23-19	IFAMP_TRIM	R	0h	Internal. Only to be used through TI provided API.
18-14	CTL_PA0_TRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
13	PATRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
12	RSSITRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-8	RESERVED	R	0h	Reserved
7-0	RSSI_OFFSET	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.9 CONFIG_CC13_FE Register (Offset = C8h) [Reset = 70000F00h]

CONFIG_CC13_FE is shown in Table 11-43.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-43. CONFIG_CC13_FE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	IFAMP_IB	R	7h	Internal. Only to be used through TI provided API.
27-24	LNA_IB	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
23-19	IFAMP_TRIM	R	0h	Internal. Only to be used through TI provided API.
18-14	CTL_PA0_TRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
13	PATRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
12	RSSITRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-8	RESERVED	R	0h	Reserved
7-0	RSSI_OFFSET	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.10 CONFIG_RF_COMMON Register (Offset = CCh) [Reset = 81C0014Dh]

CONFIG_RF_COMMON is shown in Table 11-44.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-44. CONFIG_RF_COMMON Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DISABLE_CORNER_CAP	R	1h	Internal. Only to be used through TI provided API.
30-25	SLDO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
24-22	RESERVED	R	0h	Reserved
21	PA20DBMTRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API.
20-16	CTL_PA_20DBM_TRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-9	RFLDO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API.
8-6	QUANTCTLTHRES	R	5h	Internal. Only to be used through TI provided API.
5-0	DACTRIM	R	Dh	Internal. Only to be used through TI provided API.

11.4.11 CONFIG_SYNTH_DIV2_CC26_2G4 Register (Offset = D0h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV2_CC26_2G4 is shown in Table 11-45.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-45. CONFIG_SYNTH_DIV2_CC26_2G4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.12 CONFIG_SYNTH_DIV2_CC13_2G4 Register (Offset = D4h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV2_CC13_2G4 is shown in Table 11-46.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-46. CONFIG_SYNTH_DIV2_CC13_2G4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.13 CONFIG_SYNTH_DIV2_CC26_1G Register (Offset = D8h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV2_CC26_1G is shown in Table 11-47.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-47. CONFIG_SYNTH_DIV2_CC26_1G Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.14 CONFIG_SYNTH_DIV2_CC13_1G Register (Offset = DCh) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV2_CC13_1G is shown in Table 11-48.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-48. CONFIG_SYNTH_DIV2_CC13_1G Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.15 CONFIG_SYNTH_DIV4_CC26 Register (Offset = E0h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV4_CC26 is shown in Table 11-49.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-49. CONFIG_SYNTH_DIV4_CC26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.16 CONFIG_SYNTH_DIV4_CC13 Register (Offset = E4h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV4_CC13 is shown in Table 11-50.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-50. CONFIG_SYNTH_DIV4_CC13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.17 CONFIG_SYNTH_DIV5 Register (Offset = E8h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV5 is shown in Table 11-51.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-51. CONFIG_SYNTH_DIV5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.18 CONFIG_SYNTH_DIV6_CC26 Register (Offset = ECh) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV6_CC26 is shown in Table 11-52.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-52. CONFIG_SYNTH_DIV6_CC26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.19 CONFIG_SYNTH_DIV6_CC13 Register (Offset = F0h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV6_CC13 is shown in Table 11-53.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-53. CONFIG_SYNTH_DIV6_CC13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.20 CONFIG_SYNTH_DIV10 Register (Offset = F4h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV10 is shown in Table 11-54.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-54. CONFIG_SYNTH_DIV10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.21 CONFIG_SYNTH_DIV12_CC26 Register (Offset = F8h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV12_CC26 is shown in Table 11-55.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-55. CONFIG_SYNTH_DIV12_CC26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.22 CONFIG_SYNTH_DIV12_CC13 Register (Offset = FCh) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV12_CC13 is shown in Table 11-56.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-56. CONFIG_SYNTH_DIV12_CC13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.23 CONFIG_SYNTH_DIV15 Register (Offset = 100h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV15 is shown in Table 11-57.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-57. CONFIG_SYNTH_DIV15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.24 CONFIG_SYNTH_DIV30 Register (Offset = 104h) [Reset = 0000001Fh]

CONFIG_SYNTH_DIV30 is shown in Table 11-58.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-58. CONFIG_SYNTH_DIV30 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	MIN_ALLOWED_RTRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-12	RFC_MDM_DEMIQMC0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-6	LDOVCO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
5	RFC_MDM_DEMIQMC0_TRIMCOMPLETE_N	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	RESERVED	R	0h	Reserved

11.4.25 IOCONF Register (Offset = 144h) [Reset = FFFFFF00h]

IOCONF is shown in Table 11-59.

Return to the Summary Table.

IO Configuration

Table 11-59. IOCONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6-0	GPIO_CNT	R	X	Number of available DIOs. Default value differs depending on partnumber.

11.4.26 USER_ID Register (Offset = 294h) [Reset = 10000000h]

USER_ID is shown in Table 11-60.

Return to the Summary Table.

User Identification.
Reading this register and the FCFG1:ICEPICK_DEVICE_ID register is the only supported way of identifying a device.
The value of this register will be written to AON_PMCTL:JTAGUSERCODE by boot FW while in safezone.

Table 11-60. USER_ID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	PG_REV	R	1h	Field used to distinguish revisions of the device
27-26	VER	R	X	Version number. 0x0: Bits [25:12] of this register has the stated meaning. Any other setting indicate a different encoding of these bits. Default value differs depending on partnumber.
25	PA	R	X	0: Does not support 20dBm PA 1: Supports 20dBM PA Default value differs depending on partnumber.
24	RESERVED	R	0h	Reserved
23	CC13	R	X	0: CC26x4x10 device type 1: CC13x4x10 device type Default value differs depending on partnumber.
22-19	SEQUENCE	R	X	Sequence. Used to differentiate between marketing/orderable product where other fields of this register are the same (temp range, flash size, voltage range etc) Default value differs depending on partnumber.
18-16	PKG	R	X	Package type. 0x2: 7x7mm QFN (RGZ) package 0x7: 8x8mm QFN (RSK) package Other values are reserved for future use. Packages available for a specific device are shown in the device datasheet. Default value differs depending on partnumber.
15-12	PROTOCOL	R	X	Protocols supported. 0x1: BLE 0x2: RF4CE 0x4: Zigbee/6lowpan 0x8: Proprietary More than one protocol can be supported on same device - values above are then combined. Default value differs depending on partnumber.
11-0	RESERVED	R	0h	Reserved

11.4.27 FLASH_OTP_DATA3 Register (Offset = 2B0h) [Reset = FFFFFFF8h]

FLASH_OTP_DATA3 is shown in Table 11-61.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-61. FLASH_OTP_DATA3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	FLASH_SIZE	R	X	Internal. Only to be used through TI provided API. Default value differs depending on partnumber.

11.4.28 ANA2_TRIM Register (Offset = 2B4h) [Reset = 8240087Fh]

ANA2_TRIM is shown in Table 11-62.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-62. ANA2_TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RCOSCHFCTRIMFRACT_EN	R	1h	Internal. Only to be used through TI provided API.
30-26	RCOSCHFCTRIMFRACT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
25	RESERVED	R	0h	Reserved
24-23	SET_RCOSC_HF_FINE_RESISTOR	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
22	ATESTLF_UDIGLDO_IBIAS_TRIM	R	1h	Internal. Only to be used through TI provided API.
21-15	NANOAMP_RES_TRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
14-12	DCDC_DRV_DS	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11	DITHER_EN	R	1h	Internal. Only to be used through TI provided API.
10-8	DCDC_IPEAK	R	0h	Internal. Only to be used through TI provided API.
7-6	DEAD_TIME_TRIM	R	1h	Internal. Only to be used through TI provided API.
5-3	DCDC_LOW_EN_SEL	R	7h	Internal. Only to be used through TI provided API.
2-0	DCDC_HIGH_EN_SEL	R	7h	Internal. Only to be used through TI provided API.

11.4.29 LDO_TRIM Register (Offset = 2B8h) [Reset = E0F8E0FBh]

LDO_TRIM is shown in Table 11-63.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-63. LDO_TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	Reserved
28-24	VDDR_TRIM_SLEEP	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
23-19	RESERVED	R	0h	Reserved
18-16	GLDO_CURSRC	R	0h	Internal. Only to be used through TI provided API.
15-13	RESERVED	R	0h	Reserved
12-11	ITRIM_DIGLDO_LOAD	R	0h	Internal. Only to be used through TI provided API.
10-8	ITRIM_UDIGLDO	R	0h	Internal. Only to be used through TI provided API.
7-3	RESERVED	R	0h	Reserved
2-0	VTRIM_DELTA	R	3h	Internal. Only to be used through TI provided API.

11.4.30 MAC_BLE_0 Register (Offset = 2E8h) [Reset = 00000000h]

MAC_BLE_0 is shown in Table 11-64.

Return to the Summary Table.

MAC BLE Address 0

Table 11-64. MAC_BLE_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR_0_31	R	X	The first 32-bits of the 64-bit MAC BLE address Default value holds trim value from production test.

11.4.31 MAC_BLE_1 Register (Offset = 2ECh) [Reset = 00000000h]

MAC_BLE_1 is shown in Table 11-65.

Return to the Summary Table.

MAC BLE Address 1

Table 11-65. MAC_BLE_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR_32_63	R	X	The last 32-bits of the 64-bit MAC BLE address Default value holds trim value from production test.

11.4.32 MAC_15_4_0 Register (Offset = 2F0h) [Reset = 00000000h]

MAC_15_4_0 is shown in Table 11-66.

Return to the Summary Table.

MAC IEEE 802.15.4 Address 0

Table 11-66. MAC_15_4_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR_0_31	R	X	The first 32-bits of the 64-bit MAC 15.4 address Default value holds trim value from production test.

11.4.33 MAC_15_4_1 Register (Offset = 2F4h) [Reset = 00000000h]

MAC_15_4_1 is shown in Table 11-67.

Return to the Summary Table.

MAC IEEE 802.15.4 Address 1

Table 11-67. MAC_15_4_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR_32_63	R	X	The last 32-bits of the 64-bit MAC 15.4 address Default value holds trim value from production test.

11.4.34 MISC_TRIM Register (Offset = 30Ch) [Reset = FFFE003Bh]

MISC_TRIM is shown in Table 11-68.

Return to the Summary Table.

Miscellaneous Trim Parameters

Table 11-68. MISC_TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-12	TRIM_RECHARGE_ COMP_OFFSET	R	X	Internal. Only to be used through TI provided API.
11-8	TRIM_RECHARGE_ COMP_REFLEVEL	R	X	Internal. Only to be used through TI provided API.
7-0	TEMPVSLOPE	R	3Bh	Signed byte value representing the TEMP slope with battery voltage, in degrees C / V, with four fractional bits.

11.4.35 RCOSC_HF_TEMPCOMP Register (Offset = 310h) [Reset = 00000003h]

RCOSC_HF_TEMPCOMP is shown in Table 11-69.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-69. RCOSC_HF_TEMPCOMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	FINE_RESISTOR	R	0h	Internal. Only to be used through TI provided API.
23-16	CTRIM	R	0h	Internal. Only to be used through TI provided API.
15-8	CTRIMFRACT_QUAD	R	0h	Internal. Only to be used through TI provided API.
7-0	CTRIMFRACT_SLOPE	R	3h	Internal. Only to be used through TI provided API.

11.4.36 ICEPICK_DEVICE_ID Register (Offset = 318h) [Reset = 1BB7802Fh]

ICEPICK_DEVICE_ID is shown in Table 11-70.

Return to the Summary Table.

IcePick Device Identification
Reading this register and the FCFG1:USER_ID register is the only supported way of identifying a device.

Table 11-70. ICEPICK_DEVICE_ID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	PG_REV	R	1h	Field used to distinguish revisions of the device.
27-12	WAFER_ID	R	BB78h	Field used to identify silicon die.
11-0	MANUFACTURER_ID	R	2Fh	Manufacturer code. 0x02F: Texas Instruments

11.4.37 FCFG1_REVISION Register (Offset = 31Ch) [Reset = 0000002Ah]

FCFG1_REVISION is shown in Table 11-71.

Return to the Summary Table.

Factory Configuration (FCFG1) Revision

Table 11-71. FCFG1_REVISION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R	2Ah	The revision number of the FCFG1 layout. This value will be read by application SW in order to determine which FCFG1 parameters that have valid values. This revision number must be incremented by 1 before any devices are to be produced if the FCFG1 layout has changed since the previous production of devices. Value migth change without warning.

11.4.38 MISC_OTP_DATA Register (Offset = 320h) [Reset = 0000CFFFh]

MISC_OTP_DATA is shown in Table 11-72.

Return to the Summary Table.

Misc OTP Data

Table 11-72. MISC_OTP_DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RCOSC_HF_ITUNE	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
27-20	RCOSC_HF_CRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
19-15	PER_M	R	1h	Internal. Only to be used through TI provided API.
14-12	PER_E	R	4h	Internal. Only to be used through TI provided API.
11-0	RESERVED	R	0h	Reserved

11.4.39 CONFIG_IF_ADC Register (Offset = 34Ch) [Reset = 3460F400h]

CONFIG_IF_ADC is shown in Table 11-73.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-73. CONFIG_IF_ADC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	FF2ADJ	R	3h	Internal. Only to be used through TI provided API.
27-24	FF3ADJ	R	4h	Internal. Only to be used through TI provided API.
23-20	INT3ADJ	R	6h	Internal. Only to be used through TI provided API.
19-16	FF1ADJ	R	0h	Internal. Only to be used through TI provided API.
15-14	AAFCAP	R	3h	Internal. Only to be used through TI provided API.
13-10	INT2ADJ	R	Dh	Internal. Only to be used through TI provided API.
9-5	IFDIGLDO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
4-0	IFANALDO_TRIM_OUTPUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.40 CONFIG_OSC_TOP Register (Offset = 350h) [Reset = DC07FC00h]

CONFIG_OSC_TOP is shown in Table 11-74.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-74. CONFIG_OSC_TOP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-26	XOSC_HF_ROW_Q12	R	7h	Internal. Only to be used through TI provided API.
25-10	XOSC_HF_COLUMN_Q12	R	1FFh	Internal. Only to be used through TI provided API.
9-2	RCOSCLF_CTUNE_TRIM	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
1-0	RCOSCLF_RTUNE_TRIM	R	0h	Internal. Only to be used through TI provided API.

11.4.41 SOC_ADC_ABS_GAIN Register (Offset = 35Ch) [Reset = 00000000h]

SOC_ADC_ABS_GAIN is shown in Table 11-75.

Return to the Summary Table.

AUX_ADC Gain in Absolute Reference Mode

Table 11-75. SOC_ADC_ABS_GAIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	SOC_ADC_ABS_GAIN_TEMP1	R	X	SOC_ADC gain in absolute reference mode at temperature 1 (30C). Calculated in production test.. Default value holds log information from production test.

11.4.42 SOC_ADC_REL_GAIN Register (Offset = 360h) [Reset = 00000000h]

SOC_ADC_REL_GAIN is shown in Table 11-76.

Return to the Summary Table.

AUX_ADC Gain in Relative Reference Mode

Table 11-76. SOC_ADC_REL_GAIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	SOC_ADC_REL_GAIN_TEMP1	R	X	SOC_ADC gain in relative reference mode at temperature 1 (30C). Calculated in production test.. Default value holds trim value from production test.

11.4.43 SOC_ADC_OFFSET_INT Register (Offset = 368h) [Reset = 00000000h]

SOC_ADC_OFFSET_INT is shown in Table 11-77.

Return to the Summary Table.

AUX_ADC Temperature Offsets in Absolute Reference Mode

Table 11-77. SOC_ADC_OFFSET_INT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	SOC_ADC_REL_ OFFSET_TEMP1	R	X	SOC_ADC offset in relative reference mode at temperature 1 (30C). Signed 8-bit number. Calculated in production test.. Default value holds trim value from production test.
15-8	RESERVED	R	0h	Reserved
7-0	SOC_ADC_ABS_ OFFSET_TEMP1	R	X	SOC_ADC offset in absolute reference mode at temperature 1 (30C). Signed 8-bit number. Calculated in production test.. Default value holds trim value from production test.

11.4.44 SOC_ADC_REF_TRIM_AND_OFFSET_EXT Register (Offset = 36Ch) [Reset = 0000C080h]

SOC_ADC_REF_TRIM_AND_OFFSET_EXT is shown in Table 11-78.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-78. SOC_ADC_REF_TRIM_AND_OFFSET_EXT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	SOC_ADC_REF_VOLTAGE_TRIM_TEMP1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.45 AMPCOMP_TH1 Register (Offset = 370h) [Reset = FF7B828Eh]

AMPCOMP_TH1 is shown in Table 11-79.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-79. AMPCOMP_TH1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-18	HPMRAMP3_LTH	R	1Eh	Internal. Only to be used through TI provided API.
17-16	RESERVED	R	0h	Reserved
15-10	HPMRAMP3_HTH	R	20h	Internal. Only to be used through TI provided API.
9-6	IBIASCAP_LPTOHP_OL_CNT	R	Ah	Internal. Only to be used through TI provided API.
5-0	HPMRAMP1_TH	R	Eh	Internal. Only to be used through TI provided API.

11.4.46 AMPCOMP_TH2 Register (Offset = 374h) [Reset = 6B8B0303h]

AMPCOMP_TH2 is shown in Table 11-80.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-80. AMPCOMP_TH2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	LPMUPDATE_LTH	R	1Ah	Internal. Only to be used through TI provided API.
25-24	RESERVED	R	0h	Reserved
23-18	LPMUPDATE_HTM	R	22h	Internal. Only to be used through TI provided API.
17-16	RESERVED	R	0h	Reserved
15-10	ADC_COMP_AMPTH_LPM	R	0h	Internal. Only to be used through TI provided API.
9-8	RESERVED	R	0h	Reserved
7-2	ADC_COMP_AMPTH_HPM	R	0h	Internal. Only to be used through TI provided API.
1-0	RESERVED	R	0h	Reserved

11.4.47 AMPCOMP_CTRL1 Register (Offset = 378h) [Reset = FF483F47h]

AMPCOMP_CTRL1 is shown in Table 11-81.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-81. AMPCOMP_CTRL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	AMPCOMP_REQ_MODE	R	1h	Internal. Only to be used through TI provided API.
29-24	RESERVED	R	0h	Reserved
23-20	IBIAS_OFFSET	R	4h	Internal. Only to be used through TI provided API.
19-16	IBIAS_INIT	R	8h	Internal. Only to be used through TI provided API.
15-8	LPM_IBIAS_WAIT_CNT_FINAL	R	3Fh	Internal. Only to be used through TI provided API.
7-4	CAP_STEP	R	4h	Internal. Only to be used through TI provided API.
3-0	IBIASCAP_HPTOLP_OL_CNT	R	7h	Internal. Only to be used through TI provided API.

11.4.48 ANABYPASS_VALUE2 Register (Offset = 37Ch) [Reset = FFFFC3FFh]

ANABYPASS_VALUE2 is shown in Table 11-82.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-82. ANABYPASS_VALUE2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13-0	XOSC_HF_IBIASTHERM	R	3FFh	Internal. Only to be used through TI provided API.

11.4.49 VOLT_TRIM Register (Offset = 388h) [Reset = E0E0E0E0h]

VOLT_TRIM is shown in Table 11-83.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-83. VOLT_TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	Reserved
28-24	VDDR_TRIM_HH	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
23-21	RESERVED	R	0h	Reserved
20-16	VDDR_TRIM_H	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-13	RESERVED	R	0h	Reserved
12-8	VDDR_TRIM_SLEEP_H	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
7-5	RESERVED	R	0h	Reserved
4-0	TRIMBOD_H	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.50 OSC_CONF Register (Offset = 38Ch) [Reset = F00900E6h]

OSC_CONF is shown in Table 11-84.

Return to the Summary Table.

OSC Configuration

Table 11-84. OSC_CONF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29	ADC_SH_VBUF_EN	R	1h	Trim value for DDI_0_OSC:ADCDOUBLERNANOAMPCTL.ADC_SH_VBUF_EN.
28	ADC_SH_MODE_EN	R	1h	Trim value for DDI_0_OSC:ADCDOUBLERNANOAMPCTL.ADC_SH_MODE_EN.
27	ATESTLF_RCOSCLF_ IBIAS_TRIM	R	0h	Trim value for DDI_0_OSC:ATESTCTL.ATESTLF_RCOSCLF_IBIAS_TRIM.
26-25	XOSCLF_REGULATOR_ TRIM	R	0h	Trim value for DDI_0_OSC:LFOSCCTL.XOSCLF_REGULATOR_TRIM.
24-21	XOSCLF_CMIRRWR_ RATIO	R	0h	Trim value for DDI_0_OSC:LFOSCCTL.XOSCLF_CMIRRWR_RATIO.
20-19	XOSC_HF_FAST_START	R	1h	Trim value for DDI_0_OSC:CTL1.XOSC_HF_FAST_START.
18	XOSC_OPTION	R	X	0: XOSC_HF unavailable (may not be bonded out) 1: XOSC_HF available (default) Default value differs depending on partnumber.
17	HPOSC_OPTION	R	X	Internal. Only to be used through TI provided API. Default value differs depending on partnumber.
16	HPOSC_BIAS_HOLD_ MODE_EN	R	1h	Internal. Only to be used through TI provided API.
15-12	HPOSC_CURRMIRR_ RATIO	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-8	HPOSC_BIAS_RES_SET	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
7	HPOSC_FILTER_EN	R	1h	Internal. Only to be used through TI provided API.
6-5	HPOSC_BIAS_RECHARGE _DELAY	R	3h	Internal. Only to be used through TI provided API.
4-3	RESERVED	R	0h	Reserved
2-1	HPOSC_SERIES_CAP	R	3h	Internal. Only to be used through TI provided API.
0	HPOSC_DIV3_BYPASS	R	0h	Internal. Only to be used through TI provided API.

11.4.51 FREQ_OFFSET Register (Offset = 390h) [Reset = 00000000h]

FREQ_OFFSET is shown in Table 11-85.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-85. FREQ_OFFSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	HPOSC_COMP_P0	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-8	HPOSC_COMP_P1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
7-0	HPOSC_COMP_P2	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.52 MISC_OTP_DATA_1 Register (Offset = 398h) [Reset = E08403F8h]

MISC_OTP_DATA_1 is shown in Table 11-86.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-86. MISC_OTP_DATA_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	Reserved
28-27	PEAK_DET_ITRIM	R	0h	Internal. Only to be used through TI provided API.
26-24	HP_BUF_ITRIM	R	0h	Internal. Only to be used through TI provided API.
23-22	LP_BUF_ITRIM	R	2h	Internal. Only to be used through TI provided API.
21-20	DBLR_LOOP_FILTER_RESET_VOLTAGE	R	0h	Internal. Only to be used through TI provided API.
19-10	HPM_IBIAS_WAIT_CNT	R	100h	Internal. Only to be used through TI provided API.
9-4	LPM_IBIAS_WAIT_CNT	R	3Fh	Internal. Only to be used through TI provided API.
3-0	IDAC_STEP	R	8h	Internal. Only to be used through TI provided API.

11.4.53 SHDW_DIE_ID_0 Register (Offset = 3D0h) [Reset = 00000000h]

SHDW_DIE_ID_0 is shown in Table 11-87.

Return to the Summary Table.

Shadow of DIE_ID_0 register in eFuse

Table 11-87. SHDW_DIE_ID_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ID_31_0	R	X	Shadow of DIE_ID_0 register in eFuse row number 5 Default value depends on eFuse value.

11.4.54 SHDW_DIE_ID_1 Register (Offset = 3D4h) [Reset = 00000000h]

SHDW_DIE_ID_1 is shown in Table 11-88.

Return to the Summary Table.

Shadow of DIE_ID_1 register in eFuse

Table 11-88. SHDW_DIE_ID_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ID_63_32	R	X	Shadow of DIE_ID_1 register in eFuse row number 6 Default value depends on eFuse value.

11.4.55 SHDW_DIE_ID_2 Register (Offset = 3D8h) [Reset = 00000000h]

SHDW_DIE_ID_2 is shown in Table 11-89.

Return to the Summary Table.

Shadow of DIE_ID_2 register in eFuse

Table 11-89. SHDW_DIE_ID_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ID_95_64	R	X	Shadow of DIE_ID_2 register in eFuse row number 7 Default value depends on eFuse value.

11.4.56 SHDW_DIE_ID_3 Register (Offset = 3DCh) [Reset = 00000000h]

SHDW_DIE_ID_3 is shown in Table 11-90.

Return to the Summary Table.

Shadow of DIE_ID_3 register in eFuse

Table 11-90. SHDW_DIE_ID_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ID_127_96	R	X	Shadow of DIE_ID_3 register in eFuse row number 8 Default value depends on eFuse value.

11.4.57 SHDW_SCAN_MCU3_SEC Register (Offset = 3F8h) [Reset = 00000000h]

SHDW_SCAN_MCU3_SEC is shown in Table 11-91.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-91. SHDW_SCAN_MCU3_SEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	SECURITY	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
23	RESERVED	R	0h	Reserved
22-11	ULL_MCU_RAM_0_REP	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
10-0	ULL_MCU_RAM_1_REP_1	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.

11.4.58 SHDW_SCAN_DATA1_CRC Register (Offset = 3FCh) [Reset = 00000000h]

SHDW_SCAN_DATA1_CRC is shown in Table 11-92.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-92. SHDW_SCAN_DATA1_CRC Register Field Descriptions

Bit	Field	Type	Reset	Description
31	FLASH_RDY	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
30-9	RESERVED	R	0h	Reserved
8-1	CRC	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
0	TAP_DAP_LOCK_N	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.

11.4.59 SHDW_ANA_TRIM Register (Offset = 404h) [Reset = 00000000h]

SHDW_ANA_TRIM is shown in Table 11-93.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-93. SHDW_ANA_TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	ALT_VDDR_TRIM	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
29	DET_LOGIC_DIS	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
28-27	BOD_BANDGAP_TRIM_CNF_EXT	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
26-25	BOD_BANDGAP_TRIM_CNF	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
24	VDDR_ENABLE_PG1	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
23	VDDR_OK_HYS	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
22-21	IPTAT_TRIM	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
20-16	VDDR_TRIM	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
15-11	TRIMBOD_INTMODE	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
10-6	TRIMBOD_EXTMODE	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.
5-0	TRIMTEMP	R	X	Internal. Only to be used through TI provided API. Default value depends on eFuse value.

11.4.60 OSC_CONF1 Register (Offset = 408h) [Reset = 03FF0000h]

OSC_CONF1 is shown in Table 11-94.

Return to the Summary Table.

Oscillator configuration

Table 11-94. OSC_CONF1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RCOSC_MF_BIAS_HTEMP	R	X	Defines the MF_BIAS trim code to use for high temp. Only valid if RCOSC_MF_SINGLE_TRIM_METHOD == 0 Default value holds trim value from production test.
27	RCOSC_MF_TEMP_DEPEND_MODE	R	X	Defines whether dual trim was needed: 0: Dual trims needed on this chip 1: Dual trims not needed on this chip Default value holds trim value from production test.
26	RCOSC_MF_SINGLE_TRIM_METHOD	R	X	Defines trim method used: 0: Dual trim method 1: Single trim method Default value holds trim value from production test.
25-4	RESERVED	R	0h	Reserved
3-0	RCOSC_MF_BIAS_ADJ	R	X	Value is written to DDI_0_OSC:RCOSCMFCTL.RCOSC_MF_BIAS_ADJ by boot FW while in safezone. Default value holds trim value from production test.

11.4.61 DAC_BIAS_CNF Register (Offset = 40Ch) [Reset = FFFC00FFh]

DAC_BIAS_CNF is shown in Table 11-95.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-95. DAC_BIAS_CNF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17-12	LPM_TRIM_IOUT	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
11-9	LPM_BIAS_WIDTH_TRIM	R	0h	Internal. Only to be used through TI provided API.
8	LPM_BIAS_BACKUP_EN	R	0h	Internal. Only to be used through TI provided API.
7-0	RESERVED	R	0h	Reserved

11.4.62 TFW_PROBE Register (Offset = 418h) [Reset = 00000000h]

TFW_PROBE is shown in Table 11-96.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-96. TFW_PROBE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.63 TFW_FT Register (Offset = 41Ch) [Reset = 00000000h]

TFW_FT is shown in Table 11-97.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-97. TFW_FT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.64 DAC_CAL0 Register (Offset = 420h) [Reset = 00000000h]

DAC_CAL0 is shown in Table 11-98.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-98. DAC_CAL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SOC_DAC_VOUT_CAL_DECOUPLE_C2	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-0	SOC_DAC_VOUT_CAL_DECOUPLE_C1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.65 DAC_CAL1 Register (Offset = 424h) [Reset = 00000000h]

DAC_CAL1 is shown in Table 11-99.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-99. DAC_CAL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SOC_DAC_VOUT_CAL_PRECH_C2	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-0	SOC_DAC_VOUT_CAL_PRECH_C1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.66 DAC_CAL2 Register (Offset = 428h) [Reset = 00000000h]

DAC_CAL2 is shown in Table 11-100.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-100. DAC_CAL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SOC_DAC_VOUT_CAL_ADCREF_C2	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-0	SOC_DAC_VOUT_CAL_ADCREF_C1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

11.4.67 DAC_CAL3 Register (Offset = 42Ch) [Reset = 00000000h]

DAC_CAL3 is shown in Table 11-101.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 11-101. DAC_CAL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SOC_DAC_VOUT_CAL_VDDS_C2	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.
15-0	SOC_DAC_VOUT_CAL_VDDS_C1	R	X	Internal. Only to be used through TI provided API. Default value holds trim value from production test.

This page intentionally left blank.

Chapter 12

AES and Hash Cryptoprocessor

The AES and Hash Cryptoprocessor core of the CC13x4x10 and CC26x4x10 device platform features an advanced encryption standard (AES) module with local key storage, a SHA-2 module, and DMA capability.

12.1 Introduction 860
12.2 Functional Description 861
12.3 DMA Controller 872
12.4 AES and Hash Cryptoprocessor Performance 876
12.5 Programming Guidelines 877
12.6 Conventions and Compliances 907
12.7 CRYPTO Registers 910

12.1 Introduction

The AES and Hash Cryptoprocessor module provides hardware-accelerated data encryption, decryption, authentication, and hash operations.

Encryption converts plain text data to an unintelligible form called cipher text. Decrypting cipher text converts previously encrypted data back into the original plain text form. The AES algorithm implemented by the module is a symmetric algorithm, meaning that the encryption and decryption keys are identical. The main AES features of the module are:

- Support and availability of the following block modes:
  - Electronic code book mode (ECB)
  - Cipher block chaining mode (CBC)
  - Counter mode (CTR)
- Support and availability of the following Authenticated Encryption with Associated Data (AEAD) modes, with continuation support:
  - Counter mode with CBC-MAC (CCM)
  - Galois/Counter mode (GCM)
- Support and availability of the following Message Authentication Code modes, with continuation support:
  - Cipher block chaining message authentication code (CBC-MAC)
- Key sizes: 128 bits, 192 bits, and 256 bits
- Key scheduling in hardware
- Internal DMA for transferring data to and from system memory
- Fully synchronous design

The module also supports SHA-2 hash algorithms. SHA-2 provides a secure one-way conversion of data of arbitrary length to a fixed bit size output. The main SHA-2 features of the module are:

- Support and availability of the following SHA-2 hash algorithms:
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Internal DMA for transferring data from system memory
- Fully synchronous design

Note

Using the algorithm and modes supported by the AES and Hash Cryptoprocessor module, additional algorithms and modes can be accelerated with the addition of software. For example:

- AES CMAC can be accelerated by using the CBC-MAC support
- HMAC can be accelerated by using the SHA-2 support

Note

The defined output of block mode algorithms and SHA-2 hash algorithms includes all state that may influence the next block of data processed. However, AEAD and MAC modes of operation have additional state values which influence the next block of data processed. Continuation support allows SW to read these additional state values from the module and then reload that state to continue the AEAD or MAC operation at a later time.

Note

The AES and Hash Cryptoprocessor module does not provide concurrent AES and Hash operations.

12.2 Functional Description

The AES engine is directly connected to the context and data registers so that it can immediately start processing when all data is available. The AES engine also interfaces to the I/O-control FSM and the module's DMA. AES comprises the following major functional blocks:

- Global control FSM and DMA interface
- Register interface module
- The AES engine

The AES engine, which is the major top-level component, comprises the following functional blocks:

- Mode-control FSM: manages the data flow to and from the AES engine and starts each encryption and decryption operation
- Feedback modes: the logic that implements the various feedback modes supported by AES
- AES key scheduler: generates AES encryption and decryption (round) keys
- AES encryption core: the AES encryption algorithm
- AES decryption core: the AES decryption algorithm
- Substitution-boxes (S-boxes): contain AES S-Box GF(2^8) implementations.

For a key length of 128 bits, 10 rounds or 32 clock cycles are required, because {number of clock cycles} = 2 + 3 × {number of rounds}. While one data block processes, the next block can be preloaded immediately. When a block is preloaded, the previous block must finish before additional data can be loaded. Therefore, once the pipeline is full, sequential data blocks can be passed every 32 clock cycles.

The block and AEAD modes require reading and writing of data, whereas the MAC modes require only reading of the data. In addition to the writing of data, the AEAD modes return an authentication result (known as a MAC or TAG). This result can either be read with a separate DMA operation, or read through the slave interface. For all modes, an option provides (part of) the data through the slave interface instead of using DMA. The AES engine is forced to use keys from the key-store module for its operations. A key is provided to the AES engine by triggering the key-store module to read an AES key from the key store memory, and to write it to the AES key registers. The AES engine automatically pads or masks misaligned last data blocks with zeroes for AES CBC-MAC, CCM, and GCM. This includes padding Additional Authentication Data (AAD) for CCM and GCM. For AES CTR mode, misaligned last data blocks are internally masked to support non-block size input data.

The Hash engine supports basic SHA-256, SHA-224, SHA-512, and SHA-384 operations. It only requires reading of input data, this data is transferred through the DMAC modules. The hash results can be transferred either using a DMA operation or by reading it through the slave interface. The module also supports keyed hash operations like HMAC, in which part of data can be provided by the AHB slave interface instead of using the DMA.

12.2.1 Debug Capabilities

The AES and Hash Cryptoprocessor module provides the following status registers to monitor operations of the engine:

- DMA status and port-error status registers
- Interrupt status registers in the master control module
- Key-store module status register

12.2.2 Exception Handling

The AES and Hash Cryptoprocessor module can detect AHB master bus errors and abort the DMA operation. The AES key-store module can detect key-load errors and does not store the bad key in that case. In both cases, the status register in the master control module indicates the error.

12.2.3 Power Management and Sleep Modes

There is no retention logic for cryptography registers or the AES key-store. The clocks can be enabled or gated by the following PRCM registers:

- PRCM:SECDMACLKGR.CRYPTO_CLK_EN bit while in run mode
- PRCM:SECDMACLKGS.CRYPTO_CLK_EN bit while in sleep mode

- PRCM:SECDMACLKGDS.CRYPTO_CLK_EN bit while in deep-sleep mode

To save power, the application can disable the clock to the module when not in use. The module is clock-gated in sleep mode by setting the SECDMACLKGS register CRYPTO_CLK_EN bit. The module can also be clock-gated in run mode by setting the SECDMACLKGR register CRYPTO_CLK_EN bit.

12.2.4 Interrupts

The AES and Hash Cryptoprocessor module has two interrupt outputs; both are driven from the master control module and are controlled by the respective registers (see Section 12.2.6.3).

To enable interrupts for the module, the CRYPTO:IRQTYPE.LEVEL bit must be set and the interrupt source must be configured in the CRYPTO:IRQEN register.

The CRYPTO:IRQCLR register is available to clear an interrupt output and error-status bit. The CRYPTO:IRQSET register provides the software a way to test the interrupt connections and must be used for debugging only.

The CRYPTO:IRQSTAT register provides the status of the two interrupts and error status messages. The error status bits are asserted when they are detected, and typically the value of DMA_BUS_ERR and KEY_ST_WR_ERR signals are valid after the RESULT_AVAIL bit is asserted. The KEY_ST_RD_ERR bit is valid after triggering the key-store module to read a key from memory and providing it to the AES engine.

An interrupt RESULT_AVAIL is activated when an operation that uses DMA is finished. The signal asserts when both the DMA and internal module are in the IDLE state.

Another interrupt DMA_IN_DONE is activated when only the input DMA is finished and is intended for debugging.

Note

Interrupt outputs are not triggered for operations where the DMA is not used.

12.2.5 Module Memory Map

Table 12-1 lists the memory map details. See Section 12.7 for the descriptions.

Table 12-1. Detailed Memory Map(1)

Physical Address	Register Name	Type	Reset Value	Remark
DMA Controller Registers				
0x58024000	DMACH0CTL	R/W	0x00000000	Channel 0 control register
0x58024004	DMACH0EXTADDR	R/W	0x00000000	Channel 0 external address
0x5802400C	DMACH0LEN	R/W	0x00000000	Channel 0 DMA length
0x58024018	DMASTAT	R	0x00000000	DMAC status
0x5802401C	DMASWRESET	W	0x00000000	DMAC software reset
0x58024020	DMACH1CTL	R/W	0x00000000	Channel 1 control register
0x58024024	DMACH1EXTADDR	R/W	0x00000000	Channel 1 external address
0x5802402C	DMACH1LEN	R/W	0x00000000	Channel 1 DMA length
0x58024078	DMABUSCFG	R/W	0x00002400	Master run-time parameters
0x5802407C	DMAPORTERR	R	0x00000000	Port-error raw-status register
0x580240FC	DMAHWVER	R	0x01012ED1	DMAC-version register
Key-Storage Registers				
0x58024400	KEYWRITEAREA	R/W	0x00000000	Writer-area register
0x58024404	KEYWRITTENAREA	R/W	0x00000000	Written-area register
0x58024408	KEYSIZE	R/W	0x00000001	Key-size register
0x5802440C	KEYREADAREA	R/W	0x00000008	Read-area register

Table 12-1. Detailed Memory Map[1] (continued)

Physical Address	Register Name	Type	Reset Value	Remark
AES Engine Registers				
0x58024500 to 0x5802450C	AESKEY2_0 to AESKEY2_3	W	0x00000000	Clear/wipe AESKEY2_0 to AESKEY2_3 register
0x58024510 to 0x5802451C	AESKEY3_0 to AESKEY3_3	W	0x00000000	Clear/wipe AESKEY3_0 to AESKEY3_3 register
0x58024540 to 0x5802454C	AESIV_0 to AESIV_3	R/W	0x00000000	AES IV (LSW)
0x58024550	AESCTL	R/W	0x80000000	I/O and control mode
0x40024554	AESDATALEN0	W	0x00000000	Crypto data length (LSW)
0x40024558	AESDATALEN1	W	0x00000000	Crypto data length (MSW)
0x5802455C	AESAUTHLEN	W	0x00000000	AAD data length
0x58024560	AESDATAOUT0	R	0x00000000	Data output (LSW)
0x58024560	AESDATAIN0	W	0x00000000	Data input (LSW)
0x58024564	AESDATAOUT1	R	0x00000000	Data output
0x58024564	AESDATAIN1	W	0x00000000	Data input
0x58024568	AESDATAOUT2	R	0x00000000	Data output
0x58024568	AESDATAIN2	W	0x00000000	Data input
0x5802456C	AESDATAOUT3	R	0x00000000	Data output (MSW)
0x5802456C	AESDATAIN3	W	0x00000000	Data input (MSW)
0x58024570 to 0x5802457C	AESTAGOUT_0 to AESTAGOUT_3	W	0x00000000	Tag output (LSW)
Hash Engine Registers				
0x58024600	HASHDATAIN0	W	0x00000000	Data input bits [31:0] (LSW)
0x58024604	HASHDATAIN1	W	0x00000000	Data input bits [63:32]
0x58024608	HASHDATAIN2	W	0x00000000	Data input bits [95:64]
0x5802460C	HASHDATAIN3	W	0x00000000	Data input bits [127:96]
0x58024610	HASHDATAIN4	W	0x00000000	Data input bits [159:128]
0x58024614	HASHDATAIN5	W	0x00000000	Data input bits [191:160]
0x58024618	HASHDATAIN6	W	0x00000000	Data input bits [223:192]
0x5802461C	HASHDATAIN7	W	0x00000000	Data input bits [255:224]
0x58024620	HASHDATAIN8	W	0x00000000	Data input bits [287:256]
0x58024624	HASHDATAIN9	W	0x00000000	Data input bits [319:288]
0x58024628	HASHDATAIN10	W	0x00000000	Data input bits [351:320]
0x5802462C	HASHDATAIN11	W	0x00000000	Data input bits [383:352]
0x58024630	HASHDATAIN12	W	0x00000000	Data input bits [415:384]
0x58024634	HASHDATAIN13	W	0x00000000	Data input bits [447:416]
0x58024638	HASHDATAIN14	W	0x00000000	Data input bits [479:448]
0x5802463C	HASHDATAIN15	W	0x00000000	Data input bits [511:480]
0x58024640	HASHDATAIN16	W	0x00000000	Data input bits [543:512]
0x58024644	HASHDATAIN17	W	0x00000000	Data input bits [575:544]
0x58024648	HASHDATAIN18	W	0x00000000	Data input bits [607:576]
0x5802464C	HASHDATAIN19	W	0x00000000	Data input bits [639:608]
0x58024650	HASHDATAIN20	W	0x00000000	Data input bits [671:640]
0x58024654	HASHDATAIN21	W	0x00000000	Data input bits [703:672]
0x58024658	HASHDATAIN22	W	0x00000000	Data input bits [735:704]
0x5802465C	HASHDATAIN23	W	0x00000000	Data input bits [767:736]

Table 12-1. Detailed Memory Map[1] (continued)

Physical Address	Register Name	Type	Reset Value	Remark
0x58024660	HASHDATAIN24	W	0x00000000	Data input bits [799:768]
0x58024664	HASHDATAIN25	W	0x00000000	Data input bits [831:800]
0x58024668	HASHDATAIN26	W	0x00000000	Data input bits [863:832]
0x5802466C	HASHDATAIN27	W	0x00000000	Data input bits [895:864]
0x58024670	HASHDATAIN28	W	0x00000000	Data input bits [927:896]
0x58024674	HASHDATAIN29	W	0x00000000	Data input bits [959:928]
0x58024678	HASHDATAIN30	W	0x00000000	Data input bits [991:960]
0x5802467C	HASHDATAIN31	W	0x00000000	Data input bits [1023:992] (MSW)
0x58024680	HASHIOBUFCTRL	W	0x00000000	I/O buffer control
0x58024684	HASHMODE	W	0x00000000	Mode input register
0x58024688	HASHINLENL	W	0x00000000	Length input bits [31:0] (LSW)
0x5802468C	HASHINLENH	W	0x00000000	Length input bits [63:32] (MSW)
0x580246C0	HASHDIGESTA	R/W	0x00000000	Hash digest bits [31:0] (LSW)
0x580246C4	HASHDIGESTB	R/W	0x00000000	Hash digest bits [63:32]
0x580246C8	HASHDIGESTC	R/W	0x00000000	Hash digest bits [95:64]
0x580246CC	HASHDIGESTD	R/W	0x00000000	Hash digest bits [127:96]
0x580246D0	HASHDIGESTE	R/W	0x00000000	Hash digest bits [159:128]
0x580246D4	HASHDIGESTF	R/W	0x00000000	Hash digest bits [191:160]
0x580246D8	HASHDIGESTG	R/W	0x00000000	Hash digest bits [223:192]
0x580246DC	HASHDIGESTH	R/W	0x00000000	Hash digest bits [255:224]
0x580246E0	HASHDIGESTI	R/W	0x00000000	Hash digest bits [287:256]
0x580246E4	HASHDIGESTJ	R/W	0x00000000	Hash digest bits [319:288]
0x580246E8	HASHDIGESTK	R/W	0x00000000	Hash digest bits [351:320]
0x580246EC	HASHDIGESTL	R/W	0x00000000	Hash digest bits [383:352]
0x580246F0	HASHDIGESTM	R/W	0x00000000	Hash digest bits [415:384]
0x580246F4	HASHDIGESTN	R/W	0x00000000	Hash digest bits [447:416]
0x580246F8	HASHDIGESTO	R/W	0x00000000	Hash digest bits [479:448]
0x580246FC	HASHDIGESTP	R/W	0x00000000	Hash digest bits [511:480] (MSW)
Master-Control Registers				
0x58024700	ALGSEL	R/W	0x00000000	Algorithm selection
0x58024704	DMAPROTCTL	R/W	0x00000000	Enable privileged access on master
0x58024740	SWRESET	W	0x00000000	Master-control software reset
0x58024780	IRQTYPE	R/W	0x00000000	Interrupt-configuration register
0x58024784	IRQEN	R/W	0x00000000	Interrupt-enabling register
0x58024788	IRQCLR	W	0x00000000	Interrupt-clear register
0x5802478C	IRQSET	W	0x00000000	Interrupt-set register
0x58024790	IRQSTAT	R	0x00000000	Interrupt-status register
0x580247FC	HWVER	R	0x93028778	Version register

(1) Unspecified addresses are reserved and must not be written and ignored on a read

12.2.6 Master Control and Select Module

The master control module synchronizes the DMA operations and the cryptographic module handshake signals. In this module, the crypto algorithm is selected and the DMA burst sizes are defined. When the complete encryption operation completes, an interrupt is asserted.

Note

For authentication operations, the interrupt is asserted only if the authentication result is available.

The AES module also provides an interrupt to indicate that the input DMA transfer is complete. This interrupt is primarily used to determine the end of an AAD data DMA transfer (AES-CCM, AES-GCM), which is typically set up as separate input data transfer.

12.2.6.1 Algorithm Select Register

The Algorithm Select register configures the internal destination of the DMAC.

12.2.6.1.1 Algorithm Select

Table 12-2 summarizes the allowed bit combinations of the CRYPTO:ALGSEL register.

Table 12-2. Valid Combinations for ALGSEL Flags

Operation	Flags			
	KEY STORE	AES	Hash	TAG
Hash data is loaded through the DMA; the result digest is read by the slave interface.	0	0	1	0
Hash data is loaded through the DMA; the result digest is read by the DMA interface.	0	0	1	1
Key store is loaded through the DMA.	1	0	0	0
AES data is loaded through the DMA, and encrypted and decrypted data are read through the DMA (encryption and decryption). or AES data is loaded through the DMA, and the result tag is read through the slave interface (authentication-only operations).	0	1	0	0
AES data is loaded through the DMA; the result tag is read through the DMA (authentication-only operations).	0	1	0	1

12.2.6.2 Master PROT Enable

12.2.6.2.1 Master PROT-Privileged Access-Enable

The CRYPTO:DMAPROTCTL register selects the AHB transfer protection control for DMA transfers, using the key-store module as destination.

12.2.6.3 Software Reset

For more details on the soft reset procedure, see Section 12.5.5.1.

To perform a software reset of the AES module, perform the following steps in order:

1. Write 1 to the RESET bit in the CRYPTO:SWRESET register.
2. Write 1 to the CRYPTO:AESCTL register
3. Write 0 to the CRYPTO:AESCTL register
4. Write 1 to the CRYPTO:AESCTL register
5. Write 0 to the CRYPTO:AESCTL register
6. Write 0 to the CRYPTO:AESDATALEN0 and CRYPTO:AESDATALEN1 registers
7. Write 0 to the CRYPTO:AESAUTHLEN register
8. Write 0 to the CRYPTO:AESBLKCNT0 and CRYPTO:AESBLKCNT1 registers

When the software reset completes, the RESET bit in the SWRESET register is automatically reset. Software must make sure that the software reset completes before starting any operations.

In the DMA control module, software reset is used to reset the DMAC to stop all transfers and clear the CRYPTO:DMAPORTERR register. After the software reset is performed, all channels are disabled and no new requests are performed by the channels. The DMAC waits for the existing (active) requests to finish, then sets the DMAC status registers. To reset the DMAC, perform the following operations in order:

1. Write 1 to the SWRES bit of CRYPTO:DMASWRESET register
2. Write 0 to the CRYPTO:DMACH0CTL, CRYPTO:DMACH0LEN, CRYPTO:DMACH1CTL, and CRYPTO:DMACH1LEN registers
3. Wait for both of the CH0_ACT and CH1_ACT bits in CRYPTO:DMASTAT to be 0

12.2.7 AES Engine

The composition of the AES core is as follows:

- The main data path operates on the input block, performing the required substitution, shift, and mix operations.
- The key scheduler generates the round keys. A new subkey is generated and XORed with the data each round.

The AES key scheduler generates the round keys. During each round, a new subkey is generated from the input key to be XORed with the data. Round keys are generated on-the-fly and parallel to data processing to minimize register requirements. For encryption operations, the key sequencer transfers the initial key data to the AES core. For decryption operations, the key scheduler must provide the final subkey to the AES core so it can generate the subkeys in reverse order.

The AES core operates on the input block and performs the required substitution, shift, and mix operations. For each round, the encryption core receives the proper round key from the AES key scheduler. A fundamental component of the AES algorithm is the S-box. The S-box provides a unique 8-bit output for each 8-bit input.

The architecture of the AES decryption core is generally the same as the architecture of the encryption core. One difference is that the generation of round keys for decryption requires an initial conversion of the input key (always supplied by the host in the form of an encryption key) to the corresponding decryption key. This conversion is done by performing a dummy encryption operation and storing the final round key as a decryption key. The key scheduler is then reversed to generate the round keys for the decryption operation. Consequently, for each sequence of decryption operations under the same key, a single throughput reduction equal to the time to encrypt a single block occurs. When a decryption key is generated, subsequent decryption operations with the same key use this generated decryption key directly.

12.2.7.1 Second and Third Key Registers (Internal, but Clearable)

The registers listed in Table 12-3 and Table 12-4 are not accessible through the host for reading and writing. These registers are used to store internally calculated key information and intermediate results. However, when the host performs a write to any of the respective AESKEY2_0 to AESKEY2_3 or AESKEY3_0 to AESKEY3_3 addresses, respectively, the whole 128-bit AESKEY2_0 to AESKEY2_3 or AESKEY3_0 to AESKEY3_3 register is cleared to zeroes.

The AES_GHASH_H_IN_n registers (required for GHASH, which is part of GCM) are mapped to AESKEY2_n registers. The intermediate authentication result for GCM and CCM is stored in the AESKEY3_n register.

Table 12-3. AES_KEY

AESKEY20 to AESKEY2_3 (Write Only), 32-bit Address Offset: 0x500 to 0x50C in 0x4-byte increments																															
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
AESKEY2_0 to AESKEY2_3[31:0] AESKEY2_0 to AESKEY2_3[63:32] AESKEY2_0 to AESKEY2_3[95:64] AESKEY2_0 to AESKEY2_3[127:96]																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-4. AES_KEY

AESKEY3_0 to AESKEY3_3 (Write Only), 32-bit Address Offset: 0x510 to 0x51C in 0x4-byte increments																															
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
AESKEY3_0 to AESKEY3_3[31:0] / AESKEY2_0 to AESKEY2_3[159:128] AESKEY3_0 to AESKEY3_3[63:32] / AESKEY2_0 to AESKEY2_3[191:160] AESKEY3_0 to AESKEY3_3[95:64] / AESKEY2_0 to AESKEY2_3[223:192] AESKEY3_0 to AESKEY3_3[127:96] / AESKEY2_0 to AESKEY2_3[255:224]																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-5. For CCM

Bit	Field Name	Function
255–0	–	This register is used to store intermediate values.

Table 12-6. For CBC-MAC

Bit	Field Name	Function
255–0	Zeroes	This register must remain zero.

Table 12-7. For GCM

Bit	Name	Function
127–0	GHASH_H	The internally-calculated GHASH key is stored in these registers. Only used for modes that use the GHASH function (GCM).
255–128	–	This register is used to store intermediate values and is initialized with zeroes when loading a new key.

Reusing the AESKEYm_n registers is allowed for sequential operations; however, for CBC-MAC, intermediate values must be cleared when programming the respective mode and length parameters.

When performing a GCM operation without loading a new key (through the key store), a write to one of the AESKEY3_n register locations is required to clear the register.

If a CBC-MAC operation is started without loading a new key (through the key store), and the previous operation was not a CBC-MAC operation, both AESKEY2_0 to AESKEY2_3 and AESKEY3_0 to AESKEY3_3 register locations must be written before starting the CBC-MAC operation, which is required to clear these two key registers.

12.2.7.2 AES Initialization Vector (IV) Registers

Table 12-8 shows the AES Initialization Vector registers that are used to provide and read the IV from the AES engine.

Table 12-8. AES Initialization Vector Registers

AES_IV_0, (Read/Write), 32-bit Address Offset: 0x540 AES_IV_1, (Read/Write), 32-bit Address Offset: 0x544 AES_IV_2, (Read/Write), 32-bit Address Offset: 0x548 AES_IV_3, (Read/Write), 32-bit Address Offset: 0x54C																															
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
AES_IV[31:0] AES_IV[63:32] AES_IV[95:64] AES_IV[127:96]																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-9. Initialization Vector, Used for CBC and CTR Modes

Bits	Name	Description
127–0	AES_IV	For CBC and CTR block mode AES operations, these registers must be written with a new 128-bit IV. After an operation, these registers contain the latest 128-bit result IV, generated by the AES engine. If CTR mode is selected, this value is incremented with 0x1 (after first use) for each new data block is submitted to the engine.

Table 12-10. For GCM

Bits	Name	Description
127–0	AES_IV	For GCM operations, these registers must be written with a new 128-bit IV. After an operation, these registers contain the updated 128-bit result IV, generated by the AES engine. Bits [127–96] of the IV represent the initial counter value (which is 1 for GCM) and must therefore be initialized to 0x0100 0000. This value is incremented with 0x1 (after first use) for each new data block is submitted to the engine.

Table 12-11. Initialization Vector, Used for CCM

Bits	Name	Description
127–0	A0	For CCM, this field must be written with value A0. This value is the concatenation of: A0-flags (5 bits of zero and 3 bits L), nonce and counter value. L must be a copy from the L value of the CRYPTO:AESCTL register. This L indicates the width of the nonce and counter. The loaded counter must be initialized to zero. The total width of A0 is 128 bits. This value is incremented with 0x1 (after first use) for each new data block is submitted to the engine.

Table 12-12. Initialization Vector, Used for CBC-MAC

Bit	Name	Description
127–0	Zeroes	For CBC-MAC this register must be written with zeroes at the start of each operation. After an operation, these registers contain the 128-bit TAG output, generated by the crypto core.

12.2.7.3 AES I/O Buffer Control, Mode, and Length Registers

The AES I/O buffer control and mode register (CRYPTO:AESCTL) specifies the mode of operation for the AES engine.

Note

Internal operation of the AES module can be interrupted by setting all mode bits to 0 and writing zeroes to the length registers (CRYPTO:AESDATALEN0, CRYPTO:AESDATALEN1, and CRYPTO:AESAUTHLEN (see Section 12.7).

The length registers write the length values to the AES module. While processing, the length values decrement to 0. If both lengths are 0, the data stream is finished and a new context is requested. For basic AES modes (ECB, CBC, and CTR), a crypto length of 0 can be written if multiple streams must be processed with the same key. Writing a 0 length results in continued data requests until a new context is written. For the other modes (CBC-MAC, GCM, and CCM), no new data requests are done if the length decrements to or equals 0.

TI recommends writing a new length per packet. If the length registers decrement to 0, no new data is processed until a new context or length value is written.

When writing a new mode without writing the length registers, the values of the length register from the previous context are reused.

12.2.7.4 AES Data Input and Output Registers

The CRYPTO:AESDATAINn and CRYPTO:AESDATAOUTn data registers are typically accessed through DMA and not with host writes and reads. However, for debugging purposes, the data input and output registers can be

accessed through host write and read operations. The registers buffer the input and output data blocks to and from the crypto core (see Table 12-13).

Note

The data input buffer AESDATAINn and data output buffer AESDATAOUTn are mapped to the same address locations.

Writes (both DMA and host) to these addresses load the input buffer, while reads pull from the output buffer. Therefore, for write access, the data input buffer is written; for read access, the data output buffer is read. The data input buffer must be written before starting an operation. The data output buffer contains valid data when an operation completes. Therefore, any 128-bit data block can be split over multiple 32-bit word transfers; these transfers can be mixed with other host transfers over the external interface.

For normal operations, this register is not used, because data input and output are transferred from and to the AES core through DMA. For a host write operation, these registers must be written with the 128-bit input block for the next AES operation. Writing at a word-aligned offset within this address range stores the word (4 bytes) of data into the corresponding position of 4-word-deep (16 bytes = 128-bit AES block) data-input buffer. This buffer is used for the next AES operation. If the last data block is not completely filled with valid data, it can write only the words with valid data. Finally, the AES operation is triggered by writing the CRYPTO:AESCTL.INPUT_READY bit.

For a host read operation, this register contains the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range reads one word (4 bytes) of data out of the 4-word-deep (16 bytes = 128-bits AES block) data output buffer. The words (four words, one full block) must be read before the core moves the next block to the data output buffer. To empty the data output buffer, the CRYPTO:AESCTL.OUTPUT_READY bit must be written.

For the modes with TAG outputs (CBC-MAC, GCM, and CCM), the invalid (message) bytes or words can be written with any data.

Note

AES typically operates on a 128-bit block with multiple input data. The CTR, GCM, and CCM, modes form an exception per the standards defining those modes.

The last block of a CTR-mode message may contain less than 128 bits (see the NIST 800-38A document from National Institute of Standards and Technology). For CTR mode, the remaining data in an unaligned data block are ignored.

For CCM and GCM, the last blocks of both AAD and message data may contain less than 128 bits (see NIST SP800-38C, and NIST 800-38D from National Institute of Standards and Technology). The AES engine automatically pads with zeros AAD and message data as specified by referenced NIST standards.

While last blocks may be less than 128-bits in these cases, blocks must be a multiple of 8-bits since that is the minimum data transfer size for the system.

Table 12-13. Input/Output Block Format per Operating Mode

Operation	Data Input Buffer	Data Output Buffer
ECB/CBC encrypt	128-bit plaintext block	128-bit ciphertext block
ECB/CBC decrypt	128-bit ciphertext block	128-bit plaintext block
CTR encrypt	n-bit plaintext block	n-bit ciphertext block
CTR decrypt	n-bit ciphertext block	n-bit plaintext block
GCM/CCM AAD data	n-bit plaintext block	No output data
GCM/CCM encrypt data	n-bit plaintext block	n-bit ciphertext block
GCM/CCM decrypt data	n-bit ciphertext block	n-bit plaintext block

Table 12-13. Input/Output Block Format per Operating Mode (continued)

Operation	Data Input Buffer	Data Output Buffer
CBC-MAC data	n-bit plaintext block	No output data

12.2.7.5 TAG Registers

Table 12-14 shows the TAG registers that buffers the TAG from the AES module and can be accessed through DMA or directly with host reads. The TAG registers are shared with the intermediate authentication result registers, but cannot be read until the processing is finished. While processing, a read from these registers returns zeroes. If an operation does not return a TAG (also known as a MAC), reading from these registers returns an initialization vector (IV). If an operation returns a TAG plus an IV and both must be read by the host, the host must first read the TAG followed by the IV. Reading these in reverse order returns the IV twice.

For a host-read operation, these registers contain the last 128-bit TAG output of the AES core. The TAG is available until the next context is written. This register contains valid data only if the TAG is available, and when the SAVED_CONTEXT_RDY bit in the AESCTL register is set. During processing or for operations and modes that do not return a TAG, reads from this register return data from the IV register.

Table 12-14. AES Tag Output Register

AESTAGOUT_0 to AESTAGOUT_3, (Read Only), 32-bit Address Offset: 0x570 to 0x57C in 0x4 byte increments

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
AES_TAG[31:0] AES_TAG[63:32] AES_TAG[95:64] AES_TAG[127:96]																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-15. For GCM, CCM, and CBC-MAC

Bit	Field Name	Description
127–0	TAG	This register contains the authentication TAG for the combined and authentication-only modes.

12.2.8 Key Area Registers

The local-key storage module is directly connected to 1 KB of memory. The module can store up to eight AES keys and has eight 128-bit entries. 192-bit and 256-bit keys consume two entries each. The key size is programmed in the key-store module. The key material in the key store is not accessible through read operations through the AHB master and slave interfaces.

Keys can only be written to the key store through DMA. Once a DMA operation for a key read is started, all received data is written to the key-store module. Keys that are stored in the key store memory can be transferred only to the AES key registers and are not accessible for any other purpose.

12.2.8.1 Key Store Write Area Register

The Key Store Write Area register (CRYPTO:KEYWRITEAREA) defines where the keys must be written in the key store RAM. After writing the Key Write Area register, the key-store module is ready to receive the keys using a DMA operation. If the key data transfer triggered an error in the key store, the error is available in the interrupt status register, CRYPTO:IRQSTAT, after the DMA is finished. The key store write-error, KEY_ST_WR_ERR, is asserted when the programmed or selected area is not completely written. This error is also asserted when the DMA operation writes to RAM areas that are not selected.

For more details, see Section 12.7.

12.2.8.2 Key Store Written Area Register

The Key Store Written Area register (CRYPTO:KEYWRITTENAREA) shows which areas of the key store RAM contain valid written keys.

When a new key must be written to the key store on a location that is already occupied by a valid key, this key area must be cleared first. Clear the key area by writing this register before the new key is written to the key store memory.

Trying to write to a key area that already contains a valid key is not allowed and results in an error.

For more details, see Section 12.7.

12.2.8.3 Key Store Size Register

The Key Store Size register (CRYPTO:KEYSIZE) defines the size of the keys that are written with DMA. The Key Size register must be configured before writing to the CRYPTO:KEYWRITEAREA register.

For more details, see Section 12.7.

12.2.8.4 Key Store Read Area Register

The Key Store Read Area register (CRYPTO:KEYREADAREA) selects the key store RAM area from where the key must be read that is used for an AES operation. The operation starts directly after writing this register. When the operation is finished, the status of the key store read operation is available in the CRYPTO:IRQSTAT Interrupt Status register. KEY_ST_RD_ERR asserts when a RAM area is selected that does not contain a valid written key.

For more details, see Section 12.7.

12.2.9 Hash Engine

The SHA-2 module is connected through the wrapper module to the register interface or the DMA interface; only one at a time is active. All Hash module registers, except for the I/O control register, drive the interface of the Hash engine. Only the I/O Control register has handshake logic associated with it.

12.2.9.1 Hash I/O Buffer Control and Status Register, Mode, and Length Registers

Setup of the Hash engine is accomplished with the CRYPTO:HASHMODE, CRYPTO:HASHIOBUFCTRL, CRYPTO:HASHINLENL, and CRYPTO:HASHINLENH registers.

The HASHMODE register provides a bit field for each of the SHA-224, SHA-256, SHA-384, and SHA-512 modes. Only one bit should be set, the others must be cleared. The HASHMODE register also contains a NEW_HASH bit. Setting this bit to 1 will cause the Hash engine to load the appropriate initial digest values into the CRYPTO:HASHDIGESTn registers.

The HASHIOBUFCTRL provides several control bits used when the host writes the Hash Data Input registers (CRYPTO:HASHDATAINn) and reads the Digest registers, CRYPTO:HASHDIGESTA - CRYPTO:HASHDIGESTP (as opposed to the DMA performing the reads and writes). See Section 12.2.9.2 for more information. In addition, the CRYPTO:HASHIOBUFCTRL.PAD_DMA_MESSAGEbit is provided. Set this bit to 1 when the DMA has been configured to load the last block of the message. When set to 1, the Hash engine will finalize the digest value. If the DMA is not being used or if the digest is not (yet) to be finalized, this bit must be set to 0.

The HASHINLENL and HASHINLENH are written to indicate the length, in bits, of the message being hashed. The HASHINLENL should we written first, and then the HASHINLENH since a write to HASHINLENL will clear HASHINLENH. When a hash is to be finalized, the length value written must be the total length of the message, including any bits of the message processed in a prior hash operation.

12.2.9.2 Hash Data Input and Digest Registers

The CRYPTO:HASHDATAINn and CRYPTO:HASHDIGESTA to CRYPTO:HASHDIGESTP data registers are typically accessed through DMA and not with host writes and reads. However, to support debugging, continuation of hash operations, and HMAC, the Data Input and Digest registers can be accessed through host write and read operations. The registers buffer the input and output data blocks to and from the crypto core.

Writes (both DMA and host) to the Data Input registers load the input buffer for the Hash engine. For message lengths larger than 64 bytes, multiple blocks of data are written to this input buffer using a handshake via flags of the CRYPTO:HASHIOBUFCTRL register. All blocks except the last are required to be a full block size in bits

(512 bits for SHA-224 and SHA-256, 1024 bits for SHA-384 and SHA-512). If the last block is not a full block, only the least significant bits of data have to be written, but they have to be padded with zeroes to the next 32-bit boundary.

When writing the Data Input registers using the host (and not the DMA), the RFD_IN, DATA_IN_AV, and PAD_MESSAGE bits of HASHIOBUFCTRL are used to determine when writes to the Data Input registers are allowed, when input is ready for processing by the hash engine, and how the hash engine should handle padding, respectively.

The Digest registers may be read by the DMA or host. When read by the host, the GET_DIGEST bit of the HASHIOBUFCTRL register must be written to a 1, to instruct the Hash engine to output the digest value. OUTPUT_FULL bit of HASHIOBUFCTRL is used by the hash engine to indicate that the Digest registers contain new values for reading. After the digest is read, the OUTPUT_FULL bit must be written to a 1 value to allow the Hash engine to process additional inputs.

Intermediate (non-finalized) hash digest values may be read from the Digest registers and then later written back to the Digest registers in order to extend a hash operation with additional input data.

Table 12-16. Input/Digest Block Format for Hash Algorithms

Algorithm	Data Input Block Size	Digest Output Size
SHA-224	512 bits (HASHDATAIN0 through HASHDATAIN15	224 bits (HASHDIGESTA through HASHDIGESTG)
SHA-256	512 bits (HASHDATAIN0 through HASHDATAIN15)	256 bits (HASHDIGESTA through HASHDIGESTH)
SHA-384	1024 bits (HASHDATAIN0 through HASHDATAIN31)	384 bits (HASHDIGESTA through HASHDIGESTL)
SHA-512	1024 bits (HASHDATAIN0 through HASHDATAIN31)	512 bits (HASHDIGESTA through HASHDIGESTP)

12.3 DMA Controller

The AES and Hash Cryptoprocessor contains its own dedicated DMA controller (DMAC).

Figure 12-1 shows the DMAC and its integration in the AES and Hash Cryptoprocessor.

The module is configured by the DMA configuration CRYPTO:DMABUSCFG register (see Section 12.7) and performs single 8-bit or 32-bit nonsequential single transfers by default. Transfer addresses and length parameters of the DMA transfer are byte aligned.

Note

The CC13x4x10 and CC26x4x10 device platform does not support burst or non-sequential transfers through internal interconnect. The DMABUSCFG register must not be changed for proper operation.

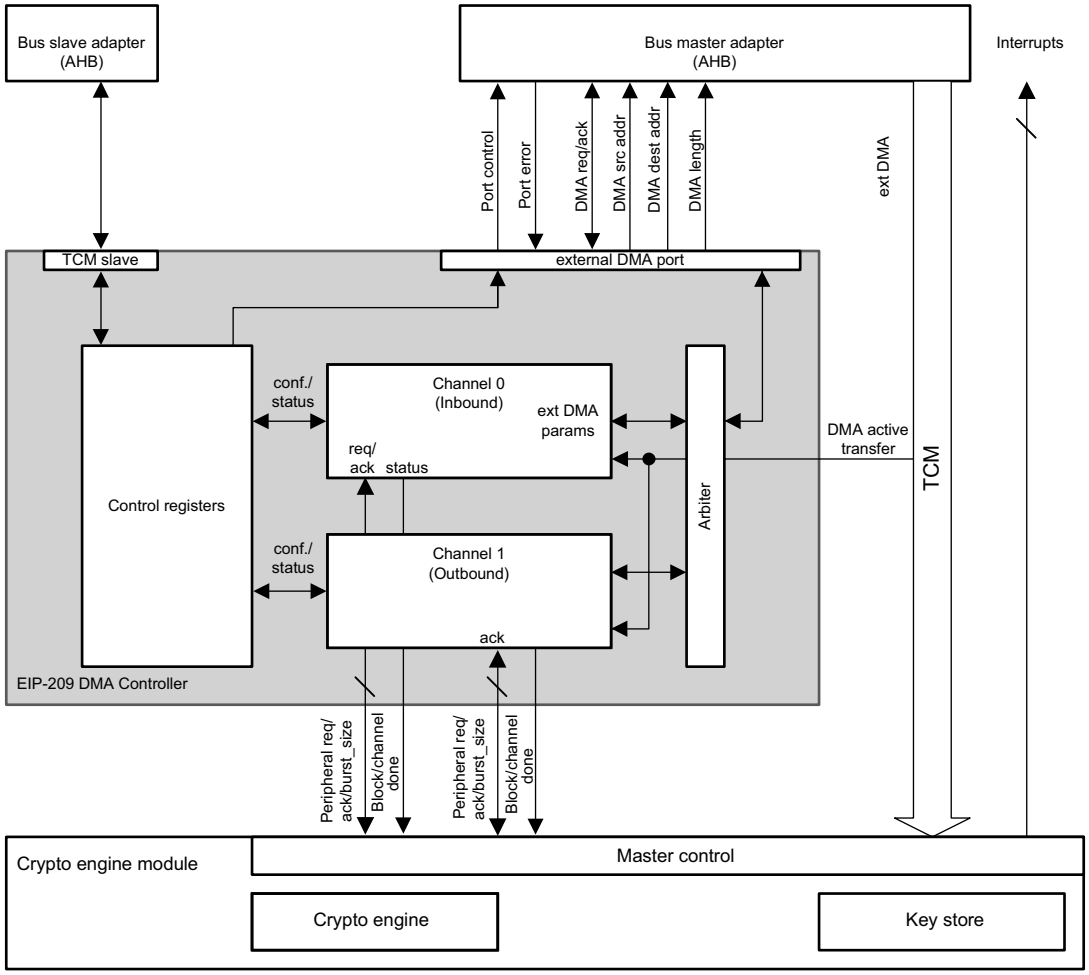


Figure 12-1. DMA Controller and Integration

The DMAC of the module controls the data transfer requests to the AHB master adapter, which transfers data to and from the AES engines and key store area.

The required parameters for proper functioning of the AHB master interface port are defined in the DMABUSCFG register. The default configuration of this register configures fixed-length transfers and a maximum burst size of 4 bytes. As a result, only nonsequential single transfers are performed on the AHB bus.

The CRYPTO:DMASTAT and CRYPTO:DMAPORTERR registers provide the actual state of each DMA channel and individual AHB port errors. A port error aborts operations on all serviced channels and prevents further transfers using that port, until the error is cleared by writing to the CRYPTO:DMASWRESET register.

If the address and lengths are 32-bit aligned, the master does only NONSEQ-type and SINGLE-type transfers with a size of 4 bytes.

The DMAC splits channel DMA operation into small DMA transfers. The size of small DMA transfers is determined by the target internal module, and equals the block size of the cryptographic operation.

The DMAC has the following features:

- Two channels (one inbound and one outbound) that can be enabled at the same time
- A maximum size of the DMA operation, controlled by a 16-bit-long register
- An arbiter to schedule channel accesses to the external AHB port
- Functionality to capture external bus errors

The DMAC consists of two DMA channels with programmable priority: one is programmable to move input data and keys from the external memory to the AES module, and another is programmable to move result data from the AES module to the external memory. Access to the channels of the AHB master port is handled by the arbiter module.

Channel control registers are used for channel enabling and priority selection. When a channel is disabled, it becomes inactive only when all ongoing requests are finished.

Note

All the channel control registers (CRYPTO:DMACHxCTL, CRYPTO:DMACHxEXTADDR, and CRYPTO:DMACHxLEN) must be programmed by the host to start a new DMA operation.

The DMAC transfers data between a source address and a destination address. Starting at a nonword-aligned boundary, byte transfers are generated until a word boundary is reached. Word transfers are then generated as long as data are available. If the transfer does not finish on word-aligned address, the remaining transfers are again byte transfers.

Note

No halfword transfers are generated.

The DMAC registers are mapped to the external register map. To start the operation, the host must program the mode of the DMAC and parameters of the operation. These parameters involve direction (read, write, or read-and-write), length (1 to 65535 bytes), external source address (for reading), and external destination address (for writing). For details of the registers, see Section 12.7.

Note

The internal destination is programmed using a dedicated algorithm selection register in master control module. The burst size is provided to the DMAC based on the setting of that register.

12.3.1 Internal Operation

The DMAC operates with the AHB master adapter that has two ports. One port is an external AHB port used to perform read and write operations to the external AHB subsystem. This port can address the complete 32-bit address range. The second port is an internal TCM port (master TCM) used to perform read and write operations to the internal modules of the crypto core AES engine, hash engine, and key store. Assignment of the internal

modules for DMA operation must be selected in the master control module (see Section 12.2.6.1.1); therefore, an internal address is not needed in the DMAC.

The data path from the TCM port of the AHB master module to the internal modules is located outside of the DMAC. The DMAC only observes the number of transferred words to determine when the requested DMA operation is finished for the corresponding channel.

The key store is a 32-bit block of memory with a depth of 32 words, surrounded by control logic. When the AES module is configured to write keys to this key-store module through DMA, the key store internally manages access to the key store RAM based on its register settings (including generation of the key store RAM addresses). The AES module supports only DMA write operations to the key store.

The AES engine has a 32-bit write interface for input data to be encrypted or decrypted, and a 32-bit read interface for result data and tag. The write interface of the AES module collects 32-bit data into a 128-bit input block (AES block size). When a full block is received, the AES calculation for the received block is started. When receiving the last word of the last block, the DMAC and master controller generate a "data done" signal to the crypto engine. The mode, message length, and optional parameters are programmed using the target interface.

The hash engine has a 32-bit write interface for input data, for the data to be hashed. It also has a 32-bit read interface to (optionally) read the result hash digest. The write interface of the Hash engine collects the 32-bit data into the input block (512-bit or 1024-bit size depending on the algorithm). When a full block is received, the hash calculation for the received block is started. When receiving the last word of the last block, the DMAC and master controller generate a "data done" signal to the hash engine. The mode and message length are programmed using the target interface.

On the TCM side, the key-store module immediately accepts all data without delay cycles, while the crypto modules operate on a data block boundary. On the TCM side, the key-store module immediately accepts all data without delay cycles, while the crypto module operates on a data block boundary (the processing of which takes a number of clock cycles). Special handshake signals are used between the DMAC and crypto modules:

- A data input request is sent to the DMA inbound channel (channel 0) when the crypto module can accept the next data block.
- A data output request is sent to the DMA output channel (channel 1) when the crypto module has the next block of data or tag available, after processing or Hash module has a digest available.
- Both channels send an acknowledge when the DMA operation starts, channel transfer completes, when a block has been transmitted and the channel transfer completes, or when all data is transmitted.

12.3.2 Supported DMA Operations

With each data request from the crypto engine, the DMAC requests a transfer from the AHB master. The transfer size is at most the block size of the corresponding algorithm. This block size depends on the selected algorithm in the master control module.

Table 12-17 provides a summary of the supported DMAC operations. The module refers to the selected module in the master control module. In Table 12-17, the *TAG enabled* label indicates that the TAG bit is set in the master control configuration register. TAG refers to the MAC output for AEAD and MAC AES modes and the digest output for SHA-2.

Table 12-17. Supported DMAC Operations

Module	Incoming Data Stream (for Channel 0)		Outcoming Data Stream (for Channel 1)	
	Source	Destination	Source	Destination
Key Store	External memory location	Key store RAM	-	-
Crypto	RAM (Authentication data only)	AES	(1)	(1)
	External memory location	AES	AES	External memory location
	(2)	(2)	AES (TAG enabled)	External memory location

Table 12-17. Supported DMAC Operations (continued)

Module	Incoming Data Stream (for Channel 0)		Outcoming Data Stream (for Channel 1)	
	Source	Destination	Source	Destination
Hash	External memory location	SHA-2 (TAG disabled)	(1)	(1)
	(2)	(2)	SHA-2 (TAG enabled)	External memory location
	External memory location	SHA-2 (TAG disabled)	SHA-2 (TAG enabled)	External memory location

(1) TAG is transferred through the slave interface or transferred with a separate DMA
(2) Data is transferred through another DMA, that has been executed before

12.4 AES and Hash Cryptoprocessor Performance

12.4.1 Introduction

The processing steps of the AES and Hash Cryptoprocessor module are the basis for the performance calculations. The following three major steps are identified for crypto operations using DMA:

1. Initialization (setup and initialization of the engines, DMA, and so forth)
2. Data processing for the complete message
3. Finalization (reading out the result, status checking)

The orange sections (full processing) of Figure 12-2 correspond to Step 1 and Step 3. Step 1 and Step 3 are under control of the host CPU, and therefore are dependent on the performance of the host. Step 2 corresponds to the green section (data processing) in Figure 12-2 and is fully handled by the hardware, which is not dependent on the performance of the host CPU.

The full processing part is required once per processing command, and precedes the processing of the first data block. The data processing blocks depend on the amount of data to be processed by the command. The finalization is required when the operation produces a result digest or TAG.

The number of required blocks is determined by the block size requirements of the algorithms selected by the command. The AES block size is 128 bits.

The Hash block size is 512 bits for SHA-256 and SHA-224, and the block size is 1024 bits for SHA-512 and SHA-384.

For longer data streams, the data processing time approaches the theoretical maximum throughput. For operations that use the slave interface as alternative for the DMA, the performance depends on the performance of the host CPU.

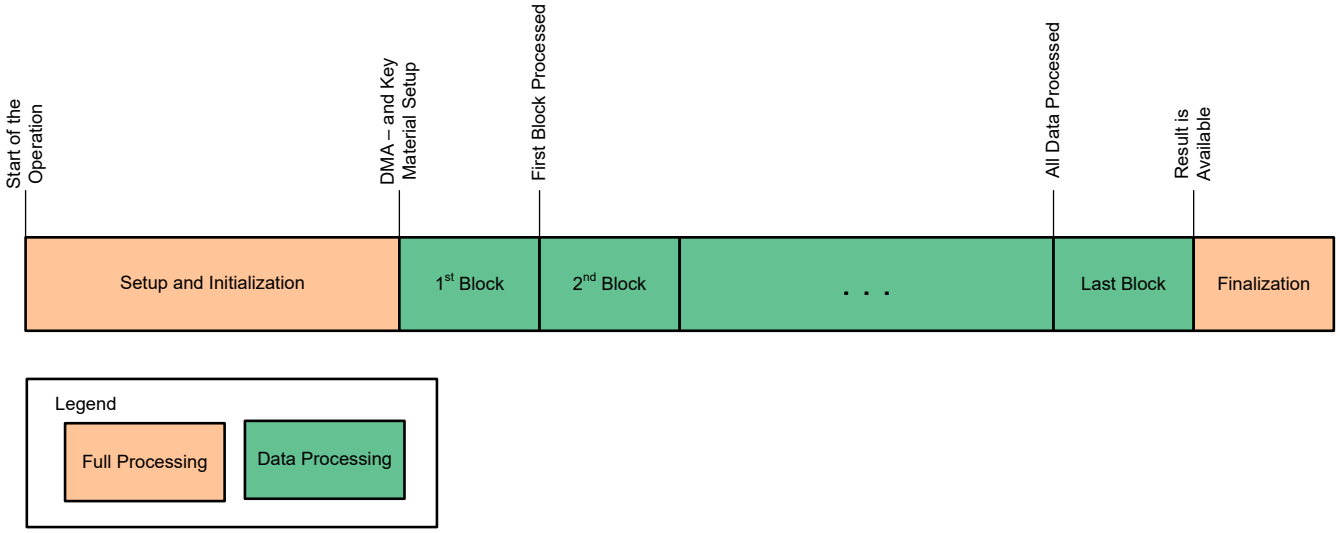


Figure 12-2. Symmetric Crypto Processing Steps

12.4.2 Performance for DMA-Based Operations

Table 12-18 shows the performance of the AES module running at 48 MHz for DMA-based cryptographic operations.

Table 12-18. Performance Table for DMA-Based Operations

Performance in Mbps at 48 MHz				
Crypto Mode	**Raw Engine Performance**	**1-Block Packet Performance(1)**	**20-Block Performance(1)**	**100-Block Performance(1)**
AES-128 (1 block = 128 bits)				
AES-128-ECB	118	26	100	114
AES-128-CBC	115	24	97	111
AES-128-CTR	118	24	99	113
AES-128-GCM(2)	118	18	91	110
AES-192 (1 block = 128 bits)				
AES-192-ECB	98	25	86	96
AES-192-CBC	97	24	84	94
AES-192-CTR	98	24	85	96
AES-192-GCM(2)	98	17	79	93
AES-256 (1 block = 128 bits)				
AES-256-ECB	85	24	75	83
AES-256-CBC	84	23	74	81
AES-256-CTR	85	23	75	83
AES-256-GCM(2)	85	15	69	80
Hash (SHA-256: 1 block = 512 bits; SHA-512: 1 block = 1024 bits)				
SHA-256	378	90	325	366
SHA-512	606	161	533	590

(1) The performance assumes full programming of the engine, loading keys, and setting up the DMA engine through the DMA slave. If the context is reused (mode or keys), the performance is increased. The maximum number of cycles overhead per packet is from 100 to 150 for the various modes and algorithms.
(2) AES-GCM raw performance numbers exclude the final operation to create the TAG; the block performance numbers include this overhead.

The engine performance depends heavily on the number of blocks processed per operation. Processing a single block results in the minimum engine performance; in this case, the configuration overhead is the most significant (assuming the engine is fully reconfigured for each operation). Therefore, processing multiple blocks per operation results in a significantly higher performance.

12.5 Programming Guidelines

This section describes the low-level programming sequences for configuring and using the AES module for the supported use cases.

12.5.1 One-Time Initialization After a Reset

The purpose of the initialization is to set the AES module into the initial mode common to all used operations. Perform the following initialization steps after a hardware reset:

1. Program the DMAC run time parameters in the CRYPTO:DMABUSCFG register with the desired values common for all DMA operations.
2. Initialize the desired interrupt type (level), and enable the interrupt output signal CRYPTO:IRQEN.RESULT_AVAIL in the master control module.

12.5.2 DMAC and Master Control

This section contains general guidelines on how to program the DMAC to perform a specific operation.

12.5.2.1 Regular Use

Program the following registers to configure the DMA channels (see Section 12.7 for details):

- Clear any outstanding interrupts and error flags if possible (see CRYPTO:IRQCLR).
- The master control module Algorithm Select register must be programmed to allow a DMA operation on the required internal module, which enables the DMA/AHB Master clock, and keeps it enabled until the clock is disabled by the host (see CRYPTO:ALGSEL).
- Channel n Control registers with channel bits enabled (see CRYPTO:DMACH0CTL and CRYPTO:DMACH1CTL).
- Channel n External Address registers (see CRYPTO:DMACH0EXTADDR and CRYPTO:DMACH1EXTADDR).
- Channel DMA n Length registers. Writing this register starts the DMA operation on the corresponding channel (see CRYPTO:DMACH0LEN and CRYPTO:DMACH1LEN).
- A complete operation is indicated by the result available interrupt output or the corresponding status register. Clear the interrupt after handling the interrupt (see CRYPTO:IRQSTAT and CRYPTO:IRQCLR).
- Master control module Algorithm Select register must be cleared to 0 to switch off the DMA/AHB Master clock (see CRYPTO:ALGSEL).

Note

The IRQSTAT register must be checked for possible errors if bus errors can occur (which are typically valid in a debugging phase) in the system or in systems where bus errors can occur during a DMA operation.

12.5.2.2 Interrupting DMA Transfers

To stop a DMA transfer and abort the operation, the host disables a channel using the CRYPTO:DMACHnCTL registers. When the EN bit of this register is set to 0, no new DMA transfer is requested by this channel and the current active transfer is finished. Alternatively, all active channels can be stopped by activating the DMAC soft reset with the CRYPTO:DMASWRESET register.

Note

When stopping the DMAC, the host must stop all active channels.

The state of the DMAC channel must be checked using the CRYPTO:DMASTAT register. When the CHx_ACT bit of this register for the disabled channel is set to 0, the DMAC channel stops.

To stop the DMAC in combination with the AES engine, the AES engine must be set in Idle mode first, which is done by writing zeroes to the length registers, followed by disabling all modes in the CRYPTO:AESCTL register.

Stopping the DMAC channels might leave the master control module in an unfinished state, due to pending events from the engines that will never occur. Therefore, to correctly recover the engine, the SWRESET register must issue the master control soft reset after all active DMAC channels are stopped.

12.5.2.3 Interrupts, Hardware, and Software Synchronization

This section describes the important relation of the RESULT_AVAIL interrupt activation and the data writing completion of the DMAC inside the crypto core.

The RESULT_AVAIL interrupt is activated when the AHB master finishes the data write transfer from the crypto core and the internal operation is completed. However, that does not ensure that data has been written to the external memory, due to latency from the AHB master to the destination (typically a memory). This latency might occur in the AHB bus subsystem outside of the crypto core, because this system possibly contains bridges.

Note

If this latency can occur, the host must ensure (using a time-out or other synchronization mechanisms) that external memory reads are performed only after all memory write operations are finished.

12.5.3 Hashing

The Hash engine has the following interfaces:

- Hash engine accepts data from two sources: AHB slave interface and DMA. Within one operation, it is possible to combine data from these two sources: write data from the slave interface, and write data from the DMA interface to complete Hash operations.
- Input digest (for resumed hash) and length must be programmed through the register interface.
- Result digest can be read through the slave interface or DMA.

12.5.3.1 Data Format and Byte Order

In most systems, the message data in stored in the host memory in little-endian fashion. The Hash engine is designed as a little-endian core to prevent data swap in the system. As a result, the message data must be provided in a little-endian fashion to the core.

The following code examples show how the data must be provided to the Hash engine, based on the FIPS 180-4 specification.

Note

The byte highlighted in red is the first byte of the data or digest block.

SHA-256 Example:

```
Data in:"abcdbcdecdefdefgefghfghighijhi
jkijkljklmklmnlmnomnopnopq"
"61626364 62636465 63646566 64656667
 65666768 66676869 6768696a 696a6b6c
 68696a6b 6a6b6c6d 6b6c6d6e 6c6d6e6f
 6d6e6f70 6e6f7071
Digest out:"248d6a61 d20638b8 e5c02693 0ce6039
```

Hash data in external RAM, loaded through DMAC:

```
Word_0 [31:0]64636261
Word_1 [31:0]65646362
Word_2 [31:0]66656463
...
```

Output digest, read through the slave interface or DMA:

```
HASHDIGESTA [31:0]:616a8d24
HASHDIGESTB [31:0]:b83806d2
HASHDIGESTC [31:0]:9326c0e5
```

12.5.3.2 Basic Hash with Data From DMA

The Hash engine hashes the data received from the external memory through DMA, and may be programmed to store the result digest into external memory using a DMA operation. The typical sequence for using the EIP-120t for operations follow:

- The Hash engine mode is programmed through the slave interface.
- For resumed Hash operations, the intermediate digest of the Hash engine from a previous session is programmed through the slave device. If the resumed operation is a continuation of the previous (unfinished) Hash operation, the Hash engine holds its current state. In this case, it may be reused for the next operation, and the initial digest does not need to be programmed.
- The master controller is programmed to move data to the Hash engine. If the result digest must be written to the external memory, the master controller should be programmed accordingly.
- DMAC channel 0 is programmed to read data from the external memory and copy it to the data input port of the Hash engine.
- If the result digest must be written to the external memory, DMAC channel 1 is programmed to store the result digest into a pre-allocated area in the external memory.
- The interrupt status is observed to check when the Hash engine has completed the operation.
- If the result digest must be ready by the host, it can be read through the slave interface. If the result digest is written to external memory through the DMA, it is available in external memory (see Section 12.5.2.3).

12.5.3.2.1 New Hash Session with Digest Read Through Slave

The following software example in pseudocode describes the actions that are typically executed by the host software. The example starts the Hash engine with a new Hash session that receives the input data through the DMA interface. In the end, the intermediate digest (nonfinal Hash operation) or the finalized Hash digest (final Hash operation) is read as a result digest through the slave interface.

```
// configure master control module
write ALGSEL 0x00000008       // enable DMA path to the SHA-2 engine
write IRQCLR 0x00000001       // clear any outstanding events
// configure hash engine
write HASHMODE = 0x00000021   // indicate the start of a new hash session and SHA512
write HASHINLENL              // write the length of the message (lo)
write HASHINLENH              // write the length of the message (hi)
// if the final digest is required (pad the input DMA data), write the following register
write HASHIOBUFCTRL = 0x80    // pad the data that is transferred via DMA
// configure DMAC
write DMACH0CTL 0x000000001              // enable DMA channel 0 for message data
write DMACH0EXTADDR <ext_memory_address> // base address of the data in ext. memory
write DMACH0LEN <length>                 // input data in bytes, equal to the message length
wait IRQSTAT[0] = '1'                    // wait for operation done (hash and DMAC are ready)
check IRQSTAT[31] == '0'                 // check for the absence of errors
// read digest
read HASHDIGESTA
...
read HASHDIGESTP
// acknowledge result and clear interrupts
write HASHIOBUFCTRL = 0x00000001         // acknowledge reading of the digest
write IRQCLR 0x00000001                  // clear the interrupt
write ALGSEL 0x00000000                  // disable the master control/DMA clock
// end of algorithm
```

12.5.3.2.2 New Hash Session with Digest to External Memory

The following example in pseudocode describes the actions that are typically executed by the host software. The example starts the Hash engine with a new Hash session that receives the input data through the DMA interface. In the end, the intermediate digest (nonfinal Hash operation) or the finalized Hash digest (final Hash operation) is read as a result digest through the DMA.

```
// configure master control module
write ALGSEL 0x80000008          // enable DMA path to the SHA-512 engine + Digest readout
write IRQCLR 0x80000001          // clear any outstanding events
// configure hash engine
write HASHMODE = 0x00000021      // indicate the start of a new hash session and SHA512
write HASHINLENL                 // write the length of the message (lo)
write HASHINLENH                 // write the length of the message (hi)
// if the final digest is required (pad the input DMA data), write the following register
write HASHIOBUFCTRL = 0x80       // pad the data that is transferred via DMA
// configure DMAC
write DMACH0CTL 0x000000001      // enable DMA channel 0 for message data
write DMACH0EXTADDR <ext_memory_address> // base address of the data in ext. memory
write DMACH0LEN <length>                 // input data in bytes, equal to the message length
write DMACH1CTL 0x000000001              // enable DMA channel 1 for result digest
write DMACH1EXTADDR <ext_memory_address> // base address of the digest buffer
write DMACH1LEN <length>                 // length of the result digest
// wait for completion and acknowledge the interrupt
wait IRQSTAT[0] = '1'                    // wait for operation done (hash and DMAC are ready)
check IRQSTAT[31] == '0'                 // check for the absence of errors
write IRQCLR 0x00000001                  // clear the interrupt
write ALGSEL 0x00000000                  // disable the master control/DMA clock
// the digest can now be ready from the external memory
// end of algorithm
```

12.5.3.2.3 Resumed Hash Session

The following software example in pseudocode describes the actions that are typically executed by the host software. The example starts the Hash engine with a new Hash session that receives the input data through the DMA interface. In the end, the intermediate digest (nonfinal Hash operation) or the finalized hash digest (final hash operation) is read as a result digest through the slave interface.

```
// configure master control module
write ALGSEL 0x00000008       // enable DMA path to the SHA-512 engine
write IRQCLR 0x00000001       // clear any outstanding events
// configure hash engine
write HASHMODE = 0x00000020   // indicate the start of a resumed hash session and SHA512
write HASHINLENL              // write the length of the total message (lo)
write HASHINLENH              // write the length of the total message (hi)
// write the initial digest
write HASHDIGESTA
...
write HASHDIGESTP
// if the final digest is required (pad the input DMA data), write the following register
write HASHIOBUFCTRL = 0x80               // pad the data that is transferred via DMA
// configure DMAC
write DMACH0CTL 0x000000001              // enable DMA channel 0
write DMACH0EXTADDR <ext_memory_address> // base address of the data in ext. memory
write DMACH0LEN <length>                 // input data in bytes
wait IRQSTAT[0] == '1'                   // wait for operation done (hash and DMAC are ready)
check IRQSTAT[31] == '0'                 // check for the absence of errors
// read digest
read HASHDIGESTA
...
read HASHDIGESTP
// acknowledge result and clear interrupts
write HASHIOBUFCTRL = 0x01              // acknowledge reading of the digest
write IRQCLR 0x00000001                 // clear the interrupt
write ALGSEL 0x00000000                 // disable the master control/DMA clock
// end of algorithm
```

12.5.3.3 HMAC

The Hash engine supports HMAC operations in two steps (excluding optional preprocessing), according to the algorithm defined in RFC2104.

Let:

- H(●) be a cryptographic hash function
- K be a secret key padded to the right with extra zeros to the block size of the hash function
- m be the message to be authenticated
- || denote concatenation
- ⊕ denote exclusive or (XOR)
- opad be the outer padding (0x5c5c5c...5c5c, one-block-long hexadecimal constant)
- ipad be the inner padding (0x363636...3636, one-block-long hexadecimal constant)

Then, HMAC (K,m) is mathematically defined by Equation 1.

$$HMAC(K,m) = H((K \oplus opad) \,||\, H((K \oplus ipad) \,||\, m)) \quad (1)$$

where:

- H(K ⊕ ipad) is the inner digest.
- H(K ⊕ opad) is the outer digest.
- H(K ⊕ ipad) || m is the intermediate digest.

12.5.3.3.1 Secure HMAC

The secure HMAC operation is executed using several basic hash operations with the assumption that all security sensitive parameters (hash keys, intermediate products, and message data) are stored and kept in the DMA accessible memory at the host.

The implementation of the secure HMAC operation is based on the following requirements:

- XORed keys are prepared in external memory and are read through DMA (alternatively, these can be written through the slave interface). If the hash key is longer than the hash block size (128-bytes for SHA-512 and SHA-384, and 64 bytes for SHA-256 and SHA224), the host must compress the key using the basic hash operation, which may be performed using a basic hash operation with the module.
- The input message is located in external memory and is read through DMA.
- The intermediate digest state is stored in DMA accessible memory and later read back through DA for the final hash (the state values can optionally be read and provided through the host interface).
- The result digest is read through the slave interface.

Figure 12-3 shows the steps that must be performed to implement a secure HMAC using the module.

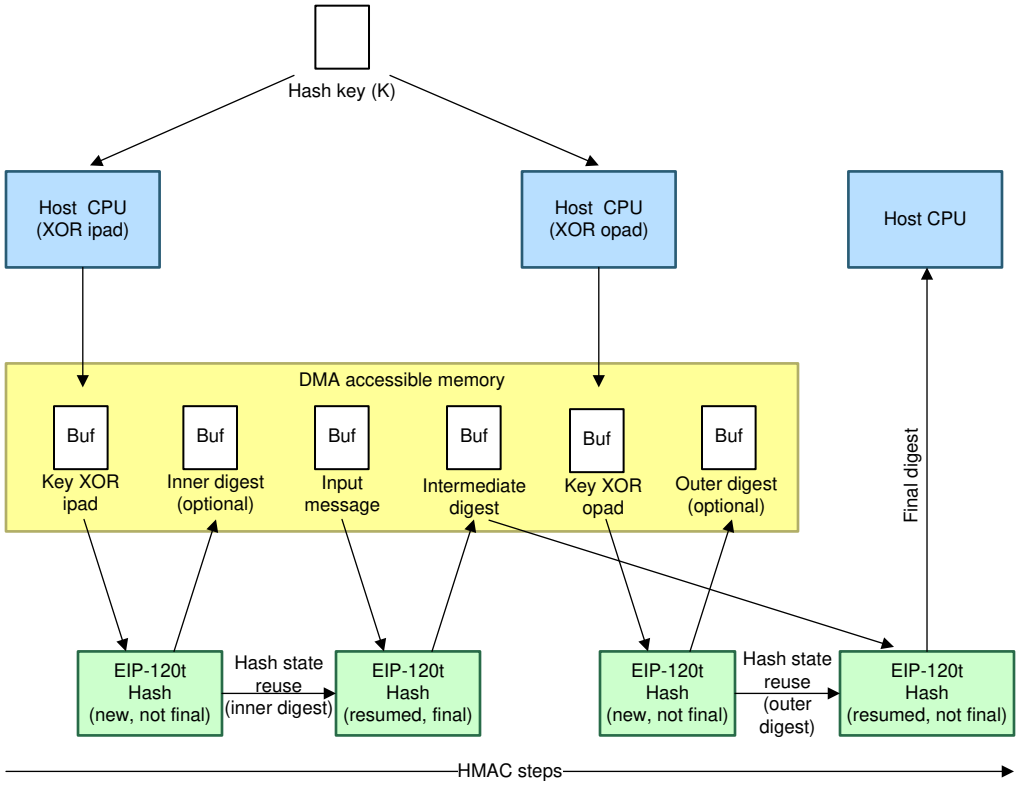


Figure 12-3. HMAC Steps

The secure HMAC uses the following basic hash operations with EIP-120t:

1. A new hash operation is established to hash the padded key (ipad), which is read through DMA and produces the inner digest. The inner digest remains in the internal state of the Hash engine and is used for the next resumed hash. The inner digest can be restored to a pre-allocated area in the external memory, and can be used for other HMAC operations that use the same key.

Note

The inner digest calculation requires a new hash operation that is not finalized because it must be resumed in the next step.

2. A resumed hash operation is established to hash the actual message. The initial digest is produced in the previous operation and is still available in the internal state. If the inner digest was prepared in external memory, the host can read this digest and program it through the slave interface. The result of this operation is stored through DMA in the pre-allocated external memory.
3. A new hash operation is established to hash the padded key (opad), which is read through DMA and produces the outer digest. The outer digest remains in the internal state of the Hash engine and can be used for the next resumed hash. The outer digest can be restored to a pre-allocated area in the external memory, and can be used for the next resumed hash. The outer digest can be restored to a pre-allocated area in the external memory such that it can be used for other HMAC operations with the same key.

Note

The outer digest calculation requires a new hash operation that is not finalized because it must be resumed in the next step.

4. A resumed hash operation is established to hash the result of Step 2 and to produce the final HMAC digest. The initial digest is produced in the previous operation and is still available in the internal state. If the outer

digest was prepared in external memory, the host can read this digest and program it through the slave interface. The final HMAC digest is read through the slave interface.

12.5.3.4 Alternative Basic Hash Where Data Originates from Slave Interface

12.5.3.4.1 New Hash Session

The following software example in pseudocode describes the actions that are typically executed by the host software. The example starts the Hash engine with a new hash session that receives the input data through the slave interface. In the end, the intermediate digest (nonfinal hash operation) or the finalized hash digest (final hash operation) is read as a result digest through the slave interface.

```
// disable the DMA path
write ALGSEL 0x00000000
// wait until the input buffer is available for writing by the host
wait HASHIOBUFCTRL[2] == '1'
write HASHMODE = 0x00000021   // indicate the start of a new hash session and SHA-512
write HASHINLENL              // the length may be written at any time during session
write HASHINLENH              // the length must be written when last data has been written
// first block: write and hand-off the block of data
write HASHDATAIN0
...
write HASHDATAIN31
write HASHIOBUFCTRL[6:0] = 0x02 // indicate that a block of data is available
loop: // intermediate blocks
      // wait until the input buffer available for writing by host
      wait HASHIOBUFCTRL[2] == '1'
      // write the intermediate block and hand off the data
      write HASHDATAIN0
      ...
      write HASHDATAIN31
      write HASHIOBUFCTRL[6:0] = 0x02 // indicate that a block of data is available if more
                                      // than one input block, go to loop
endloop
wait HASHIOBUFCTRL[2] == '1'
// write the last block and hand-off the data
// Note: if last block is misaligned, the last 32-bit word must be padded (any data is accepted)
write HASHDATAIN0
...
write HASHDATAIN31
// if an intermediate digest is required (input data is hash block size aligned)
write HASHIOBUFCTRL[6:0] = 0x42     // indicate that data is available and get the
                                    // intermediate digest (no internal padding)
// else if final digest is required (input data padded by hash engine)
write HASHIOBUFCTRL[6:0] = 0x22     // indicate that data is available and get the final
                                    // digest of the padded data
// wait until output data ready
wait AESCTL[0] == '1'
// read digest
read HASHDIGESTA
...
read HASHDIGESTP
write HASHIOBUFCTRL = 0x01 // acknowledge that the digest is read
// end of algorithm
```

12.5.3.4.2 Resumed Hash Session

The following software example in pseudocode describes the actions that are typically executed by the host software. The example starts the Hash engine with a new hash session that receives the input data through the slave interface. In the end, the intermediate digest (nonfinal hash operation) or the finalized hash digest (final hash operation) is read as a result digest through the slave interface.

```
// disable the DMA path
write ALGSEL 0x00000000
// wait until the input buffer is available for writing by the Host
wait HASHIOBUFCTRL[2] == '1'
write HASHMODE = 0x00000020 // indicate the start of the resumed hash session and SHA256
// write initial digest
write HASHDIGESTA
...
write HASHDIGESTP
write HASHINLENL // the length may be written at any time during session
write HASHINLENH // the length must be written when last data has been written
// first block: write and hand-off the block of data
write HASHDATAIN0
...
write HASHDATAIN31
write HASHIOBUFCTRL[6:0] = 0x02     // indicate that a block of data is available
loop: // intermediate blocks // wait until the input buffer available for writing by Host
    wait HASHIOBUFCTRL[2] == '1'
    // write the intermediate block and hand off the data
    write HASHDATAIN0
    ...
    write HASHDATAIN31
    write HASHIOBUFCTRL[6:0] = 0x02 // indicate that a block of data is available if more
                                    // than one input block, go to loop
endloop
wait HASHIOBUFCTRL[2] == '1'
// write the last block and hand-off the data
// Note: if last block is misaligned, the last 32-bit word must be padded (any data is accepted)
write HASHDATAIN0
...
write HASHDATAIN31
// if an intermediate digest is required (input data is hash block size aligned)
write HASHIOBUFCTRL[6:0] = 0x42    // indicate that data is available and
                                   // get the intermediate digest
// else if final digest is required (input data padded by hash engine)
write HASHIOBUFCTRL[6:0] = 0x22    // indicate that data is available and
                                   // get the final digest of the padded data
// wait until output data ready
wait HASHIOBUFCTRL[2] == '1'
// read digest
read HASHDIGESTA
...
read HASHDIGESTP
write HASHIOBUFCTRL = 0x01         // acknowledge reading of the digest
// end of algorithm
```

12.5.4 Encryption and Decryption

The crypto engine (AES) transfers data over the following interfaces:

- AES accepts input data from two sources: AHB slave interface and DMA. Within one operation, it is possible to combine data from these two sources: write data from the slave interface, and write data from the DMA to complete the operation.
- Input IV and length must be supplied using the AHB slave interface. The output IV can be read only using the slave interface.
- Result data must be read using the same interface as the input data: using either the slave interface or DMA.
- The result tag for operations with authentication can be read using the slave interface or DMA.

12.5.4.1 Data Format and Byte Order

The following examples show how the data must be submitted to the AES engine. Because the AHB slave interface is mostly transparent for data (no data modifications), the alignment for the register interface is identical, as described in the following examples.

Note

The byte in bold type is a first byte of the key or message.

NIST SP 800-38a, AES-ECB 256-bit Encrypt:

```
AES Key In:      603deb10 15ca71be 2b73aef0 857d7781
                 1f352c07 3b6108d7 2d9810a3 0914dff4
AES Data In:     6bc1bee2 2e409f96 e93d7e11 7393172a
AES Data Out:    f3eed1bd b5d2a03c 064b5a7e 3db181f
```

AES Key in external RAM, loaded through the key-store module:

```
Word_0[31:0]: 10eb3d60
Word_1[31:0]: be71ca15
Word_2[31:0]: f0ae732b
Word_3[31:0]: 81777d85
Word_4[31:0]: 072c351f
Word_5[31:0]: d708613b
Word_6[31:0]: a310982d
Word_7[31:0]: f4df1409
```

Input data using slave interface or DMA:

```
AESDATAIN0[31:0]: e2bec16b
AESDATAIN1[31:0]: 969f402e
AESDATAIN2[31:0]: 117e3de9
AESDATAIN3[31:0]: 2a179373
```

Output data using slave interface or DMA:

```
AESDATAOUT0[31:0]: bdd1eef3
AESDATAOUT1[31:0]: 3ca0d2b5
AESDATAOUT2[31:0]: 7e5a4b06
AESDATAOUT3[31:0]: f881b13d
```

12.5.4.2 Key Store

Before any encryption or decryption operation starts, the key-store module must have at least one key loaded and available for crypto operations. Keys can be loaded only from external memory using a DMA operation. DMAC channel 0 (inbound) is used for this purpose.

12.5.4.2.1 Load Keys from External Memory

The following software example in pseudocode describes the actions that are typically executed by the host software to load one or more keys into the key-store module.

```
// configure master control module
write ALGSEL 0x00000001            // enable DMA path to the key store module
write IRQCLR 0x00000001            // clear any outstanding events
// configure key store module (area, size)
write KEYSIZE 0x00000001           // 128-bit key size
write KEYWRITEAREA 0x00000001      // enable keys to write (e.g. Key 0)
// configure DMAC
write DMACH0CTL 0x000000001              // enable DMA channel 0
write DMACH0EXTADDR <ext_memory_address> // base address of the key in ext. memory
write DMACH0LEN <length>                 // total key length in bytes
                                         // (e.g. 16 for 1 x 128-bit key)
// wait for completion
wait IRQSTAT[0] == '1'                   // wait for operation completed
check IRQSTAT[31:30] == '00'             // check for absence of errors in DMA and key store
write IRQCLR 0x00000001                  // acknowledge the interrupt
write ALGSEL 0x00000000                  // disable master control/DMA clock
// check status
check KEYWRITTENAREA 0x000000001         // check that Key 0 was written
// end of algorithm
```

12.5.4.3 Basic AES Modes

12.5.4.3.1 AES-ECB

For AES-ECB operations, the following configuration parameters are required:

- Key from the key-store module
- Control register settings (mode, direction, key size)
- Length of the data

The length field can have any value. If a data stream is finished and the next data stream uses the same key and control, only the length field must be written with a new value. The length field may also be 0, for continued processing.

12.5.4.3.2 AES-CBC

For AES-CBC operations, the following configuration parameters are required:

- Key from the key-store module
- IV from the slave interface
- Control register settings (mode, direction, key size)
- Length of the data

The length field can have any value. If a data stream is finished and the next data stream uses the same key and control, it is allowed to write only the IV and length field with a new value. The length field may also be 0, for continued processing.

If the result IV must be read by the host, the CRYPTO:AESCTL.SAVE_CONTEXT bit must be set to 1 after processing the programmed number of bytes.

12.5.4.3.3 AES-CTR

For AES-CTR operations, the following configuration parameters are required:

- Key from the key-store module
- IV from the slave interface, including initial counter value (usually 0x00000001)
- Control register settings (mode, direction, key size)
- Length of the data (may be nonblock-size aligned)

The length field can have any value. If a data stream is finished and the next data stream uses the same key and control, only the IV and length field can be written with a new value. The length field can be 0, resulting in continued processing.

If the result IV must be read by the host, the save_context bit must be set to 1 after processing the programmed number of bytes.

12.5.4.3.4 Programming Sequence with DMA Data

The following software example in pseudocode describes the actions that are typically executed by the host software to encrypt (using a basic AES mode) a message, stored in external memory, and place an encrypted result into a pre-allocated area in the external memory.

```
// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key
                             // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'      // check that the key is loaded without errors
// Write the IV for non-ECB modes
// The IV must be written with the same conventions as the data
// if ((not ECB mode) and (not IV reuse))
// then: write the initialization vector when a new IV is required
write AESIV_0
...
write AESIV_3
// endif
// configure AES engine
write AESCTL = 0b0010_0000_0000_0000_0000_0000_0010_1100 // Program AES-CBC-128 encryption
                                                         // and save IV
write AESDATALEN0            // write length of the message (lo)
write AESDATALEN1            // write length of the message (hi)
write DMACH0CTL 0x000000001  // enable DMA channel 0
// configure DMAC
write DMACH0EXTADDR <address> // base address of the input data in ext.memory
write DMACH0LEN <length>      // input data length in bytes, equal to the
                              // message length (may be non-block size aligned)
write DMACH1CTL 0x000000001  // enable DMA channel 1
write DMACH1EXTADDR <address> // base address of the output data buffer
write DMACH1LEN <length>      // output data length in bytes, equal to the
                              // result data length (may be non-blocksize aligned)
// wait for completion
wait IRQSTAT[0] == '1'        // wait for operation completed
check IRQSTAT[31] == '0'      // check for absence of errors
write AESALGSEL 0x00000000    // disable master control/DMA clock
// if (not ECB mode)
// then: only if the IV needs to be re-used/read
wait AESCTL[30] == '1'        // wait for SAVED_CONTEXT_RDY bit [30]
read AESIV_0
...
read AESIV_3                  // this read clears the SAVED_CONTEXT_RDY flag
// endif
// end of algorithm
```

12.5.4.4 CBC-MAC

For CBC-MAC operations, the following configuration parameters are required:

- Key from the key-store module
- IV must be written with zeroes
- Control register settings (mode, direction, key size)
- Length of the authenticated data (may be nonblock-size aligned)

The input data can end misaligned for CBC-MAC operations. If this is the case, the crypto core internally pads the last input data block.

The length field can have any value. If a data stream is finished and the next data stream uses the same key and control, writing only a part of the next context is not allowed. A new data stream must always write the complete context. The length field must never be written with zeroes.

For CMAC operations, AESKEY2_n and AESKEY3_n must be written.

For all CBC-MAC operations, the data may end misaligned. If this is the case, the last data block must be padded as follows:

- For regular CBC-MAC operations, the host must pad the input data to a 128-bit boundary with zeroes. More formally, the padding must satisfy the bit string: 0n, with 0 ≤ n ≤127, since the EIP-120t only supports bytes, (n MOD 8) = 0.
- For CMAC operations, the host must pad the input data to a 128-bit boundary with a string that satisfies 1·0n , with 0 ≤ n <127. Because the EIP-39 only supports bytes, n must be such that (n MOD 8) = 7.

For regular CBC-MAC operations, the CRYPTO:AESDATALENn register can have any value but for CMAC operations, the actual length of the entire message or of the last data block, must be programmed. This is because the message length MOD 16 is used for selecting the key to use for the last round. The AESDATALENn register may never be written with zeroes. To support the degenerate case of CMAC on the empty message, program the length to 1 instead. The data block to be loaded consists of only padding then. If a message MAC is complete and the next message uses the same key and control, writing only a part of the next context is not allowed; a new message operation must always write the complete context.

12.5.4.4.1 Programming Sequence for Regular CBC-MAC

The following software example in pseudocode describes the actions that are typically executed by the host software to authenticate a message, stored in external memory, with AES-CBC-MAC mode. The result TAG is read using the slave interface.

The following sequence processes a packet of at least 1 input data byte.

```
// configure the master control module
write ALGSEL 0x00000002       // enable the DMA path to the AES engine
write IRQCLR 0x00000001       // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000  // load the key from ram area 0 (NOTE: The key
                              // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'   // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'       // check that the key is loaded without errors
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0000_1000_0000_0100_1100 // program AES-CBC-MAC-128 authentication
write AESDATALEN0             // write length of the crypto block (lo)
write AESDATALEN1             // write the length of the crypto block (hi)
                              // (may be non-block size aligned)
write DMACH0CTL 0x000000001   // enable DMA channel 0
// configure DMAC
write DMACH0EXTADDR <address> // base address of the input data in ext. memory
write DMACH0LEN <length>      // input data length in bytes, equal to the message
                              // length len({aad data, pad, crypto_data, pad})
                              // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'        // wait for operation completed
check IRQSTAT[31] == '0'      // check for the absence of errors
write ALGSEL 0x00000000       // disable master control/DMA clock
// read tag
wait AESCTL[30] == '1'        // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3              // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.4.2 Programming Sequence for Regular CBC-MAC with Continuation

The following software example in pseudo code, describes the actions that are typically executed by the Host software to authenticate a message, stored in external memory, with AES-CBC-MAC mode. The result TAG is read via the slave interface. The packet is divided into two parts: part 1 and part 2. The length of part 1 must be multiples of the block size.

```
// configure the master control module
write ALGSEL 0x00000002       // enable the DMA path to the AES engine
write IRQCLR 0x00000001       // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000  // load the key from ram area 0 (NOTE: The key
                              // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'   // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'       // check that the key is loaded without errors
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0000_1000_0000_0000_1100 // program AES-CBC-MAC-128 authentication
write AESDATALEN0              // write length of the crypto block part 1 (lo)
write AESDATALEN1              // write the length of the crypto block part 1 (hi)
                               // (must be block size aligned)
// configure DMAC
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the input data in ext. memory
write DMACH0DMALENGTH <length> // input data length in bytes, equal to the
                               // message part 1 length (must be block size aligned)
// wait for completion
```

```
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable master control/DMA clock
// read intermediate tag
wait AESCTL[30] == '1'          // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                // this read clears the SAVED_CONTEXT_RDY flag
// configure the master control module
write ALGSEL 0x00000002         // enable the DMA path to the AES engine
write IRQCLR 0x00000001         // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000    // load the key from ram area 0 (NOTE: The key
                                // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'     // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'         // check that the key is loaded without errors
// write the intermediate tag to IV
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0000_1000_0000_0000_1100 // program AES-CBC-MAC-
                                                         // 128 authentication
write AESDATALEN0               // write length of the crypto block part 2 (lo)
write AESDATALEN1               // write the length of the crypto block part 2 (hi)
                                // (may be non-block size aligned)
// configure DMAC
write DMACH0CTRL 0x000000001    // enable DMA channel 0
write DMACH0EXTADDR <address>   // base address of the input data in ext. memory
write DMACH0DMALENGTH <length>  // input data length in bytes, equal to
                                // the message part 2 length (may be
                                // non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable master control/DMA clock
// read final tag
wait AESCTL[30] == '1'          // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.4.3 Programming Sequence for CMAC CBC-MAC

The following software example in pseudo code, describes the actions that are typically executed by the Host software to authenticate a message, stored in external memory, with AES-CMAC-CBC-MAC mode. The result TAG is read via the slave interface.

The following sequence processes a packet of at least one input data byte.

```
// configure the master control module
write ALGSEL 0x00000002        // enable the DMA path to the AES engine
write IRQCLR 0x00000001        // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000   // load the key from ram area 0 (NOTE: The key
                               // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'    // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'       // check that the key is loaded without errors
// write the KEY2
write AESKEY2_0
...
write AESKEY2_3
// write the KEY3
write AESKEY3_0
...
write AESKEY3_3
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0010_0000_0000_0000_0000_0000_1100 // program AES-XCBC-
                                                         // MAC-128 authentication
```

```
write AESDATALEN0              // write length of the crypto block (lo)
write AESDATALEN1              // write the length of the crypto block (hi)
                               // (may be non-block size aligned)
// configure DMAC
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the input data in ext. memory
write DMACH0DMALENGTH <length> // input data length in bytes, equal to the
                               // message length
                               // len({AAD data, pad, crypto_data, pad})
                               // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'         // wait for operation completed
check IRQSTAT[31] == '0'       // check for the absence of errors
write ALGSEL 0x00000000        // disable master control/DMA clock
// read tag
wait AESCTL[30] == '1'         // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3               // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.4.4 Programming Sequence for CMAC CBC-MAC with Continuation

The following software example in pseudo code, describes the actions that are typically executed by the Host software to authenticate a message, stored in external memory, with AES-CMAC-CBC-MAC mode. The result TAG is read via the slave interface. The packet is divided into two parts: part 1 and part 2. For part 1 the length must be multiples of the block size and AESKEY2_x registers must be written with 0x00000000.

```
// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0
                             // (NOTE: The key must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'     // check that the key is loaded without errors
// write zero to the KEY2
write AESKEY2_0 0x00000000
...
write AESKEY2_3 0x00000000
// write the KEY3
write AESKEY3_0
...
write AESKEY3_3
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0010_0000_0000_0000_0000_0000_1100 // program AES-XCBC-
                                                         // MAC-128 authentication
write AESDATALEN0              // write length of the crypto block part 1 (lo)
write AESDATALEN1              // write the length of the crypto block part 1 (hi)
                               // (must be block size aligned)
// configure DMAC
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the input data in ext. memory
write DMACH0DMALENGTH <length> // input data length in bytes, equal to the message part 1 length
                               // (must be block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'         // wait for operation completed
check IRQSTAT[31] == '0'       // check for the absence of errors
write ALGSEL 0x00000000        // disable master control/DMA clock
// read intermediate tag
wait AESCTL[30] == '1'         // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3               // this read clears the SAVED_CONTEXT_RDY flag
// configure the master control module
write ALGSEL 0x00000002        // enable the DMA path to the AES engine
write IRQCLR 0x00000001        // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000   // load the key from ram area 0
                               // (NOTE: The key must be pre-loaded to this area)
```

```
wait KEYREADAREA[31] == '0'     // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'        // check that the key is loaded without errors
// write the KEY2
write AESKEY2_0
...
write AESKEY2_3
// write the KEY3
write AESKEY3_0
...
write AESKEY3_3
// write intermediate tag to the IV
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0010_0000_0000_0000_0000_0000_1100 // program AES-XCBC-
                                                         // MAC-128 authentication
write AESDATALEN0               // write length of the crypto block part 2 (lo)
write AESDATALEN1               // write the length of the crypto block part 2 (hi)
                                // (may be non-block size aligned)
// configure DMAC
write DMACH0CTRL 0x000000001    // enable DMA channel 0
write DMACH0EXTADDR <address>   // base address of the input data in ext. memory
write DMACH0DMALENGTH <length> // input data length in bytes, equal to the message part 2 length
                                // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable master control/DMA clock
// read tag
wait AESCTL[30] == '1'          // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.5 AES-CCM

For AES-CCM operations, the following configuration parameters are required:

- Key from the key-store module
- The IV must be written with the flags for the cryptographic operation and the NONCE bytes, for both authentication and encryption (see Section 12.2.7.2)
- Control register settings (mode, direction, key size)
- Length of the crypto data (may be nonblock size aligned)
- Length of the AAD data; must be less than $2^{16} – 2^8$ bytes (may be nonblock size-aligned)

CRYPTO:AESCTL.CCM_L must be 001_b, 011_b, or 111_b, representing a crypto data length field of 2, 4, or 8 bytes, respectively.

CRYPTO:AESCTL.CCM_M can be set to any value and has no effect on the processing. The host must select the valid TAG bytes from the 128-bit TAG.

The AAD and cryptographic data may end misaligned. In this case, the crypto core pads both data types to a 128-bit boundary with zeroes. Padding is done as follows: the AAD and crypto data padding satisfy the bit string, 0n, with 0 ≤ n ≤127, such that the input data block length, including padding, is 128-bit aligned. The AAD data must be transferred to the AES engine with a separate DMA operation (it may not be combined with the payload data) or using slave transfers.

The context length field can have any value. If a data stream is done and the next data stream uses the same key and control, only the IV and length fields can be written with a new value. The user cannot write both length fields with zeroes.

The result TAG is typically read using the slave interface, but can also be written to an external memory location using a separate DMA operation.

12.5.4.5.1 Continued CCM Processing

A CCM operation can be interrupted during the AAD and crypto data processing phase at 128-bit aligned boundaries and the Host can store the intermediate processing state of the engine. The data to be stored for this consists of the intermediate IV, the TAG value, the internal block counter and for the situations where the AAD phase is interrupted also the AES-CCM alignment information need to be stored. The keys and mode settings are also part of the state, but these are not changed during operation.

To request the interruption of a CCM operation the ‘get_digest’ bit has to be set in the CRYPTO:AESCTL register, after writing the last data word of a 128-bit aligned crypto data part.

To continue a CCM operation that was previously interrupted, the intermediate processing state has to be restored:

- The intermediate IV must be written to the AESIVn registers.
- The intermediate TAG must be written to the AESKEY3_n registers (TAG accumulation is internally done in the AESKEY3_n registers).
- The intermediate block counter value must be written to the AESBLKCNTn registers.
- For AAD continuation, the AES-CCM alignment information must be written to the CRYPTO:AESCCMALNWRD register.

The static part of the context, such as the AES keys, length registers and mode for the operation must also be loaded with their original values to ensure correct continuation. When writing the mode, make sure that the CRYPTO:AESCTL.GCM_CCM_CONTINUE bit and, depending on the data phase to continue from, the CRYPTO:AESCTLG.CM_CCM_CONTINUE_AAD bit are set to notify the engine it has to continue processing from the loaded state.

12.5.4.5.2 Programming Sequence for AES-CCM

The following software example in pseudocode describes the actions that are typically executed by the host software to encrypt and authenticate a message (AAD and payload data), stored in external memory, with AES-CCM mode. The encrypted result is placed into a pre-allocated area in external memory. The result TAG is read using the slave interface.

The following sequence processes a packet of at least 1 byte of AAD data and at least 1 crypto data byte.

```
// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key
                             // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'      // check that the key is loaded without errors
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
write AESDATALEN0 // write the length of the crypto block (lo)
write AESDATALEN1 // write the length of the crypto block (hi)
                  // (may be non-block size aligned)
write AESAUTHLEN  // write the length of the AAD data block (may be non-block size aligned)
// configure DMAC to fetch the AAD data
write DMACH0CTL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address> // base address of the AAD input data in ext. memory
write DMACH0LEN <length>      // AAD data length in bytes, equal to the AAD length len({aad data})
                              // (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'        // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'      // check for the absence of errors
// configure DMAC
write DMACH0CTL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address> // base address of the payload data in ext. memory
write DMACH0LEN <length>      // payload data length in bytes, equal to the
                              // message length len({crypto_data})
write DMACH1CTL 0x000000001   // enable DMA channel 1
write DMACH1EXTADDR <address> // base address of the output data buffer
write DMACH1LEN <length>      // output data length in bytes, equal to
                              // the result data length len({crypto data})
// wait for completion
wait IRQSTAT[0] == '1'        // wait for operation completed
check IRQSTAT[31] == '0'      // check for the absence of errors
write ALGSEL 0x00000000       // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'        // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3              // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.5.3 Programming Sequence for Continued AES-CCM in the AAD Phase

Three main phases can be distinguished:

- Initialization of the engine and start processing the first part of the AAD data.
- Prepare the engine for interruption and retrieve the intermediate processing state.
- Restore the intermediate processing state and start processing the next or last AAD data blocks.

```
-- initialization and process until interruption --

// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key
                             // must be pre-loaded to this area)
```

```
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'      // check that the key is loaded without errors
// write zeros to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write initial IV
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
write AESDATALEN0     // write the length of the crypto block (lo)
write AESDATALEN1     // write the length of the crypto block (hi)
                      // (may be non-block size aligned)
write AESAUTHLEN      // write the length of the AAD data block
                      // (may be non-block size aligned)
// configure DMAC to fetch the first part of AAD data
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length> // AAD data length in bytes, equal to the
                               // AAD length len({AAD data}) (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'         // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'       // check for the absence of errors

-- interrupt processing and store the engine state in the AAD phase --

// configure the AES engine with get_digest
write AESCTL = 0b0010_1000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
// read intermediate tag
wait AESCTL[30] == '1' // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3
// read intermediate block counter
read AESBLKCNT0
read AESBLKCNT1
// read intermediate IV
read AESIV_0
...
read AESIV_0
// read intermediate CCM align data word
read AESCCMALNWRD

-- restore and continue an interrupted operation in the AAD phase --

// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000  // load the key from ram area 0 (NOTE: The key
                              // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'   // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'      // check that the key is loaded without errors
// write intermediate tag to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write intermediate IV
write AESIV_0
...
write AESIV_3
// configure the AES engine with gcm_ccm_continue_aad
write AESCTL = 0b0010_0100_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
write AESDATALEN0              // write the length of the crypto block (lo)
write AESDATALEN1              // write the length of the crypto block (hi)
                               // (may be non-block size aligned)
// write intermediate block counter
write AESBLKCNT0
write AESBLKCNT1
// write intermediate CCM align data word
write AESCCMALNWRD
write AESAUTHLEN               // write the length of the AAD data block
                               // (may be non-block size aligned)
// configure DMAC to fetch the second part of AAD data
```

```
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length> // AAD data length in bytes, equal to the
                               // AAD length len({AAD data})
                               // (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'          // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'        // check for the absence of errors
// configure DMAC to process the payload data
write DMACH0CTRL 0x000000001    // enable DMA channel 0
write DMACH0EXTADDR <address>   // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length>  // payload data length in bytes, equal
                               // to the payload length
                               // len({crypto_data}) (may be non-block size aligned)
write DMACH1CTL 0x000000001     // enable DMA channel 1
write DMACH1EXTADDR <address>   // base address of the output data buffer
write DMACH1LEN <length>        // output data length in bytes, equal to
                               // the result data length len({crypto data})
                               // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'          // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.5.4 Programming Sequence for Continued AES-CCM in the Payload Phase

Three main phases can be distinguished:

- Initialization of the engine and start processing all or the remaining AAD data and the first crypto data blocks.
- Prepare the engine for interruption and retrieve the intermediate processing state.
- Restore the intermediate processing state and start processing the next or last crypto data blocks.

```
-- initialization and process until interruption --

// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key must
                             // be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] = '0'      // check that the key is loaded without errors
// write zeros to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write initial IV
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
write AESDATALEN0              // write the length of the crypto block (lo)
write AESDATALEN1              // write the length of the crypto block (hi)
                               // (may be non-block size aligned)
write AESAUTHLEN               // write the length of the AAD data block
                               // (may be non-block size aligned)
// configure DMAC to fetch the AAD data
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length> // AAD data length in bytes, equal to the
                               // AAD length len({AAD data})
                               // (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'         // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'       // check for the absence of errors
// configure DMAC to process the first part of payload data
```

```
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length> // payload data length in bytes, equal to
                               // the payload length len({crypto_data})
                               // (may be non-block size aligned)
write DMACH1CTL 0x000000001    // enable DMA channel 1
write DMACH1EXTADDR <address>  // base address of the output data buffer
write DMACH1LEN <length>       // output data length in bytes, equal to the
                               // result data length len({crypto data})
                               // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable the master control/DMA clock

-- interrupt processing and store the engine state in the payload phase --

// configure the AES engine with get_digest
write AESCTL = 0b0010_1000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
// read intermediate tag
wait AESCTL[30] == '1'         // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3
// read intermediate block counter
read AESBLKCNT0
read AESBLKCNT1
// read intermediate IV
read AESIV_0
...
read AESIV_3

-- restore and continue an interrupted operation in the AAD phase --

// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000    // load the key from ram area 0 (NOTE: The key
                                // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'     // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'        // check that the key is loaded without errors
// write intermediate tag to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write intermediate IV
write AESIV_0
...
write AESIV_3
// configure the AES engine with gcm_ccm_continue
write AESCTL = 0b0011_0000_0101_1100_0000_0000_0100_1100 // program AES-CCM-128
                                                         // encryption (M=1, L=3)
write AESDATALEN0 // write the length of the crypto block (lo)
write AESDATALEN1 // write the length of the crypto block (hi) (may be non-block size aligned)
// write intermediate block counter
write AESBLKCNT0
write AESBLKCNT1
write AESAUTHLEN               // write the length of the AAD data block
                               // (may be non-block size aligned)
// configure DMAC to process the second part of payload data
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length> // payload data length in bytes, equal to the
                               // payload length len({crypto_data}) (may be non-block size aligned)
write DMACH1CTL 0x000000001    // enable DMA channel 1
write DMACH1EXTADDR <address>  // base address of the output data buffer
write DMACH1LEN <length>       // output data length in bytes, equal to
                               // the result data length len({crypto data})
                               // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'         // wait for operation completed
check IRQSTAT[31] == '0'       // check for the absence of errors
write ALGSEL 0x00000000        // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'         // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
```

```
read AESTAGOUT_3                // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.6 AES-GCM

For AES-GCM operations, the following configuration parameters are required:

- Key from the key-store module
- IV from the slave interface, including an initial counter value of 1
- Control register settings (mode, direction, key size)
- Length of the cipher data (may be nonblock-size aligned)
- Length of the AAD data (may be nonblock-size aligned)

The AES module is used in the mode where it calculates the Y0 encrypted and H (hash key) internally based on cipher key from the key-store module.

The AAD and cryptographic data may end misaligned. In this case, the module internally pads both sets of data to a 128-bit boundary with zeroes. Padding is done as follows: the AAD and crypto data padding satisfies the bit string: 0n, with $0 \leq n \leq 127$ such that the input AAD and data block lengths including padding are 128-bit aligned. The AAD data must be transferred to the AES engine with a separate DMA operation (it may not be combined with the payload data) or using slave transfers.

The length field can have any value. If a data stream is done and the next data stream uses the same key and control, only the IV and length fields can be written with a new value. It is not allowed to write both length fields with zeroes. A GCM operation cannot be interrupted. The result TAG is typically read through the slave interface, but can also be written to an external memory location through a separate DMA operation.

12.5.4.6.1 Continued AES-GCM Processing

A GCM operation can be interrupted during the AAD and crypto data processing phase at 128-bit aligned boundaries and the Host can store the intermediate processing state of the engine. The data to be stored for this consists of the intermediate IV, the TAG value and the internal block counter. The keys and mode settings are also part of the state, but these are not changed during operation.

To request the interruption of a GCM operation the CRYPTO:AESCTLG.ET_DIGEST bit has to be set, after writing the last data word of a 128-bit aligned crypto data part.

To continue a GCM operation that was previously interrupted, the intermediate processing state has to be restored:

- The intermediate IV must be written to the AESIVn registers.
- The intermediate TAG must be written to the AESKEY3_n registers (TAG accumulation is internally done in the AESKEY3_n registers).
- The intermediate block counter value must be written to the CRYPTO:AESBLKCNTn registers.

The static part of the context, such as the AES keys, GHASH keys, length registers and mode for the operation must also be loaded with their original values to ensure correct continuation. When writing the mode, make sure that the GCM_CCM_CONTINUE bit and, depending on the data phase to continue from, the GCM_CCM_CONTINUE_AAD bit are set in the CRYPTO:AESCTL register to notify the engine it has to continue processing from the loaded state.

12.5.4.6.2 Programming Sequence for AES-GCM

The following software example in pseudocode describes the actions that are typically executed by the host software to encrypt and authenticate a message using AES-GCM mode. The message (AAD and payload data) is fetched from external memory and the encrypted result is placed in a pre-allocated area in the external memory.

The result TAG is read through the slave interface. The following sequence processes a packet of at least 1byte of AAD data and at least 1 crypto data byte.

```
// configure the master control module
write ALGSEL 0x00000002       // enable the DMA path to the AES engine
write IRQCLR 0x00000001       // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000  // load the key from ram area 0 (NOTE:
                              // The key must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'   // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'      // check that the key is loaded without errors
// write the initialization vector
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0011_0000_0000_0100_1100 // Program the AES-GCM-128
                                                         //encryption (autonomous)
write AESDATALEN0             // write the length of the crypto block (lo)
write AESDATALEN1             // write the length of the crypto block (hi)
                              // (may be non-block size aligned)
write AESAUTHLEN              // write the length of the AAD data block
                              // (may be non-block size aligned)
// configure DMAC to fetch the AAD data
write DMACH0CTL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address> // base address of the AAD data in ext. memory
write DMACH0LEN <length>      // AAD data length in bytes, equal to the aad
                              // length len({aad data}) (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'        // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'      // check for the absence of errors
// configure DMAC to process the payload data
write DMACH0CTL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address> // base address of the payload data in ext. memory
write DMACH0LEN <length>      // payload data length in bytes, equal to the
                              // payload length len({crypto_data}) (may be non-block size aligned)
write DMACH1CTL 0x000000001   // enable DMA channel 1
write DMACH1EXTADDR <address> // base address of the output data buffer
write DMACH1LEN <length>      // output data length in bytes, equal to the
                              // result data length len({crypto data})
                              // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'        // wait for operation completed
check IRQSTAT[31] == '0'      // check for the absence of errors
write ALGSEL 0x00000000       // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'        // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3              // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.6.3 Programming Sequence for Continued AES-GCM in the AAD Phase

Three main phases can be distinguished:

- Initialization of the engine and start processing the first part of the AAD data.
- Prepare the engine for interruption and retrieve the intermediate processing state.
- Restore the intermediate processing state and start processing the next or last AAD data blocks.

```
-- initialization and process until interruption --

// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key must
                             // be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'     // check that the key is loaded without errors
// write zeros to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write initial IV
write AESIV_0
```

```
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
write AESDATALEN0              // write the length of the crypto block (lo)
write AESDATALEN1              // write the length of the crypto block (hi) (may
                               // be non-block size aligned)
write AESAUTHLEN               // write the length of the AAD data block (may be
                               // non-block size aligned)
// configure DMAC to fetch the first part of AAD data
write DMACH0CTRL 0x000000001   // enable DMA channel 0
write DMACH0EXTADDR <address>  // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length> // AAD data length in bytes, equal to the AAD
                               // length len({AAD data}) (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'         // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'       // check for the absence of errors

-- interrupt processing and store the engine state in the AAD phase --

// configure the AES engine with get_digest
write AESCTL = 0b0010_1000_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
// read intermediate tag
wait AESCTL[30] == '1'         // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3
// read intermediate block counter
read AESBLKCNT0
read AESBLKCNT1

-- restore and continue an interrupted operation in the AAD phase --

// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000  // load the key from ram area 0 (NOTE: The key
                              // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'   // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'      // check that the key is loaded without errors
// write intermediate tag to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write initial IV
write AESIV_0
...
write AESIV_3
// configure the AES engine with gcm_ccm_continue_aad
write AESCTL = 0b0010_0100_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
write AESDATALEN0              // write the length of the crypto block (lo)
write AESDATALEN1              // write the length of the crypto block (hi) (may
                               // be non-block size aligned)
// write intermediate block counter
write AESBLKCNT0
write AESBLKCNT1
write AESAUTHLEN                  // write the length of the AAD data block (may be
                                  // non-block size aligned)
// configure DMAC to fetch the second part of AAD data
write DMACH0CTRL 0x000000001      // enable DMA channel 0
write DMACH0EXTADDR <address>     // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length>    // AAD data length in bytes, equal to the AAD
                                  // length len({AAD data}) (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'            // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'          // check for the absence of errors
// configure DMAC to process the payload data
write DMACH0CTRL 0x000000001      // enable DMA channel 0
write DMACH0EXTADDR <address>     // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length>    // payload data length in bytes, equal to the
                                  // payload length len({crypto_data})
                                  // (may be non-block size aligned)
write DMACH1CTL 0x000000001       // enable DMA channel 1
write DMACH1EXTADDR <address>     // base address of the output data buffer
write DMACH1LEN <length>          // output data length in bytes, equal to the
```

```
                                    // result data length len({crypto data})
                                    // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'              // wait for operation completed
check IRQSTAT[31] == '0'            // check for the absence of errors
write ALGSEL 0x00000000             // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'              // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                    // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.4.6.4 Programming Sequence for Continued AES-GCM in the Payload Phase

Three main phases can be distinguished:

- Initialization of the engine and start processing all or the remaining AAD data and the first crypto data blocks.
- Prepare the engine for interruption and retrieve the intermediate processing state.
- Restore the intermediate processing state and start processing the next or last crypto data blocks.

```
-- initialization and process until interruption --

// configure the master control module
write ALGSEL 0x00000002      // enable the DMA path to the AES engine
write IRQCLR 0x00000001      // clear any outstanding events
// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key must
                             // be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'     // check that the key is loaded without errors
// write zeros to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write initial IV
write AESIV_0
...
write AESIV_3
// configure the AES engine
write AESCTL = 0b0010_0000_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
write AESDATALEN0     // write the length of the crypto block (lo)
write AESDATALEN1     // write the length of the crypto block (hi) (may be non-block size aligned)
write AESAUTHLEN      // write the length of the AAD data block (may be non-block size aligned)
// configure DMAC to fetch the AAD data
write DMACH0CTRL 0x000000001    // enable DMA channel 0
write DMACH0EXTADDR <address>   // base address of the AAD data in ext. memory
write DMACH0DMALENGTH <length>  // AAD data length in bytes, equal to the AAD
                                // length len({AAD data}) (may be non-block size aligned)
// wait for completion of the AAD data transfer
wait IRQSTAT[1] == '1'          // wait for DMA_IN_DONE
check IRQSTAT[31] == '0'        // check for the absence of errors
// configure DMAC to process the first part of payload data
write DMACH0CTRL 0x000000001    // enable DMA channel 0
write DMACH0EXTADDR <address>   // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length>  // payload data length in bytes, equal to the
                                // payload length len({crypto_data})
                                // (may be non-block size aligned)
write DMACH1CTL 0x000000001     // enable DMA channel 1
write DMACH1EXTADDR <address>   // base address of the output data buffer
write DMACH1LEN <length>        // output data length in bytes, equal to the
                                // result data length len({crypto data})
                                // (may be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'          // wait for operation completed
check IRQSTAT[31] == '0'        // check for the absence of errors
write ALGSEL 0x00000000         // disable the master control/DMA clock

-- interrupt processing and store the engine state in the payload phase --

// configure the AES engine with get_digest
write AESCTL = 0b0010_1000_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
```

```
// read intermediate tag
wait AESCTL[30] == '1'          // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3
// read intermediate block counter
read AESBLKCNT0
read AESBLKCNT1
// read intermediate IV
read AESIV_0
...
read AESIV_3

-- restore and continue an interrupted operation in the AAD phase --

// configure the key store to provide a pre-loaded AES key
write KEYREADAREA 0x00000000 // load the key from ram area 0 (NOTE: The key
                             // must be pre-loaded to this area)
wait KEYREADAREA[31] == '0'  // wait until the key is loaded to the AES module
check IRQSTAT[29] == '0'     // check that the key is loaded without errors
// write intermediate tag to KEY3
write AESKEY3_0
...
write AESKEY3_3
// write intermediate IV
write AESIV_0
...
write AESIV_3
// configure the AES engine with gcm_ccm_continue
write AESCTL = 0b0011_0000_0000_0011_0000_0000_0100_1100 // program AES-GCM-128
                                                         // encryption (autonomous)
write AESDATALEN0 // write the length of the crypto block (lo)
write AESDATALEN1 // write the length of the crypto block (hi) (may be non-block size aligned)
// write intermediate block counter
write AESBLKCNT0
write AESBLKCNT1
write AESAUTHLEN      // write the length of the AAD data block (may be non-
                      // block size aligned)
// configure DMAC to process the second part of payload data
write DMACH0CTRL 0x000000001      // enable DMA channel 0
write DMACH0EXTADDR <address>     // base address of the payload data in ext. memory
write DMACH0DMALENGTH <length>    // payload data length in bytes, equal to the
                                  // payload length len({crypto_data})
                                  // (may be non-block size aligned)
write DMACH1CTL 0x000000001       // enable DMA channel 1
write DMACH1EXTADDR <address>     // base address of the output data buffer
write DMACH1LEN <length>          // output data length in bytes, equal to the
                                  // result data length len({crypto data}) (may
                                  // be non-block size aligned)
// wait for completion
wait IRQSTAT[0] == '1'            // wait for operation completed
check IRQSTAT[31] == '0'          // check for the absence of errors
write ALGSEL 0x00000000           // disable the master control/DMA clock
// read tag
wait AESCTL[30] == '1'            // wait for the SAVED_CONTEXT_RDY bit [30]
read AESTAGOUT_0
...
read AESTAGOUT_3                  // this read clears the SAVED_CONTEXT_RDY flag
// end of algorithm
```

12.5.5 Exceptions Handling

12.5.5.1 Soft Reset

If required, the AES module can be forced to abort its current active operation and go into the IDLE state using the soft reset.

The IDLE state means the following:

- The DMAC is not actively performing DMA operations.
- The cryptographic modules are in the IDLE state.
- The key-store module does not have any keys loaded.
- The master control module is in the IDLE state.
- A soft reset must be executed in the following order:

  - If DMA is used and in operation, it must be stopped.
  - The master control module must be reset through the CRYPTO:SWRESET register.
- Write the mode and length registers for the crypto core with zeroes.
  The mode and length registers are:
  - CRYPTO:AESCTL
  - CRYPTO:AESDATALEN0
  - CRYPTO:AESDATALEN1
  - CRYPTO:AESAUTHLEN

12.5.5.2 External Port Errors

The AHB master interface and the DMAC inside the crypto core can detect AHB port errors received through CRYPTO:DMAPORTERR.PORT1_AHB_ERROR.

In this situation, the DMAC disables all channels so that no new transfers are requested, while the error is captured in the status registers. The DMAPORTERR register contains information about the active channel when the AHB port error occurred. The DMAC indicates the channel completion to the master control module. The recovery procedure is as follows:

1. Issue a soft reset to the DMAC using the CRYPTO:DMASWRESET register to clear the DMAPORTERR register and initialize the channels to their default state.
2. Issue a soft reset to the master control module to clear its intermediate state.

12.5.5.3 Key Store Errors

Key store error generation is implemented for debugging purposes. In normal or specified operation, the crypto core key store writes and reads must not trigger any errors. A bus error is the only exceptional case that can result in a key store write error.

The key-store module checks that the keys are properly written to the key store RAM. When a key write error occurs, the KEY_ST_WR_ERR flag is asserted in the CRYPTO:IRQSTAT register. In this case, the key is not stored. The host must check the status of the KEY_ST_WR_ERR flag and ensure that the corresponding RAM area is not used for AES operations.

If the host tries to use a key from a nonwritten RAM area (due to software malfunction), the key-store module generates a read error. In this case, the KEY_ST_RD_ERR flag is asserted in the IRQSTAT register. The host must check the status of this flag and ensure that all remaining steps for the AES operation are not performed.

Note

In case of a read error, the key store writes a key with all bytes set to 0 to the AES engine.

12.6 Conventions and Compliances

12.6.1 Conventions Used in This Manual

Table 12-19 lists acronyms used in this document.

Table 12-19. Acronyms

Acronym	Full Term
AAD	Additional Authenticated Data
AEAD	Authenticated Encryption with Associated Data
ACT2	Addition Chaining Table with 2 address bits (4 entries)
ACT4	Addition Chaining Table with 4 address bits (16 entries)
AES	Advanced Encryption Standard
AES-CCM	AES Counter with CBC-MAC
AHB	Advanced High-Speed Bus
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CRT	Chinese Remainder Theorem
CTR	Counter Mode
DMAC	DMA Controller
DPRAM	Dual-Port Random Access Memory
ECB	Electronic Code Block
ECC	Elliptic Curve Cryptography
EIP	Embedded Intellectual Property
FIFO	First-In, First-Out
FIPS	Federal Information Processing Standard
HMAC	Hashed MAC
HW	Hardware
IP	Internet Protocol Intellectual Property
IV	Initialization Vector
LNME	Large Number Multiplier and Exponentiator
MAC	Message Authentication Code
PKA	Public Key Accelerator
PKCP	Public Key Coprocessor
RAM	Random Access Memory
ROM	Read Only Memory
TRNG	True Random Number Generator

12.6.1.1 Terminology

This manual makes frequent use of certain terms. These terms refer to structures that the crypto core uses for operations.

External memory: A memory that is externally attached to the crypto core AHB master port, and only accessible using DMAC operations. This is memory external to the module, not memory external to the MCU.

Slave interface (host processor bus): Interface of the AES and Hash Cryptoprocessor that the host processor uses to read or write registers of the module.

Tag or digest: Two interchangeable terms that indicate the result of an authentication operation. The *term digest* is used for regular hash operations, while *MAC* is used for authenticated encryption operations (AES-CCM, AES-GCM), and MAC operations (CBC-MAC).

Crypto context: A collection of parameters that define the crypto operation: mode, key, IV, and so forth.

12.6.1.2 Formulas and Nomenclature

This document contains formulas and nomenclature for different data types. Table 12-20 lists the presentation of syntax.

Table 12-20. Formulas and Nomenclature

Form	Definition
0x00 or 0h	Hexadecimal value
0_b	Binary value
0d	Decimal value
0	Digital logic 0 or low
1	Digital logic 1 or high
bit	Binary digit
8 bits	1 byte
16 bits	Halfword
32 bits	Word
64 bits	Dual-word
128 bits	Quad-word
MOD	Modulo
REM	Remainder
A & B	A logical AND B
A OR B	A logical OR B
NOR	Logical NOR
NOT A	Logical NOT
A NOR B	A logical NOR B
AB	A logic exclusive OR B or XOR
XNOR	Logic exclusive NOR
NAND	Logical NAND
DIV	Integer division
||	Concatenation
[n:m]	Size of a register or signal in bits where n > m^2 (1)

(1) 31:0 indicates a size of 32 bits with most significant bit 31 and LSB 0. 11:3 indicates a size of 9 bits with most significant bit 11 and LSB 3.

12.6.2 Compliance

AES encryption complies with NIST FIPS 197.

AES block modes (ECB, CBC, CTR) comply with NIST SP800-38A

AES CCM AEAD mode complies with NIST SP800-38C, with an addition to support zero-length MAC values to support CCM* (also known as CCM-Star) mode.

AES GCM AEAD mode complies with NIST SP800-38D

SHA-2 hash complies with NIST FIPS 180-4.

12.7 CRYPTO Registers

Table 12-21 lists the memory-mapped registers for the CRYPTO registers. All register offset addresses not listed in Table 12-21 should be considered as reserved locations and the register contents should not be modified.

Table 12-21. CRYPTO Registers

Offset	Acronym	Register Name	Section
0h	DMACH0CTL	Channel 0 Control	Section 12.7.1
4h	DMACH0EXTADDR	Channel 0 External Address	Section 12.7.2
Ch	DMACH0LEN	Channel 0 DMA Length	Section 12.7.3
18h	DMASTAT	DMAC Status	Section 12.7.4
1Ch	DMASWRESET	DMAC Software Reset	Section 12.7.5
20h	DMACH1CTL	Channel 1 Control	Section 12.7.6
24h	DMACH1EXTADDR	Channel 1 External Address	Section 12.7.7
2Ch	DMACH1LEN	Channel 1 DMA Length	Section 12.7.8
78h	DMABUSCFG	DMAC Master Run-time Parameters	Section 12.7.9
7Ch	DMAPORTERR	DMAC Port Error Raw Status	Section 12.7.10
FCh	DMAHWVER	DMAC Version	Section 12.7.11
400h	KEYWRITEAREA	Key Store Write Area	Section 12.7.12
404h	KEYWRITTENAREA	Key Store Written Area	Section 12.7.13
408h	KEYSIZE	Key Store Size	Section 12.7.14
40Ch	KEYREADAREA	Key Store Read Area	Section 12.7.15
500h + formula	AESKEY2_y	AES_KEY2_0 / AES_GHASH_H_IN_0	Section 12.7.16
510h + formula	AESKEY3_y	AES_KEY3_0 / AES_KEY2_4	Section 12.7.17
540h + formula	AESIV_y	AES initialization vector registers	Section 12.7.18
550h	AESCTL	AES Control	Section 12.7.19
554h	AESDATALEN0	AES Crypto Length 0 (LSW)	Section 12.7.20
558h	AESDATALEN1	AES Crypto Length 1 (MSW)	Section 12.7.21
55Ch	AESAUTHLEN	AES Authentication Length	Section 12.7.22
560h	AESDATAOUT0	Data Input/Output	Section 12.7.23
560h	AESDATAIN0	AES Data Input_Output 0	Section 12.7.24
564h	AESDATAOUT1	Data Input/Output	Section 12.7.25
564h	AESDATAIN1	AES Data Input_Output 0	Section 12.7.26
568h	AESDATAOUT2	Data Input/Output	Section 12.7.27
568h	AESDATAIN2	AES Data Input_Output 2	Section 12.7.28
56Ch	AESDATAOUT3	Data Input/Output	Section 12.7.29
56Ch	AESDATAIN3	AES Data Input_Output 3	Section 12.7.30
570h + formula	AESTAGOUT_y	AES Tag Out 0	Section 12.7.31
5D4h	AESCCMALNWRD	AES CCM AAD Alignment Data Word	Section 12.7.32
5D8h	AESBLKCNT0	AES Block Counter Word 0	Section 12.7.33
5DCh	AESBLKCNT1	AES Block Counter Word 1	Section 12.7.34
600h	HASHDATAIN0	HASH Data Input 0	Section 12.7.35
604h	HASHDATAIN1	HASH Data Input 1	Section 12.7.36
608h	HASHDATAIN2	HASH Data Input 2	Section 12.7.37
60Ch	HASHDATAIN3	HASH Data Input 3	Section 12.7.38
610h	HASHDATAIN4	HASH Data Input 4	Section 12.7.39
614h	HASHDATAIN5	HASH Data Input 5	Section 12.7.40
618h	HASHDATAIN6	HASH Data Input 6	Section 12.7.41

Table 12-21. CRYPTO Registers (continued)

Offset	Acronym	Register Name	Section
61Ch	HASHDATAIN7	HASH Data Input 7	Section 12.7.42
620h	HASHDATAIN8	HASH Data Input 8	Section 12.7.43
624h	HASHDATAIN9	HASH Data Input 9	Section 12.7.44
628h	HASHDATAIN10	HASH Data Input 10	Section 12.7.45
62Ch	HASHDATAIN11	HASH Data Input 11	Section 12.7.46
630h	HASHDATAIN12	HASH Data Input 12	Section 12.7.47
634h	HASHDATAIN13	HASH Data Input 13	Section 12.7.48
638h	HASHDATAIN14	HASH Data Input 14	Section 12.7.49
63Ch	HASHDATAIN15	HASH Data Input 15	Section 12.7.50
640h	HASHDATAIN16	HASH Data Input 16	Section 12.7.51
644h	HASHDATAIN17	HASH Data Input 17	Section 12.7.52
648h	HASHDATAIN18	HASH Data Input 18	Section 12.7.53
64Ch	HASHDATAIN19	HASH Data Input 19	Section 12.7.54
650h	HASHDATAIN20	HASH Data Input 20	Section 12.7.55
654h	HASHDATAIN21	HASH Data Input 21	Section 12.7.56
658h	HASHDATAIN22	HASH Data Input 22	Section 12.7.57
65Ch	HASHDATAIN23	HASH Data Input 23	Section 12.7.58
660h	HASHDATAIN24	HASH Data Input 24	Section 12.7.59
664h	HASHDATAIN25	HASH Data Input 25	Section 12.7.60
668h	HASHDATAIN26	HASH Data Input 26	Section 12.7.61
66Ch	HASHDATAIN27	HASH Data Input 27	Section 12.7.62
670h	HASHDATAIN28	HASH Data Input 28	Section 12.7.63
674h	HASHDATAIN29	HASH Data Input 29	Section 12.7.64
678h	HASHDATAIN30	HASH Data Input 30	Section 12.7.65
67Ch	HASHDATAIN31	HASH Data Input 31	Section 12.7.66
680h	HASHIOBUFCTRL	HASH Input_Output Buffer Control	Section 12.7.67
684h	HASHMODE	HASH Mode	Section 12.7.68
688h	HASHINLENL	HASH Input Length LSB	Section 12.7.69
68Ch	HASHINLENH	HASH Input Length MSB	Section 12.7.70
6C0h	HASHDIGESTA	HASH Digest A	Section 12.7.71
6C4h	HASHDIGESTB	HASH Digest B	Section 12.7.72
6C8h	HASHDIGESTC	HASH Digest C	Section 12.7.73
6CCh	HASHDIGESTD	HASH Digest D	Section 12.7.74
6D0h	HASHDIGESTE	HASH Digest E	Section 12.7.75
6D4h	HASHDIGESTF	HASH Digest F	Section 12.7.76
6D8h	HASHDIGESTG	HASH Digest G	Section 12.7.77
6DCh	HASHDIGESTH	HASH Digest H	Section 12.7.78
6E0h	HASHDIGESTI	HASH Digest I	Section 12.7.79
6E4h	HASHDIGESTJ	HASH Digest J	Section 12.7.80
6E8h	HASHDIGESTK	HASH Digest K	Section 12.7.81
6ECh	HASHDIGESTL	HASH Digest L	Section 12.7.82
6F0h	HASHDIGESTM	HASH Digest M	Section 12.7.83
6F4h	HASHDIGESTN	HASH Digest N	Section 12.7.84
6F8h	HASHDIGESTO	HASH Digest 0	Section 12.7.85
6FCh	HASHDIGESTP	HASH Digest P	Section 12.7.86

Table 12-21. CRYPTO Registers (continued)

Offset	Acronym	Register Name	Section
700h	ALGSEL	Algorithm Select	Section 12.7.87
704h	DMAPROTCTL	DMA Protection Control	Section 12.7.88
740h	SWRESET	Software Reset	Section 12.7.89
780h	IRQTYPE	Control Interrupt Configuration	Section 12.7.90
784h	IRQEN	Control Interrupt Enable	Section 12.7.91
788h	IRQCLR	Control Interrupt Clear	Section 12.7.92
78Ch	IRQSET	Control Interrupt Set	Section 12.7.93
790h	IRQSTAT	Control Interrupt Status	Section 12.7.94
7FCh	HWVER	Hardware Version	Section 12.7.95

Complex bit access types are encoded to fit into small table cells. Table 12-22 shows the codes that are used for access types in this section.

Table 12-22. CRYPTO Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

12.7.1 DMACH0CTL Register (Offset = 0h) [Reset = 00000000h]

DMACH0CTL is shown in Table 12-23.

Return to the Summary Table.

Channel 0 Control
This register is used for channel enabling and priority selection. When a channel is disabled, it becomes inactive only when all ongoing requests are finished.

Table 12-23. DMACH0CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	PRIO	R/W	0h	Channel priority 0: Low 1: High If both channels have the same priority, access of the channels to the external port is arbitrated using the round robin scheme. If one channel has a high priority and another one low, the channel with the high priority is served first, in case of simultaneous access requests.
0	EN	R/W	0h	Channel enable 0: Disabled 1: Enable Note: Disabling an active channel interrupts the DMA operation. The ongoing block transfer completes, but no new transfers are requested.

12.7.2 DMACH0EXTADDR Register (Offset = 4h) [Reset = 00000000h]

DMACH0EXTADDR is shown in Table 12-24.

Return to the Summary Table.

Channel 0 External Address

Table 12-24. DMACH0EXTADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Channel external address value When read during operation, it holds the last updated external address after being sent to the master interface. Note: The crypto DMA copies out upto 3 bytes until it hits a word boundary, thus the address need not be word aligned.

12.7.3 DMACH0LEN Register (Offset = Ch) [Reset = 00000000h]

DMACH0LEN is shown in Table 12-25.

Return to the Summary Table.

Channel 0 DMA Length

Table 12-25. DMACH0LEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DMALEN	R/W	0h	Channel DMA length in bytes During configuration, this register contains the DMA transfer length in bytes. During operation, it contains the last updated value of the DMA transfer length after being sent to the master interface. Note: Setting this register to a nonzero value starts the transfer if the channel is enabled. Therefore, this register must be written last when setting up a DMA channel.

12.7.4 DMASTAT Register (Offset = 18h) [Reset = 00000000h]

DMASTAT is shown in Table 12-26.

Return to the Summary Table.

DMAC Status
This register provides the actual state of each DMA channel. It also reports port errors in case these were received by the master interface module during the data transfer.

Table 12-26. DMASTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reserved
17	PORT_ERR	R	0h	Reflects possible transfer errors on the AHB port.
16-2	RESERVED	R	0h	Reserved
1	CH1_ACT	R	0h	A value of 1 indicates that channel 1 is active (DMA transfer on-going).
0	CH0_ACT	R	0h	A value of 1 indicates that channel 0 is active (DMA transfer on-going).

12.7.5 DMASWRESET Register (Offset = 1Ch) [Reset = 00000000h]

DMASWRESET is shown in Table 12-27.

Return to the Summary Table.

DMAC Software Reset
Software reset is used to reset the DMAC to stop all transfers and clears the port error status register. After the software reset is performed, all the channels are disabled and no new requests are performed by the channels. The DMAC waits for the existing (active) requests to finish and accordingly sets the DMASTAT.

Table 12-27. DMASWRESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SWRES	W	0h	Software reset enable 0 : Disabled 1 : Enabled (self-cleared to 0) Completion of the software reset must be checked through the DMASTAT

12.7.6 DMACH1CTL Register (Offset = 20h) [Reset = 00000000h]

DMACH1CTL is shown in Table 12-28.

Return to the Summary Table.

Channel 1 Control
This register is used for channel enabling and priority selection. When a channel is disabled, it becomes inactive only when all ongoing requests are finished.

Table 12-28. DMACH1CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	PRIO	R/W	0h	Channel priority 0: Low 1: High If both channels have the same priority, access of the channels to the external port is arbitrated using the round robin scheme. If one channel has a high priority and another one low, the channel with the high priority is served first, in case of simultaneous access requests.
0	EN	R/W	0h	Channel enable 0: Disabled 1: Enable Note: Disabling an active channel interrupts the DMA operation. The ongoing block transfer completes, but no new transfers are requested.

12.7.7 DMACH1EXTADDR Register (Offset = 24h) [Reset = 00000000h]

DMACH1EXTADDR is shown in Table 12-29.

Return to the Summary Table.

Channel 1 External Address

Table 12-29. DMACH1EXTADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Channel external address value. When read during operation, it holds the last updated external address after being sent to the master interface. Note: The crypto DMA copies out upto 3 bytes until it hits a word boundary, thus the address need not be word aligned.

12.7.8 DMACH1LEN Register (Offset = 2Ch) [Reset = 00000000h]

DMACH1LEN is shown in Table 12-30.

Return to the Summary Table.

Channel 1 DMA Length

Table 12-30. DMACH1LEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DMALEN	R/W	0h	Channel DMA length in bytes. During configuration, this register contains the DMA transfer length in bytes. During operation, it contains the last updated value of the DMA transfer length after being sent to the master interface. Note: Setting this register to a nonzero value starts the transfer if the channel is enabled. Therefore, this register must be written last when setting up a DMA channel.

12.7.9 DMABUSCFG Register (Offset = 78h) [Reset = 00002400h]

DMABUSCFG is shown in Table 12-31.

Return to the Summary Table.

DMAC Master Run-time Parameters
This register defines all the run-time parameters for the AHB master interface port. These parameters are required for the proper functioning of the EIP-101m AHB master adapter.

Table 12-31. DMABUSCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	AHB_MST1_BURST_SIZE	R/W	2h	Maximum burst size that can be performed on the AHB bus 2h = 4_BYTE : 4 bytes 3h = 8_BYTE : 8 bytes 4h = 16_BYTE : 16 bytes 5h = 32_BYTE : 32 bytes 6h = 64_BYTE : 64 bytes
11	AHB_MST1_IDLE_EN	R/W	0h	Idle insertion between consecutive burst transfers on AHB 0h = Do not insert idle transfers. 1h = Idle transfer insertion enabled
10	AHB_MST1_INCR_EN	R/W	1h	Burst length type of AHB transfer 0h = Unspecified length burst transfers 1h = Fixed length bursts or single transfers
9	AHB_MST1_LOCK_EN	R/W	0h	Locked transform on AHB 0h = Transfers are not locked 1h = Transfers are locked
8	AHB_MST1_BIGEND	R/W	0h	Endianess for the AHB master 0h = Little Endian 1h = Big Endian
7-0	RESERVED	R	0h	Reserved

12.7.10 DMAPORTERR Register (Offset = 7Ch) [Reset = 00000000h]

DMAPORTERR is shown in Table 12-32.

Return to the Summary Table.

DMAC Port Error Raw Status
This register provides the actual status of individual port errors. It also indicates which channel is serviced by an external AHB port (which is frozen by a port error). A port error aborts operations on all serviced channels (channel enable bit is forced to 0) and prevents further transfers via that port until the error is cleared by writing to the DMASWRESET register.

Table 12-32. DMAPORTERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	Reserved
12	PORT1_AHB_ERROR	R	0h	A value of 1 indicates that the EIP-101 has detected an AHB bus error
11-10	RESERVED	R	0h	Reserved
9	PORT1_CHANNEL	R	0h	Indicates which channel has serviced last (channel 0 or channel 1) by AHB master port.
8-0	RESERVED	R	0h	Reserved

12.7.11 DMAHWVER Register (Offset = FCh) [Reset = 01012ED1h]

DMAHWVER is shown in Table 12-33.

Return to the Summary Table.

DMAC Version
This register contains an indication (or signature) of the EIP type of this DMAC, as well as the hardware version/ patch numbers.

Table 12-33. DMAHWVER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	HW_MAJOR_VERSION	R	1h	Major version number
23-20	HW_MINOR_VERSION	R	0h	Minor version number
19-16	HW_PATCH_LEVEL	R	1h	Patch level Starts at 0 at first delivery of this version
15-8	EIP_NUMBER_COMPL	R	2Eh	Bit-by-bit complement of the EIP_NUMBER field bits.
7-0	EIP_NUMBER	R	D1h	Binary encoding of the EIP-number of this DMA controller (209)

12.7.12 KEYWRITEAREA Register (Offset = 400h) [Reset = 00000000h]

KEYWRITEAREA is shown in Table 12-34.

Return to the Summary Table.

Key Store Write Area
This register defines where the keys should be written in the key store RAM. After writing this register, the key store module is ready to receive the keys through a DMA operation. In case the key data transfer triggered an error in the key store, the error will be available in the interrupt status register after the DMA is finished. The key store write-error is asserted when the programmed/selected area is not completely written. This error is also asserted when the DMA operation writes to ram areas that are not selected.
The key store RAM is divided into 8 areas of 128 bits.
192-bit keys written in the key store RAM should start on boundaries of 256 bits. This means that writing a 192-bit key to the key store RAM must be done by writing 256 bits of data with the 64 most-significant bits set to 0. These bits are ignored by the AES engine.

Table 12-34. KEYWRITEAREA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	RAM_AREA7	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA7 is not selected to be written. 1: RAM_AREA7 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
6	RAM_AREA6	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA6 is not selected to be written. 1: RAM_AREA6 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
5	RAM_AREA5	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA5 is not selected to be written. 1: RAM_AREA5 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written

Table 12-34. KEYWRITEAREA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	RAM_AREA4	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA4 is not selected to be written. 1: RAM_AREA4 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
3	RAM_AREA3	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA3 is not selected to be written. 1: RAM_AREA3 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
2	RAM_AREA2	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA2 is not selected to be written. 1: RAM_AREA2 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
1	RAM_AREA1	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA1 is not selected to be written. 1: RAM_AREA1 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written
0	RAM_AREA0	R/W	0h	Each RAM_AREAx represents an area of 128 bits. Select the key store RAM area(s) where the key(s) needs to be written 0: RAM_AREA0 is not selected to be written. 1: RAM_AREA0 is selected to be written. Writing to multiple RAM locations is possible only when the selected RAM areas are sequential. Keys that require more than one RAM locations (key size is 192 or 256 bits), must start at one of the following areas: RAM_AREA0, RAM_AREA2, RAM_AREA4, or RAM_AREA6. 0h = This RAM area is not selected to be written 1h = This RAM area is selected to be written

12.7.13 KEYWRITTENAREA Register (Offset = 404h) [Reset = 00000000h]

KEYWRITTENAREA is shown in Table 12-35.

Return to the Summary Table.

Key Store Written Area
This register shows which areas of the key store RAM contain valid written keys.
When a new key needs to be written to the key store, on a location that is already occupied by a valid key, this key area must be cleared first. This can be done by writing this register before the new key is written to the key store memory.
Attempting to write to a key area that already contains a valid key is not allowed and results in an error.

Table 12-35. KEYWRITTENAREA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	RAM_AREA_WRITTEN7	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
6	RAM_AREA_WRITTEN6	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
5	RAM_AREA_WRITTEN5	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
4	RAM_AREA_WRITTEN4	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
3	RAM_AREA_WRITTEN3	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
2	RAM_AREA_WRITTEN2	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information

Table 12-35. KEYWRITTENAREA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RAM_AREA_WRITTEN1	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory. 0h = This RAM area is not written with valid key information 1h = This RAM area is written with valid key information
0	RAM_AREA_WRITTEN0	R/W1C	0h	On read this bit returns the key area written status. This bit can be reset by writing a 1. Note: This register will be reset on a soft reset initiated by writing to DMASWRESET.SWRES. After a soft reset, all keys must be rewritten to the key store memory.

12.7.14 KEYSIZE Register (Offset = 408h) [Reset = 00000001h]

KEYSIZE is shown in Table 12-36.

Return to the Summary Table.

Key Store Size
This register defines the size of the keys that are written with DMA. This register should be configured before writing to the KEY_STORE_WRITE_AREA register.

Table 12-36. KEYSIZE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	SIZE	R/W	1h	Key size: 00: Reserved When writing this to this register, the KEY_STORE_WRITTEN_AREA register is reset. 1h = 128_BIT : 128 bits 2h = 192_BIT : 192 bits 3h = 256_BIT : 256 bits

12.7.15 KEYREADAREA Register (Offset = 40Ch) [Reset = 00000008h]

KEYREADAREA is shown in Table 12-37.

Return to the Summary Table.

Key Store Read Area
This register selects the key store RAM area from where the key needs to be read that will be used for an AES operation. The operation directly starts after writing this register. When the operation is finished, the status of the key store read operation is available in the interrupt status register. Key store read error is asserted when a RAM area is selected which does not contain valid written key.

Table 12-37. KEYREADAREA Register Field Descriptions

Bit	Field	Type	Reset	Description
31	BUSY	R	0h	Key store operation busy status flag (read only): 0: Operation is complete. 1: Operation is not completed and the key store is busy.
30-4	RESERVED	R	0h	Reserved
3-0	RAM_AREA	R/W	8h	Selects the area of the key store RAM from where the key needs to be read that will be writen to the AES engine RAM_AREA: RAM areas RAM_AREA0, RAM_AREA2, RAM_AREA4 and RAM_AREA6 are the only valid read areas for 192 and 256 bits key sizes. Only RAM areas that contain valid written keys can be selected. 0h = RAM Area 0 1h = RAM Area 1 2h = RAM Area 2 3h = RAM Area 3 4h = RAM Area 4 5h = RAM Area 5 6h = RAM Area 6 7h = RAM Area 7 8h = No RAM

12.7.16 AESKEY2_y Register (Offset = 500h + formula) [Reset = 00000000h]

AESKEY2_y is shown in Table 12-38.

Return to the Summary Table.

AES_KEY2_0 / AES_GHASH_H_IN_0
Second Key / GHASH Key (internal, but clearable)
The following registers are not accessible through the host for reading and writing. They are used to store internally calculated key information and intermediate results. However, when the host performs a write to the any of the respective AES_KEY2_n or AES_KEY3_n addresses, respectively the whole 128-bit AES_KEY2_n or AES_KEY3_n register is cleared to 0s.
The AES_GHASH_H_IN_n registers (required for GHASH, which is part of GCM) are mapped to the AES_KEY2_n registers. The (intermediate) authentication result for GCM and CCM is stored in the AES_KEY3_n register.

Offset = 500h + (y * 4h); where y = 0h to 3h

Table 12-38. AESKEY2_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_KEY2	W	0h	AES_KEY2/AES_GHASH_H[31:0] For GCM: -[127:0] - GHASH_H - The internally calculated GHASH key is stored in these registers. Only used for modes that use the GHASH function (GCM). -[255:128] - This register is used to store intermediate values and is initialized with 0s when loading a new key. For CCM: -[255:0] - This register is used to store intermediate values. For CBC-MAC: -[255:0] - ZEROES - This register must remain 0.

12.7.17 AESKEY3_y Register (Offset = 510h + formula) [Reset = 00000000h]

AESKEY3_y is shown in Table 12-39.

Return to the Summary Table.

AES_KEY3_0 / AES_KEY2_4
Third Key / Second Key (internal, but clearable)
The following registers are not accessible through the host for reading and writing. They are used to store internally calculated key information and intermediate results. However, when the host performs a write to the any of the respective AES_KEY2_n or AES_KEY3_n addresses, respectively the whole 128-bit AES_KEY2_n or AES_KEY3_n register is cleared to 0s.
The AES_GHASH_H_IN_n registers (required for GHASH, which is part of GCM) are mapped to the AES_KEY2_n registers. The (intermediate) authentication result for GCM and CCM is stored in the AES_KEY3_n register.

Offset = 510h + (y * 4h); where y = 0h to 3h

Table 12-39. AESKEY3_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_KEY3	W	0h	AES_KEY3[31:0]/AES_KEY2[159:128] For GCM: -[127:0] - GHASH_H - The internally calculated GHASH key is stored in these registers. Only used for modes that use the GHASH function (GCM). -[255:128] - This register is used to store intermediate values and is initialized with 0s when loading a new key. For CCM: -[255:0] - This register is used to store intermediate values. For CBC-MAC: -[255:0] - ZEROES - This register must remain 0.

12.7.18 AESIV_y Register (Offset = 540h + formula) [Reset = 00000000h]

AESIV_y is shown in Table 12-40.

Return to the Summary Table.

AES initialization vector registers
These registers are used to provide and read the IV from the AES engine.

Offset = 540h + (y * 4h); where y = 0h to 3h

Table 12-40. AESIV_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_IV	R/W	0h	AES_IV[31:0] Initialization vector Used for regular non-ECB modes (CBC/CTR): -[127:0] - AES_IV - For regular AES operations (CBC and CTR) these registers must be written with a new 128-bit IV. After an operation, these registers contain the latest 128-bit result IV, generated by the EIP-120t. If CTR mode is selected, this value is incremented with 0x1: After first use - When a new data block is submitted to the engine For GCM: -[127:0] - AES_IV - For GCM operations, these registers must be written with a new 128-bit IV. After an operation, these registers contain the updated 128-bit result IV, generated by the EIP-120t. Note that bits [127:96] of the IV represent the initial counter value (which is 1 for GCM) and must therefore be initialized to 0x01000000. This value is incremented with 0x1: After first use - When a new data block is submitted to the engine. For CCM: -[127:0] - A0: For CCM this field must be written with value A0, this value is the concatenation of: A0-flags (5-bits of 0 and 3-bits 'L'), Nonce and counter value. 'L' must be a copy from the 'L' value of the AES_CTRL register. This 'L' indicates the width of the Nonce and counter. The loaded counter must be initialized to 0. The total width of A0 is 128-bit. For CBC-MAC: -[127:0] - Zeroes - For CBC-MAC this register must be written with 0s at the start of each operation. After an operation, these registers contain the 128-bit TAG output, generated by the EIP-120t.

12.7.19 AESCTL Register (Offset = 550h) [Reset = 80000000h]

AESCTL is shown in Table 12-41.

Return to the Summary Table.

AES Control
AES input/output buffer control and mode register
This register specifies the AES mode of operation for the EIP-120t.
Electronic codebook (ECB) mode is automatically selected if bits [28:5] of this register are all 0.

Table 12-41. AESCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CONTEXT_READY	R	1h	If 1, this read-only status bit indicates that the context data registers can be overwritten and the host is permitted to write the next context.
30	SAVED_CONTEXT_RDY	R/W	0h	If 1, this status bit indicates that an AES authentication TAG and/or IV block(s) is/are available for the host to retrieve. This bit is only asserted if the save_context bit is set to 1. The bit is mutual exclusive with the context_ready bit. Writing one clears the bit to 0, indicating the AES core can start its next operation. This bit is also cleared when the 4th word of the output TAG and/or IV is read. Note: All other mode bit writes are ignored when this mode bit is written with 1. Note: This bit is controlled automatically by the EIP-120t for TAG read DMA operations.
29	SAVE_CONTEXT	R/W	0h	This bit indicates that an authentication TAG or result IV needs to be stored as a result context. Typically this bit must be set for authentication modes returning a TAG (CBC-MAC, GCM and CCM), or for basic encryption modes that require future continuation with the current result IV. If this bit is set, the engine retains its full context until the TAG and/or IV registers are read. The TAG or IV must be read before the AES engine can start a new operation.
28	GCM_CCM_CONTINUE	R/W	0h	Continue processing of an interrupted AES-GCM or AES-CCM operation in the crypto/payload phase. Set this write-only signal to ‘1b’ together with the regular mode bit settings for a GCM or CCM operation, to continue processing from the next full block (128 bits) boundary. Before setting this bit all applicable context to resume processing must have been loaded into the engine: Keys, IV, intermediate digest/TAG and block counter. The mode can be written together with this bit, as it is part of the same register.
27	GET_DIGEST	R/W	0h	Interrupt processing and generate an intermediate digest during an AES-GCM or AES-CCM operation. Set this write-only signal to ‘1b’ to interrupt GCM or CCM processing at the next full block (128 bits) boundary. An intermediate digest may be requested during the encryption/decryption data phase or in the AAD phase. Interruption can only be done on full block (128 bits) boundaries.
26	GCM_CCM_CONTINUE_AAD	R/W	0h	Continue processing of an interrupted AES-GCM or AES-CCM operation in the AAD phase. Set this write-only signal to ‘1b’ together with the regular mode bit settings for a GCM or CCM operation, to continue processing from the next full AAD block (128 bits) boundary. Before setting this bit all applicable context to resume processing must have been loaded into the engine: Keys, IV, intermediate digest/TAG, block counter and the CCM align data word (the latter is for CCM mode only). The mode can be written together with this bit, as it is part of the same register.
25	XCBC_MAC	R/W	0h	Set to ‘1’ to select AES-XCBC MAC mode. The direction bit must be set to ‘1’ for this mode. Selecting this mode requires writing the length register

Table 12-41. AESCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-22	CCM_M	R/W	0h	Defines M, which indicates the length of the authentication field for CCM operations the authentication field length equals two times (the value of CCM-M plus one). Note: The EIP-120t always returns a 128-bit authentication field, of which the M least significant bytes are valid. All values are supported.
21-19	CCM_L	R/W	0h	Defines L, which indicates the width of the length field for CCM operations the length field in bytes equals the value of CMM-L plus one. All values are supported.
18	CCM	R/W	0h	If set to 1, AES-CCM is selected AES-CCM is a combined mode, using AES for authentication and encryption. Note: Selecting AES-CCM mode requires writing of the AAD length register after all other registers. Note: The CTR mode bit in this register must also be set to 1 to enable AES-CTR selecting other AES modes than CTR mode is invalid.
17-16	GCM	R/W	0h	Set these bits to 11 to select AES-GCM mode. AES-GCM is a combined mode, using the Galois field multiplier GF(2 to the power of 128) for authentication and AES-CTR mode for encryption. Note: The CTR mode bit in this register must also be set to 1 to enable AES-CTR Bit combination description: 00 = No GCM mode 01 = Reserved, do not select 10 = Reserved, do not select 11 = Autonomous GHASH (both H- and Y0-encrypted calculated internally) Note: The EIP-120t-1 configuration only supports mode 11 (autonomous GHASH), other GCM modes are not allowed.
15	CBC_MAC	R/W	0h	Set to 1 to select AES-CBC MAC mode. The direction bit must be set to 1 for this mode. Selecting this mode requires writing the length register after all other registers.
14-9	RESERVED	R	0h	Reserved
8-7	CTR_WIDTH	R/W	0h	Specifies the counter width for AES-CTR mode 00 = 32-bit counter 01 = 64-bit counter 10 = 96-bit counter 11 = 128-bit counter 0h = 32_BIT : 32 bits 1h = 64_BIT : 64 bits 2h = 96_BIT : 96 bits 3h = 128_BIT : 128 bits
6	CTR	R/W	0h	If set to 1, AES counter mode (CTR) is selected. Note: This bit must also be set for GCM and CCM, when encryption/decryption is required.
5	CBC	R/W	0h	If set to 1, cipher-block-chaining (CBC) mode is selected.
4-3	KEY_SIZE	R	0h	This read-only field specifies the key size. The key size is automatically configured when a new key is loaded through the key store module. 00 = N/A - Reserved 01 = 128-bit 10 = 192-bit 11 = 256-bit
2	DIR	R/W	0h	If set to 1 an encrypt operation is performed. If set to 0 a decrypt operation is performed. This bit must be written with a 1 when CBC-MAC is selected.

Table 12-41. AESCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	INPUT_READY	R/W	0h	If 1, this status bit indicates that the 16-byte AES input buffer is empty. The host is permitted to write the next block of data. Writing 0 clears the bit to 0 and indicates that the AES core can use the provided input data block. Writing 1 to this bit is ignored. Note: For DMA operations, this bit is automatically controlled by the EIP-120t. After reset, this bit is 0. After writing a context, this bit becomes 1.
0	OUTPUT_READY	R/W	0h	If 1, this status bit indicates that an AES output block is available to be retrieved by the host. Writing 0 clears the bit to 0 and indicates that output data is read by the host. The AES core can provide a next output data block. Writing 1 to this bit is ignored. Note: For DMA operations, this bit is automatically controlled by the EIP-120t.

12.7.20 AESDATALEN0 Register (Offset = 554h) [Reset = 00000000h]

AESDATALEN0 is shown in Table 12-42.

Return to the Summary Table.

AES Crypto Length 0 (LSW)
These registers are used to write the Length values to the EIP-120t. While processing, the length values decrement to 0. If both lengths are 0, the data stream is finished and a new context is requested. For basic AES modes (ECB, CBC, and CTR), a crypto length of 0 can be written if multiple streams need to be processed with the same key. Writing 0 length results in continued data requests until a new context is written. For the other modes (CBC-MAC, GCM, and CCM) no (new) data requests are done if the length decrements to or equals 0. It is advised to write a new length per packet. If the length registers decrement to 0, no new data is processed until a new context or length value is written.
When writing a new mode without writing the length registers, the length register values from the previous context is reused.

Table 12-42. AESDATALEN0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	C_LENGTH	W	0h	C_LENGTH[31:0] Bits [60:0] of the crypto length registers (LSW and MSW) store the cryptographic data length in bytes for all modes. Once processing with this context is started, this length decrements to 0. Data lengths up to (261: 1) bytes are allowed. For GCM, any value up to 236 - 32 bytes can be used. This is because a 32-bit counter mode is used the maximum number of 128-bit blocks is 232 - 2, resulting in a maximum number of bytes of 236 - 32. A write to this register triggers the engine to start using this context. This is valid for all modes except GCM and CCM. Note: For the combined modes (GCM and CCM), this length does not include the authentication only data the authentication length is specified in the AESAUTHLEN register All modes must have a length greater than 0. For the combined modes, it is allowed to have one of the lengths equal to 0. For the basic encryption modes (ECB, CBC, and CTR) it is allowed to program zero to the length field in that case the length is assumed infinite. All data must be byte (8-bit) aligned for stream cipher modes bit aligned data streams are not supported by the EIP-120t. For block cipher modes, the data length must be programmed in multiples of the block cipher size, 16 bytes. For a host read operation, these registers return all-0s.

12.7.21 AESDATALEN1 Register (Offset = 558h) [Reset = 00000000h]

AESDATALEN1 is shown in Table 12-43.

Return to the Summary Table.

AES Crypto Length 1 (MSW)
These registers are used to write the Length values to the EIP-120t. While processing, the length values decrement to 0. If both lengths are 0, the data stream is finished and a new context is requested. For basic AES modes (ECB, CBC, and CTR), a crypto length of 0 can be written if multiple streams need to be processed with the same key. Writing 0 length results in continued data requests until a new context is written. For the other modes (CBC-MAC, GCM and CCM) no (new) data requests are done if the length decrements to or equals 0.
It is advised to write a new length per packet. If the length registers decrement to 0, no new data is processed until a new context or length value is written.
When writing a new mode without writing the length registers, the length register values from the previous context is reused.

Table 12-43. AESDATALEN1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	Reserved
28-0	C_LENGTH	W	0h	C_LENGTH[60:32] Bits [60:0] of the crypto length registers (LSW and MSW) store the cryptographic data length in bytes for all modes. Once processing with this context is started, this length decrements to 0. Data lengths up to (261: 1) bytes are allowed. For GCM, any value up to 236 - 32 bytes can be used. This is because a 32-bit counter mode is used the maximum number of 128-bit blocks is 232 - 2, resulting in a maximum number of bytes of 236 - 32. A write to this register triggers the engine to start using this context. This is valid for all modes except GCM and CCM. Note: For the combined modes (GCM and CCM), this length does not include the authentication only data the authentication length is specified in the AESAUTHLEN register All modes must have a length greater than 0. For the combined modes, it is allowed to have one of the lengths equal to 0. For the basic encryption modes (ECB, CBC, and CTR) it is allowed to program zero to the length field in that case the length is assumed infinite. All data must be byte (8-bit) aligned for stream cipher modes bit aligned data streams are not supported by the EIP-120t. For block cipher modes, the data length must be programmed in multiples of the block cipher size, 16 bytes. For a host read operation, these registers return all-0s.

12.7.22 AESAUTHLEN Register (Offset = 55Ch) [Reset = 00000000h]

AESAUTHLEN is shown in Table 12-44.

Return to the Summary Table.

AES Authentication Length

Table 12-44. AESAUTHLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AUTH_LENGTH	W	0h	Bits [31:0] of the authentication length register store the authentication data length in bytes for combined modes only (GCM or CCM). Supported AAD-lengths for CCM are from 0 to (2^{16} - 2^{8}) bytes. For GCM any value up to (2^{32} - 1) bytes can be used. Once processing with this context is started, this length decrements to 0. A write to this register triggers the engine to start using this context for GCM and CCM. For a host read operation, these registers return all-0s.

12.7.23 AESDATAOUT0 Register (Offset = 560h) [Reset = 00000000h]

AESDATAOUT0 is shown in Table 12-45.

Return to the Summary Table.

Data Input/Output

Table 12-45. AESDATAOUT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Data register 0 for output block data from the Crypto peripheral. These bits = AES Output Data[31:0] of {127:0] For normal operations, this register is not used, since data input and output is transferred from and to the AES engine via DMA. For a Host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range will read one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, AESCTL.OUTPUT_READY must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.24 AESDATAIN0 Register (Offset = 560h) [Reset = 00000000h]

AESDATAIN0 is shown in Table 12-46.

Return to the Summary Table.

AES Data Input_Output 0
The data registers are typically accessed through the DMA and not with host writes and/or reads. However, for debugging purposes the data input/output registers can be accessed via host write and read operations. The registers are used to buffer the input/output data blocks to/from the EIP-120t.
Note: The data input buffer (AES_DATA_IN_n) and data output buffer (AES_DATA_OUT_n) are mapped to the same address locations.
Writes (both DMA and host) to these addresses load the Input Buffer while reads pull from the Output Buffer. Therefore, for write access, the data input buffer is written
for read access, the data output buffer is read. The data input buffer must be written before starting an operation. The data output buffer contains valid data on completion of an operation. Therefore, any 128-bit data block can be split over multiple 32-bit word transfers
these can be mixed with other host transfers over the external interface.

Table 12-46. AESDATAIN0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_DATA_IN_OUT	W	0h	AES input data[31:0] / AES output data[31:0] Data registers for input/output block data to/from the EIP-120t. For normal operations, this register is not used, since data input and output is transferred from and to the AES core via DMA. For a host write operation, these registers must be written with the 128-bit input block for the next AES operation. Writing at a word-aligned offset within this address range stores the word (4 bytes) of data into the corresponding position of 4-word deep (16 bytes = 128-bit AES block) data input buffer. This buffer is used for the next AES operation. If the last data block is not completely filled with valid data (see notes below), it is allowed to write only the words with valid data. Next AES operation is triggered by writing to the input_ready flag of the AES_CTRL register. For a host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range reads one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, the output_ready flag of the AES_CTRL register must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: AES typically operates on 128 bits block multiple input data. The CTR, GCM and CCM modes form an exception. The last block of a CTR-mode message may contain less than 128 bits (refer to [NIST 800-38A]). For GCM/CCM, the last block of both AAD and message data may contain less than 128 bits (refer to [NIST 800-38D]). The EIP-120t automatically pads or masks misaligned ending data blocks with 0s for GCM, CCM and CBC-MAC. For CTR mode, the remaining data in an unaligned data block is ignored. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.25 AESDATAOUT1 Register (Offset = 564h) [Reset = 00000000h]

AESDATAOUT1 is shown in Table 12-47.

Return to the Summary Table.

Data Input/Output

Table 12-47. AESDATAOUT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Data register 0 for output block data from the Crypto peripheral. These bits = AES Output Data[31:0] of {127:0] For normal operations, this register is not used, since data input and output is transferred from and to the AES engine via DMA. For a Host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range will read one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, AESCTL.OUTPUT_READY must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.26 AESDATAIN1 Register (Offset = 564h) [Reset = 00000000h]

AESDATAIN1 is shown in Table 12-48.

Return to the Summary Table.

AES Data Input_Output 0
The data registers are typically accessed through the DMA and not with host writes and/or reads. However, for debugging purposes the data input/output registers can be accessed via host write and read operations. The registers are used to buffer the input/output data blocks to/from the EIP-120t.
Note: The data input buffer (AES_DATA_IN_n) and data output buffer (AES_DATA_OUT_n) are mapped to the same address locations.
Writes (both DMA and host) to these addresses load the Input Buffer while reads pull from the Output Buffer. Therefore, for write access, the data input buffer is written
for read access, the data output buffer is read. The data input buffer must be written before starting an operation. The data output buffer contains valid data on completion of an operation. Therefore, any 128-bit data block can be split over multiple 32-bit word transfers
these can be mixed with other host transfers over the external interface.

Table 12-48. AESDATAIN1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_DATA_IN_OUT	W	0h	AES input data[31:0] / AES output data[63:32] Data registers for input/output block data to/from the EIP-120t. For normal operations, this register is not used, since data input and output is transferred from and to the AES core via DMA. For a host write operation, these registers must be written with the 128-bit input block for the next AES operation. Writing at a word-aligned offset within this address range stores the word (4 bytes) of data into the corresponding position of 4-word deep (16 bytes = 128-bit AES block) data input buffer. This buffer is used for the next AES operation. If the last data block is not completely filled with valid data (see notes below), it is allowed to write only the words with valid data. Next AES operation is triggered by writing to the input_ready flag of the AES_CTRL register. For a host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range reads one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, the output_ready flag of the AES_CTRL register must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: AES typically operates on 128 bits block multiple input data. The CTR, GCM and CCM modes form an exception. The last block of a CTR-mode message may contain less than 128 bits (refer to [NIST 800-38A]). For GCM/CCM, the last block of both AAD and message data may contain less than 128 bits (refer to [NIST 800-38D]). The EIP-120t automatically pads or masks misaligned ending data blocks with 0s for GCM, CCM and CBC-MAC. For CTR mode, the remaining data in an unaligned data block is ignored. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.27 AESDATAOUT2 Register (Offset = 568h) [Reset = 00000000h]

AESDATAOUT2 is shown in Table 12-49.

Return to the Summary Table.

Data Input/Output

Table 12-49. AESDATAOUT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Data register 0 for output block data from the Crypto peripheral. These bits = AES Output Data[31:0] of {127:0] For normal operations, this register is not used, since data input and output is transferred from and to the AES engine via DMA. For a Host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range will read one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, AESCTL.OUTPUT_READY must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.28 AESDATAIN2 Register (Offset = 568h) [Reset = 00000000h]

AESDATAIN2 is shown in Table 12-50.

Return to the Summary Table.

AES Data Input_Output 2
The data registers are typically accessed via DMA and not with host writes and/or reads. However, for debugging purposes the Data Input/Output Registers can be accessed via host write and read operations. The registers are used to buffer the input/output data blocks to/from the EIP-120t.
Note: The data input buffer (AES_DATA_IN_n) and data output buffer (AES_DATA_OUT_n) are mapped to the same address locations.
Writes (both DMA and host) to these addresses load the Input Buffer while reads pull from the Output Buffer. Therefore, for write access, the data input buffer is written
for read access, the data output buffer is read. The data input buffer must be written before starting an operation. The data output buffer contains valid data on completion of an operation. Therefore, any 128-bit data block can be split over multiple 32-bit word transfers
these can be mixed with other host transfers over the external interface.

Table 12-50. AESDATAIN2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_DATA_IN_OUT	W	0h	AES input data[95:64] / AES output data[95:64] Data registers for input/output block data to/from the EIP-120t. For normal operations, this register is not used, since data input and output is transferred from and to the AES core via DMA. For a host write operation, these registers must be written with the 128-bit input block for the next AES operation. Writing at a word-aligned offset within this address range stores the word (4 bytes) of data into the corresponding position of 4-word deep (16 bytes = 128-bit AES block) data input buffer. This buffer is used for the next AES operation. If the last data block is not completely filled with valid data (see notes below), it is allowed to write only the words with valid data. Next AES operation is triggered by writing to the input_ready flag of the AES_CTRL register. For a host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range reads one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, the output_ready flag of the AES_CTRL register must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: AES typically operates on 128 bits block multiple input data. The CTR, GCM and CCM modes form an exception. The last block of a CTR-mode message may contain less than 128 bits (refer to [NIST 800-38A]). For GCM/CCM, the last block of both AAD and message data may contain less than 128 bits (refer to [NIST 800-38D]). The EIP-120t automatically pads or masks misaligned ending data blocks with 0s for GCM, CCM and CBC-MAC. For CTR mode, the remaining data in an unaligned data block is ignored. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.29 AESDATAOUT3 Register (Offset = 56Ch) [Reset = 00000000h]

AESDATAOUT3 is shown in Table 12-51.

Return to the Summary Table.

Data Input/Output

Table 12-51. AESDATAOUT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Data register 0 for output block data from the Crypto peripheral. These bits = AES Output Data[31:0] of {127:0] For normal operations, this register is not used, since data input and output is transferred from and to the AES engine via DMA. For a Host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range will read one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, AESCTL.OUTPUT_READY must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.30 AESDATAIN3 Register (Offset = 56Ch) [Reset = 00000000h]

AESDATAIN3 is shown in Table 12-52.

Return to the Summary Table.

AES Data Input_Output 3
The data registers are typically accessed via DMA and not with host writes and/or reads. However, for debugging purposes the Data Input/Output Registers can be accessed via host write and read operations. The registers are used to buffer the input/output data blocks to/from the EIP-120t.
Note: The data input buffer (AES_DATA_IN_n) and data output buffer (AES_DATA_OUT_n) are mapped to the same address locations.
Writes (both DMA and host) to these addresses load the Input Buffer while reads pull from the Output Buffer. Therefore, for write access, the data input buffer is written
for read access, the data output buffer is read. The data input buffer must be written before starting an operation. The data output buffer contains valid data on completion of an operation. Therefore, any 128-bit data block can be split over multiple 32-bit word transfers
these can be mixed with other host transfers over the external interface.

Table 12-52. AESDATAIN3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_DATA_IN_OUT	W	0h	AES input data[127:96] / AES output data[127:96] Data registers for input/output block data to/from the EIP-120t. For normal operations, this register is not used, since data input and output is transferred from and to the AES core via DMA. For a host write operation, these registers must be written with the 128-bit input block for the next AES operation. Writing at a word-aligned offset within this address range stores the word (4 bytes) of data into the corresponding position of 4-word deep (16 bytes = 128-bit AES block) data input buffer. This buffer is used for the next AES operation. If the last data block is not completely filled with valid data (see notes below), it is allowed to write only the words with valid data. Next AES operation is triggered by writing to the input_ready flag of the AES_CTRL register. For a host read operation, these registers contain the 128-bit output block from the latest AES operation. Reading from a word-aligned offset within this address range reads one word (4 bytes) of data out the 4-word deep (16 bytes = 128-bits AES block) data output buffer. The words (4 words, one full block) should be read before the core will move the next block to the data output buffer. To empty the data output buffer, the output_ready flag of the AES_CTRL register must be written. For the modes with authentication (CBC-MAC, GCM and CCM), the invalid (message) bytes/words can be written with any data. Note: AES typically operates on 128 bits block multiple input data. The CTR, GCM and CCM modes form an exception. The last block of a CTR-mode message may contain less than 128 bits (refer to [NIST 800-38A]). For GCM/CCM, the last block of both AAD and message data may contain less than 128 bits (refer to [NIST 800-38D]). The EIP-120t automatically pads or masks misaligned ending data blocks with 0s for GCM, CCM and CBC-MAC. For CTR mode, the remaining data in an unaligned data block is ignored. Note: The AAD / authentication only data is not copied to the output buffer but only used for authentication.

12.7.31 AESTAGOUT_y Register (Offset = 570h + formula) [Reset = 00000000h]

AESTAGOUT_y is shown in Table 12-53.

Return to the Summary Table.

AES Tag Out 0
The tag registers can be accessed via DMA or directly with host reads.
These registers buffer the TAG from the EIP-120t. The registers are shared with the intermediate authentication result registers, but cannot be read until the processing is finished. While processing, a read from these registers returns 0s. If an operation does not return a TAG, reading from these registers returns an IV. If an operation returns a TAG plus an IV and both need to be read by the host, the host must first read the TAG followed by the IV. Reading these in reverse order will return the IV twice.

Offset = 570h + (y * 4h); where y = 0h to 3h

Table 12-53. AESTAGOUT_y Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_TAG	R	0h	AES_TAG[31:0] Bits [31:0] of this register stores the authentication value for the combined and authentication only modes. For a host read operation, these registers contain the last 128-bit TAG output of the EIP-120t the TAG is available until the next context is written. This register will only contain valid data if the TAG is available and when the AESCTL.SAVED_CONTEXT_RDY register is set. During processing or for operations/modes that do not return a TAG, reads from this register return data from the IV register.

12.7.32 AESCCMALNWRD Register (Offset = 5D4h) [Reset = 00000000h]

AESCCMALNWRD is shown in Table 12-54.

Return to the Summary Table.

AES CCM AAD Alignment Data Word
This register needs to be read and stored when an AES-CCM operation is interrupted. This value needs to be restored by writing this register, when resuming that AES-CCM operation in a later session

Table 12-54. AESCCMALNWRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_CCM_ALN_WRD	R/W	0h	This register provides access to an internal location where the AAD align data is stored for AES-CCM operation. Reading this register provides the current AAD alignment data word that is stored internally, and is used internally to concatenate to the next AAD block. Write this register to restore the AAD alignment data word for a resumed AES-CCM operation Note: This register should only be used for continued processing with AES-CCM. Do not write this register for other processing modes

12.7.33 AESBLKCNT0 Register (Offset = 5D8h) [Reset = 00000000h]

AESBLKCNT0 is shown in Table 12-55.

Return to the Summary Table.

AES Block Counter Word 0
This counter keeps track of the number of data blocks during AES-CCM and AES-GCM operations. Reading and writing this counter allows interruption and resuming of long CCM or GCM operations. Note that internally, the block counter is used for AAD data as well as encryption/decryption data

Table 12-55. AESBLKCNT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	AES_BLK_CNT_31_0	R/W	0h	[31:0] of Internal block counter for AES GCM and CCM operations. These bits read the block count value that represents the number of blocks to go. This value is valid with saved_context_ready after a request for an intermediate GCM/CCM digest. Writing these registers will restore the internal block counter to the programmed value. This only needs to be done to prepare the engine to continue processing of an interrupted GCM or CCM operation. Also refer to the get_digest and gcm_ccm_continue bits in AES_CTRL register.

12.7.34 AESBLKCNT1 Register (Offset = 5DCh) [Reset = 00000000h]

AESBLKCNT1 is shown in Table 12-56.

Return to the Summary Table.

AES Block Counter Word 1
This counter keeps track of the number of data blocks during AES-CCM and AES-GCM operations. Reading and writing this counter allows interruption and resuming of long CCM or GCM operations. Note that internally, the block counter is used for AAD data as well as encryption/decryption data

Table 12-56. AESBLKCNT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-0	AES_BLK_CNT_56_32	R/W	0h	[56:32] of Internal block counter for AES GCM and CCM operations. These bits read the block count value that represents the number of blocks to go. This value is valid with saved_context_ready after a request for an intermediate GCM/CCM digest. Writing these registers will restore the internal block counter to the programmed value. This only needs to be done to prepare the engine to continue processing of an interrupted GCM or CCM operation. Also refer to the get_digest and gcm_ccm_continue bits in AES_CTRL register.

12.7.35 HASHDATAIN0 Register (Offset = 600h) [Reset = 00000000h]

HASHDATAIN0 is shown in Table 12-57.

Return to the Summary Table.

HASH Data Input 0
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-57. HASHDATAIN0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[31:0] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.36 HASHDATAIN1 Register (Offset = 604h) [Reset = 00000000h]

HASHDATAIN1 is shown in Table 12-58.

Return to the Summary Table.

HASH Data Input 1
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-58. HASHDATAIN1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[63:32] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.37 HASHDATAIN2 Register (Offset = 608h) [Reset = 00000000h]

HASHDATAIN2 is shown in Table 12-59.

Return to the Summary Table.

HASH Data Input 2
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-59. HASHDATAIN2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[95:64] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.38 HASHDATAIN3 Register (Offset = 60Ch) [Reset = 00000000h]

HASHDATAIN3 is shown in Table 12-60.

Return to the Summary Table.

HASH Data Input 3
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-60. HASHDATAIN3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[127:96] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when the rfd_in bit of the HASH_IO_BUF_CTRL register is high. If the rfd_in bit is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASH_IO_BUF_CTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.39 HASHDATAIN4 Register (Offset = 610h) [Reset = 00000000h]

HASHDATAIN4 is shown in Table 12-61.

Return to the Summary Table.

HASH Data Input 4
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-61. HASHDATAIN4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[159:128] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is '1'. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.40 HASHDATAIN5 Register (Offset = 614h) [Reset = 00000000h]

HASHDATAIN5 is shown in Table 12-62.

Return to the Summary Table.

HASH Data Input 5
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-62. HASHDATAIN5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[191:160] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.41 HASHDATAIN6 Register (Offset = 618h) [Reset = 00000000h]

HASHDATAIN6 is shown in Table 12-63.

Return to the Summary Table.

HASH Data Input 6
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-63. HASHDATAIN6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[223:192] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.42 HASHDATAIN7 Register (Offset = 61Ch) [Reset = 00000000h]

HASHDATAIN7 is shown in Table 12-64.

Return to the Summary Table.

HASH Data Input 7
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-64. HASHDATAIN7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[255:224] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.43 HASHDATAIN8 Register (Offset = 620h) [Reset = 00000000h]

HASHDATAIN8 is shown in Table 12-65.

Return to the Summary Table.

HASH Data Input 8
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-65. HASHDATAIN8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[287:256] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.44 HASHDATAIN9 Register (Offset = 624h) [Reset = 00000000h]

HASHDATAIN9 is shown in Table 12-66.

Return to the Summary Table.

HASH Data Input 9
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-66. HASHDATAIN9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[319:288] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.45 HASHDATAIN10 Register (Offset = 628h) [Reset = 00000000h]

HASHDATAIN10 is shown in Table 12-67.

Return to the Summary Table.

HASH Data Input 10
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-67. HASHDATAIN10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[351:320] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.46 HASHDATAIN11 Register (Offset = 62Ch) [Reset = 00000000h]

HASHDATAIN11 is shown in Table 12-68.

Return to the Summary Table.

HASH Data Input 11
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-68. HASHDATAIN11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[383:352] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.47 HASHDATAIN12 Register (Offset = 630h) [Reset = 00000000h]

HASHDATAIN12 is shown in Table 12-69.

Return to the Summary Table.

HASH Data Input 12
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-69. HASHDATAIN12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[415:384] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.48 HASHDATAIN13 Register (Offset = 634h) [Reset = 00000000h]

HASHDATAIN13 is shown in Table 12-70.

Return to the Summary Table.

HASH Data Input 13
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-70. HASHDATAIN13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[447:416] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.49 HASHDATAIN14 Register (Offset = 638h) [Reset = 00000000h]

HASHDATAIN14 is shown in Table 12-71.

Return to the Summary Table.

HASH Data Input 14
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-71. HASHDATAIN14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[479:448] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.50 HASHDATAIN15 Register (Offset = 63Ch) [Reset = 00000000h]

HASHDATAIN15 is shown in Table 12-72.

Return to the Summary Table.

HASH Data Input 15
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-72. HASHDATAIN15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[511:480] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.51 HASHDATAIN16 Register (Offset = 640h) [Reset = 00000000h]

HASHDATAIN16 is shown in Table 12-73.

Return to the Summary Table.

HASH Data Input 16
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-73. HASHDATAIN16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[543:512] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.52 HASHDATAIN17 Register (Offset = 644h) [Reset = 00000000h]

HASHDATAIN17 is shown in Table 12-74.

Return to the Summary Table.

HASH Data Input 17
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-74. HASHDATAIN17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[575:544] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.53 HASHDATAIN18 Register (Offset = 648h) [Reset = 00000000h]

HASHDATAIN18 is shown in Table 12-75.

Return to the Summary Table.

HASH Data Input 18
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-75. HASHDATAIN18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[607:576] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.54 HASHDATAIN19 Register (Offset = 64Ch) [Reset = 00000000h]

HASHDATAIN19 is shown in Table 12-76.

Return to the Summary Table.

HASH Data Input 19
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-76. HASHDATAIN19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[639:608] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.55 HASHDATAIN20 Register (Offset = 650h) [Reset = 00000000h]

HASHDATAIN20 is shown in Table 12-77.

Return to the Summary Table.

HASH Data Input 20
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-77. HASHDATAIN20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[671:640] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.56 HASHDATAIN21 Register (Offset = 654h) [Reset = 00000000h]

HASHDATAIN21 is shown in Table 12-78.

Return to the Summary Table.

HASH Data Input 21
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-78. HASHDATAIN21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[703:672] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.57 HASHDATAIN22 Register (Offset = 658h) [Reset = 00000000h]

HASHDATAIN22 is shown in Table 12-79.

Return to the Summary Table.

HASH Data Input 22
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-79. HASHDATAIN22 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[735:704] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.58 HASHDATAIN23 Register (Offset = 65Ch) [Reset = 00000000h]

HASHDATAIN23 is shown in Table 12-80.

Return to the Summary Table.

HASH Data Input 23
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-80. HASHDATAIN23 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[767:736] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.59 HASHDATAIN24 Register (Offset = 660h) [Reset = 00000000h]

HASHDATAIN24 is shown in Table 12-81.

Return to the Summary Table.

HASH Data Input 24
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-81. HASHDATAIN24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[799:768] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.60 HASHDATAIN25 Register (Offset = 664h) [Reset = 00000000h]

HASHDATAIN25 is shown in Table 12-82.

Return to the Summary Table.

HASH Data Input 25
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-82. HASHDATAIN25 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[831:800] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.61 HASHDATAIN26 Register (Offset = 668h) [Reset = 00000000h]

HASHDATAIN26 is shown in Table 12-83.

Return to the Summary Table.

HASH Data Input 26
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-83. HASHDATAIN26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[863:832] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.62 HASHDATAIN27 Register (Offset = 66Ch) [Reset = 00000000h]

HASHDATAIN27 is shown in Table 12-84.

Return to the Summary Table.

HASH Data Input 27
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-84. HASHDATAIN27 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[895:864] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.63 HASHDATAIN28 Register (Offset = 670h) [Reset = 00000000h]

HASHDATAIN28 is shown in Table 12-85.

Return to the Summary Table.

HASH Data Input 28
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-85. HASHDATAIN28 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[923:896] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.64 HASHDATAIN29 Register (Offset = 674h) [Reset = 00000000h]

HASHDATAIN29 is shown in Table 12-86.

Return to the Summary Table.

HASH Data Input 29
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-86. HASHDATAIN29 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[959:924] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.65 HASHDATAIN30 Register (Offset = 678h) [Reset = 00000000h]

HASHDATAIN30 is shown in Table 12-87.

Return to the Summary Table.

HASH Data Input 30
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-87. HASHDATAIN30 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[991:960] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.66 HASHDATAIN31 Register (Offset = 67Ch) [Reset = 00000000h]

HASHDATAIN31 is shown in Table 12-88.

Return to the Summary Table.

HASH Data Input 31
The data input registers should be used to provide input data to the hash module through the slave interface.

Table 12-88. HASHDATAIN31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DATA_IN	W	0h	HASH_DATA_IN[1023:992] These registers must be written with the 512-bit or 1024-bit (depending on block size of chosen SHA-2 algorithm) input data. The data lines are connected directly to the data input of the hash module and hence into the engine's internal data buffer. Writing to each of the registers triggers a corresponding 32-bit write enable to the internal buffer. Note: The host may only write the input data buffer when HASHIOBUFCTRL.RFD_IN is 1. If the HASHIOBUFCTRL.RFD_IN is 0, the engine is busy with processing. During processing, it is not allowed to write new input data. For message lengths larger than a block size, multiple blocks of data are written to this input buffer using a handshake through flags of the HASHIOBUFCTRL register. All blocks except the last are required to be 512 bits (or 1024 bits depending on block size) in size. If the last block is not 512 bits (or 1024 bits depending on block size) long, only the least significant bits of data must be written, but they must be padded with 0s to the next 32-bit boundary. Host read operations from these register addresses return 0s.

12.7.67 HASHIOBUFCTRL Register (Offset = 680h) [Reset = 00000004h]

HASHIOBUFCTRL is shown in Table 12-89.

Return to the Summary Table.

HASH Input_Output Buffer Control
This register pair shares a single address location and contains bits that control and monitor the data flow between the host and the hash engine.

Table 12-89. HASHIOBUFCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	PAD_DMA_MESSAGE	R/W	0h	Note: This bit must only be used when data is supplied through the DMA. It should not be used when data is supplied through the slave interface. This bit indicates whether the hash engine has to pad the message, received through the DMA and finalize the hash. When set to 1, the hash engine pads the last block using the programmed length. After padding, the final hash result is calculated. When set to 0, the hash engine treats the last written block as block-size aligned and calculates the intermediate digest. This bit is automatically cleared when the last DMA data block is arrived in the hash engine.
6	GET_DIGEST	R/W	0h	Note: The bit description below is only applicable when data is sent through the slave interface. This bit must be set to 0 when data is received through the DMA. This bit indicates whether the hash engine should provide the hash digest. When provided simultaneously with data_in_av, the hash digest is provided after processing the data that is currently in the HASHDATAINn register. When provided without data_in_av, the current internal digest buffer value is copied to the HASHDIGESTn registers. The host must write a 1 to this bit to make the intermediate hash digest available. Writing 0 to this bit has no effect. This bit is automatically cleared (that is, reads 0) when the hash engine has processed the contents of the HASHDATAINn register. In the period between this bit is set by the host and the actual HASHDATAINn processing, this bit reads 1.
5	PAD_MESSAGE	R/W	0h	Note: The bit description below is only applicable when data is sent through the slave interface. This bit must be set to 0 when data is received through the DMA. This bit indicates that the HASHDATAINn registers hold the last data of the message and hash padding must be applied. The host must write this bit to 1 in order to indicate to the hash engine that the HASHDATAINn register currently holds the last data of the message. When pad_message is set to 1, the hash engine will add padding bits to the data currently in the HASHDATAINn register. When the last message block is smaller than 1024 bits for SHA-512/384 or 512 bits for SHA-256/224, the pad_message bit must be set to ‘1’ together with the data_in_av bit. When the last message block is equal to the block size, pad_message may be set together with data_in_av. In this case, the pad_message bit may also be set after the last data block has been written to the hash engine (so when the rfd_in bit has become ‘1’ again after writing the last data block). Writing 0 to this bit has no effect. This bit is automatically cleared (i.e. reads 0) by the hash engine. This bit reads 1 between the time it was set by the host and the hash engine interpreted its value.
4-3	RESERVED	R/W	0h	Write 0s and ignore on reading

Table 12-89. HASHIOBUFCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	RFD_IN	R/W	1h	Note: The bit description below is only applicable when data is sent through the slave interface. This bit can be ignored when data is received through the DMA. Read-only status of the input buffer of the hash engine. When 1, the input buffer of the hash engine can accept new data the HASHDATAINn registers can safely be populated with new data. When 0, the input buffer of the hash engine is processing the data that is currently in HASHDATAINn writing new data to these registers is not allowed.
1	DATA_IN_AV	R/W	0h	Note: The bit description below is only applicable when data is sent through the slave interface. This bit must be set to 0 when data is received through the DMA. This bit indicates that the HASHDATAINn registers contain new input data for processing. The host must write a 1 to this bit to start processing the data in HASHDATAINn the hash engine will process the new data as soon as it is ready for it (rfd_in bit is 1). Writing 0 to this bit has no effect. This bit is automatically cleared (i.e. reads as 0) when the hash engine starts processing the HASHDATAINn contents. This bit reads 1 between the time it was set by the host and the hash engine actually starts processing the input data block.
0	OUTPUT_FULL	R/W	0h	Indicates that the output buffer registers (HASHDIGESTn) are available for reading by the host. When this bit reads 0, the output buffer registers are released the hash engine is allowed to write new data to it. In this case, the registers should not be read by the host. When this bit reads 1, the hash engine has stored the result of the latest hash operation in the output buffer registers. As long as this bit reads 1, the host may read output buffer registers and the hash engine is prevented from writing new data to the output buffer. After retrieving the hash result data from the output buffer, the host must write a 1 to this bit to clear it. This makes the digest output buffer available for the hash engine to store new hash results. Writing 0 to this bit has no effect. Note: If this bit is asserted (1) no new operation should be started before the digest is retrieved from the hash engine and this bit is cleared (0).

12.7.68 HASHMODE Register (Offset = 684h) [Reset = 00000000h]

HASHMODE is shown in Table 12-90.

Return to the Summary Table.

HASH Mode

Table 12-90. HASHMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	W	0h	Write 0s and ignore on reading
6	SHA384_MODE	W	0h	The host must write this bit with 1 prior to processing a SHA 384 session.
5	SHA512_MODE	W	0h	The host must write this bit with 1 prior to processing a SHA 512 session.
4	SHA224_MODE	W	0h	The host must write this bit with 1 prior to processing a SHA 224 session.
3	SHA256_MODE	W	0h	The host must write this bit with 1 prior to processing a SHA 256 session.
2-1	RESERVED	W	0h	Write 0s and ignore on reading
0	NEW_HASH	W	0h	When set to 1, it indicates that the hash engine must start processing a new hash session. The [HASHDIGESTn.*] registers will automatically be loaded with the initial hash algorithm constants of the selected hash algorithm. When this bit is 0 while the hash processing is started, the initial hash algorithm constants are not loaded in the HASHDIGESTn registers. The hash engine will start processing with the digest that is currently in its internal HASHDIGESTn registers. This bit is automatically cleared when hash processing is started.

12.7.69 HASHINLENL Register (Offset = 688h) [Reset = 00000000h]

HASHINLENL is shown in Table 12-91.

Return to the Summary Table.

HASH Input Length LSB

Table 12-91. HASHINLENL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LENGTH_IN	W	0h	LENGTH_IN[31:0] Message length registers. The content of these registers is used by the hash engine during the message padding phase of the hash session. The data lines of this registers are directly connected to the interface of the hash engine. For a write operation by the host, these registers should be written with the message length in bits. Final hash operations: The total input data length must be programmed for new hash operations that require finalization (padding). The input data must be provided through the slave or DMA interface. Continued hash operations (finalized): For continued hash operations that require finalization, the total message length must be programmed, including the length of previously hashed data that corresponds to the written input digest. Non-final hash operations: For hash operations that do not require finalization (input data length is multiple of 1024-bits for SHA-512, which is SHA-512 data block size. Or multiple of 512-bits for SHA-256, which is SHA-256 data block size), the length field does not need to be programmed since it is not used by the operation If the message length in bits is below (2^{32}-1), then only this register needs to be written. The hardware automatically sets HASH_LENGTH_IN_H to 0s in this case. The host may write the length register at any time during the hash session when the HASHIOBUFCTRL.RFD_IN is high. The length register must be written before the last data of the active hash session is written into the hash engine. host read operations from these register locations will return 0s. Note: When getting data from DMA, this register must be programmed before DMA is programmed to start.

12.7.70 HASHINLENH Register (Offset = 68Ch) [Reset = 00000000h]

HASHINLENH is shown in Table 12-92.

Return to the Summary Table.

HASH Input Length MSB

Table 12-92. HASHINLENH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LENGTH_IN	W	0h	LENGTH_IN[63:32] Message length registers. The content of these registers is used by the hash engine during the message padding phase of the hash session. The data lines of this registers are directly connected to the interface of the hash engine. For a write operation by the host, these registers should be written with the message length in bits. Final hash operations: The total input data length must be programmed for new hash operations that require finalization (padding). The input data must be provided through the slave or DMA interface. Continued hash operations (finalized): For continued hash operations that require finalization, the total message length must be programmed, including the length of previously hashed data that corresponds to the written input digest. Non-final hash operations: For hash operations that do not require finalization (input data length is multiple of 1024-bits for SHA-512, which is SHA-512 data block size. Or multiple of 512-bits for SHA-256, which is SHA-256 data block size), the length field does not need to be programmed since it is not used by the operation. If the message length in bits is below (2^{32}-1), then only HASHINLENL needs to be written. The hardware automatically sets HASH_LENGTH_IN_H to 0s in this case. The host may write the length register at any time during the hash session when the HASHIOBUFCTRL.RFD_IN is high. The length register must be written before the last data of the active hash session is written into the hash engine. host read operations from these register locations will return 0s. Note: When getting data from DMA, this register must be programmed before DMA is programmed to start.

12.7.71 HASHDIGESTA Register (Offset = 6C0h) [Reset = 00000000h]

HASHDIGESTA is shown in Table 12-93.

Return to the Summary Table.

HASH Digest A
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-93. HASHDIGESTA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[31:0] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.72 HASHDIGESTB Register (Offset = 6C4h) [Reset = 00000000h]

HASHDIGESTB is shown in Table 12-94.

Return to the Summary Table.

HASH Digest B
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-94. HASHDIGESTB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[63:32] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.73 HASHDIGESTC Register (Offset = 6C8h) [Reset = 00000000h]

HASHDIGESTC is shown in Table 12-95.

Return to the Summary Table.

HASH Digest C
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-95. HASHDIGESTC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[95:64] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.74 HASHDIGESTD Register (Offset = 6CCh) [Reset = 00000000h]

HASHDIGESTD is shown in Table 12-96.

Return to the Summary Table.

HASH Digest D
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-96. HASHDIGESTD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[127:96] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.75 HASHDIGESTE Register (Offset = 6D0h) [Reset = 00000000h]

HASHDIGESTE is shown in Table 12-97.

Return to the Summary Table.

HASH Digest E
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-97. HASHDIGESTE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[159:128] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.76 HASHDIGESTF Register (Offset = 6D4h) [Reset = 00000000h]

HASHDIGESTF is shown in Table 12-98.

Return to the Summary Table.

HASH Digest F
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-98. HASHDIGESTF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[191:160] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.77 HASHDIGESTG Register (Offset = 6D8h) [Reset = 00000000h]

HASHDIGESTG is shown in Table 12-99.

Return to the Summary Table.

HASH Digest G
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-99. HASHDIGESTG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[223:192] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.78 HASHDIGESTH Register (Offset = 6DCh) [Reset = 00000000h]

HASHDIGESTH is shown in Table 12-100.

Return to the Summary Table.

HASH Digest H
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-100. HASHDIGESTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[255:224] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.79 HASHDIGESTI Register (Offset = 6E0h) [Reset = 00000000h]

HASHDIGESTI is shown in Table 12-101.

Return to the Summary Table.

HASH Digest I
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-101. HASHDIGESTI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[287:256] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.80 HASHDIGESTJ Register (Offset = 6E4h) [Reset = 00000000h]

HASHDIGESTJ is shown in Table 12-102.

Return to the Summary Table.

HASH Digest J
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-102. HASHDIGESTJ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[319:288] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.81 HASHDIGESTK Register (Offset = 6E8h) [Reset = 00000000h]

HASHDIGESTK is shown in Table 12-103.

Return to the Summary Table.

HASH Digest K
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-103. HASHDIGESTK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[351:320] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.82 HASHDIGESTL Register (Offset = 6ECh) [Reset = 00000000h]

HASHDIGESTL is shown in Table 12-104.

Return to the Summary Table.

HASH Digest L
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-104. HASHDIGESTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[383:352] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.83 HASHDIGESTM Register (Offset = 6F0h) [Reset = 00000000h]

HASHDIGESTM is shown in Table 12-105.

Return to the Summary Table.

HASH Digest M
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-105. HASHDIGESTM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[415:384] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.84 HASHDIGESTN Register (Offset = 6F4h) [Reset = 00000000h]

HASHDIGESTN is shown in Table 12-106.

Return to the Summary Table.

HASH Digest N
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-106. HASHDIGESTN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[447:416] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.85 HASHDIGESTO Register (Offset = 6F8h) [Reset = 00000000h]

HASHDIGESTO is shown in Table 12-107.

Return to the Summary Table.

HASH Digest 0
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-107. HASHDIGESTO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[479:448] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.86 HASHDIGESTP Register (Offset = 6FCh) [Reset = 00000000h]

HASHDIGESTP is shown in Table 12-108.

Return to the Summary Table.

HASH Digest P
The hash digest registers consist of sixteen 32-bit registers, named HASHDIGESTA to HASHDIGESTP. After processing a message, the output digest can be read from these registers. These registers can be written with an intermediate hash result for continued hash operations.

Table 12-108. HASHDIGESTP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	HASH_DIGEST	R/W	0h	HASH_DIGEST[511:480] Hash digest registers Write operation: Continued hash: These registers should be written with the context data, before the start of a resumed hash session (the HASHMODE.NEW_HASH bit is 0 when starting a hash session). New hash: When initiating a new hash session (theHASHMODE.NEW_HASH bit is 1), the internal digest registers are automatically set to the SHA-256 algorithm constant and these register should not be written. Reading from these registers provides the intermediate hash result (non-final hash operation) or the final hash result (final hash operation) after data processing.

12.7.87 ALGSEL Register (Offset = 700h) [Reset = 00000000h]

ALGSEL is shown in Table 12-109.

Return to the Summary Table.

Algorithm Select
This algorithm selection register configures the internal destination of the DMA controller.

Table 12-109. ALGSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	TAG	R/W	0h	If this bit is cleared to 0, the DMA operation involves only data. If this bit is set, the DMA operation includes a TAG (Authentication Result / Digest). For SHA-2 operation, a DMA must be set up for both input data and TAG. For any other selected module, setting this bit only allows a DMA that reads the TAG. No data allowed to be transferred to or from the selected module via the DMA.
30-4	RESERVED	R	0h	Reserved
3	HASH_SHA_512	R/W	0h	If set to one, selects the Hash engine as destination for the DMA. The maximum transfer size to DMA engine is set to 128 bytes for reading and 64 bytes for writing (the latter is only applicable if the hash result is written out via DMA).
2	HASH_SHA_256	R/W	0h	If set to one, selects the hash engine in 256B mode as destination for the DMA The maximum transfer size to DMA engine is set to 64 bytes for reading and 32 bytes for writing (the latter is only applicable if the hash result is written out through the DMA).
1	AES	R/W	0h	If set to one, selects the AES engine as source/destination for the DMA The read and write maximum transfer size to the DMA engine is set to 16 bytes.
0	KEY_STORE	R/W	0h	If set to one, selects the Key Store as destination for the DMA The maximum transfer size to DMA engine is set to 32 bytes (however transfers of 16, 24 and 32 bytes are allowed)

12.7.88 DMAPROTCTL Register (Offset = 704h) [Reset = 00000000h]

DMAPROTCTL is shown in Table 12-110.

Return to the Summary Table.

DMA Protection Control
Master PROT privileged access enable
This register enables the second bit (bit [1]) of the AHB HPROT bus of the AHB master interface when a read action of key(s) is performed on the AHB master interface for writing keys into the store module.

Table 12-110. DMAPROTCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	PROT_EN	R/W	0h	Select AHB transfer protection control for DMA transfers using the key store area as destination. 0 : transfers use 'USER' type access. 1 : transfers use 'PRIVILEGED' type access.

12.7.89 SWRESET Register (Offset = 740h) [Reset = 00000000h]

SWRESET is shown in Table 12-111.

Return to the Summary Table.

Software Reset

Table 12-111. SWRESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SW_RESET	R/W	0h	If this bit is set to 1, the following modules are reset: - Master control internal state is reset. That includes interrupt, error status register, and result available interrupt generation FSM. - Key store module state is reset. That includes clearing the written area flags therefore, the keys must be reloaded to the key store module. Writing 0 has no effect. The bit is self cleared after executing the reset.

12.7.90 IRQTYPE Register (Offset = 780h) [Reset = 00000000h]

IRQTYPE is shown in Table 12-112.

Return to the Summary Table.

Control Interrupt Configuration

Table 12-112. IRQTYPE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	LEVEL	R/W	0h	If this bit is 0, the interrupt output is a pulse. If this bit is set to 1, the interrupt is a level interrupt that must be cleared by writing the interrupt clear register. This bit is applicable for both interrupt output signals.

12.7.91 IRQEN Register (Offset = 784h) [Reset = 00000000h]

IRQEN is shown in Table 12-113.

Return to the Summary Table.

Control Interrupt Enable

Table 12-113. IRQEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	DMA_IN_DONE	R/W	0h	If this bit is set to 0, the DMA input done interrupt disabled If this bit is set to 1, the DMA input done interrupt enabled.
0	RESULT_AVAIL	R/W	0h	If this bit is set to 0, the Result Available interrupt is disabled If this bit is set to 1, the Result Available interrupt is enabled.

12.7.92 IRQCLR Register (Offset = 788h) [Reset = 00000000h]

IRQCLR is shown in Table 12-114.

Return to the Summary Table.

Control Interrupt Clear

Table 12-114. IRQCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DMA_BUS_ERR	W	0h	If 1 is written to this bit, the DMA bus error status is cleared. Writing 0 has no effect.
30	KEY_ST_WR_ERR	W	0h	If 1 is written to this bit, the key store write error status is cleared. Writing 0 has no effect.
29	KEY_ST_RD_ERR	W	0h	If 1 is written to this bit, the key store read error status is cleared. Writing 0 has no effect.
28-2	RESERVED	R	0h	Reserved
1	DMA_IN_DONE	W	0h	If 1 is written to this bit, the DMA in done interrupt status is cleared. Writing 0 has no effect. Note that clearing an interrupt makes sense only if the interrupt output is programmed as level (refer to IRQTYPE).
0	RESULT_AVAIL	W	0h	If 1 is written to this bit, the result available interrupt status is cleared. Writing 0 has no effect. Note that clearing an interrupt makes sense only if the interrupt output is programmed as level (refer to IRQTYPE).

12.7.93 IRQSET Register (Offset = 78Ch) [Reset = 00000000h]

IRQSET is shown in Table 12-115.

Return to the Summary Table.

Control Interrupt Set

Table 12-115. IRQSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	DMA_IN_DONE	W	0h	If 1 is written to this bit, the DMA data in done interrupt is set. Writing 0 has no effect. If the interrupt configuration register is programmed to pulse, clearing the DMA data in done interrupt is not needed. If it is programmed to level, clearing the interrupt output should be done by writing the interrupt clear register (IRQCLR.DMA_IN_DONE).
0	RESULT_AVAIL	W	0h	If 1 is written to this bit, the result available interrupt is set Writing 0 has no effect. If the interrupt configuration register is programmed to pulse, clearing the result available interrupt is not needed. If it is programmed to level, clearing the interrupt should be done by writing the interrupt clear register (IRQCLR.RESULT_AVAIL).

12.7.94 IRQSTAT Register (Offset = 790h) [Reset = 00000000h]

IRQSTAT is shown in Table 12-116.

Return to the Summary Table.

Control Interrupt Status

Table 12-116. IRQSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DMA_BUS_ERR	R	0h	This bit is set when a DMA bus error is detected during a DMA operation. The value of this register is held until it is cleared through the IRQCLR.DMA_BUS_ERR Note: This error is asserted if an error is detected on the AHB master interface during a DMA operation.
30	KEY_ST_WR_ERR	R	0h	This bit is set when a write error is detected during the DMA write operation to the key store memory. The value of this register is held until it is cleared through the IRQCLR.KEY_ST_WR_ERR register. Note: This error is asserted if a DMA operation does not cover a full key area or more areas are written than expected.
29	KEY_ST_RD_ERR	R	0h	This bit is set when a read error is detected during the read of a key from the key store, while copying it to the AES core. The value of this register is held until it is cleared through the IRQCLR.KEY_ST_RD_ERR register. Note: This error is asserted if a key location is selected in the key store that is not available.
28-2	RESERVED	R	0h	Reserved
1	DMA_IN_DONE	R	0h	This read only bit returns the actual DMA data in done interrupt status
0	RESULT_AVAIL	R	0h	This read only bit returns the actual result available interrupt status

12.7.95 HWVER Register (Offset = 7FCh) [Reset = 93028778h]

HWVER is shown in Table 12-117.

Return to the Summary Table.

Hardware Version

Table 12-117. HWVER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	HW_MAJOR_VER	R	3h	Major version number
23-20	HW_MINOR_VER	R	0h	Minor version number
19-16	HW_PATCH_LVL	R	2h	Patch level Starts at 0 at first delivery of this version
15-8	VER_NUM_COMPL	R	87h	These bits simply contain the complement of bits [7:0] (0x87), used by a driver to ascertain that the EIP-120t register is indeed read.
7-0	VER_NUM	R	78h	These bits encode the EIP number for the EIP-120t, this field contains the value 120 (decimal) or 0x78.

Chapter 13

PKA Engine

The PKA engine of the CC13x4x10 and CC26x4x10 device platform is a public key acceleration (PKA) module with Chinese Remainder Theorem (CRT) support. It operates as a co-processor, offloading public key operations from the processing core.

13.1 Introduction 1014
13.2 Functional Description 1015
13.3 PKA Engine Performance 1026
13.4 PKA Registers 1029

13.1 Introduction

The PKA engine provides the following basic operations:

- Large vector addition, subtraction, and combined addition and subtraction
- Large vector shift left or right
- Large vector multiplication and division (with or without quotient)
- Large vector compare and copy

The PKA engine provides the following complex operations:

- Large vector unsigned value modular exponentiation
- Large vector unsigned value modular exponentiation using the CRT method with precalculated Q inverse vector
- Modular inversion: Given A and M, calculate B such that [(A × B) MOD M] = 1
- ECC point addition/doubling on elliptic curve $y^2 = x^3 + ax + b \pmod p$, where
  - p is a prime number
  - a and b are input values to the operation

  Adding two identical points automatically performs point doubling.
- ECC point multiplication on elliptic curve $y^2 = x^3 + ax + b \pmod p$, where:
  - p is a prime number
  - a and b are input values to the operation

  A version of the *Montgomery ladder* algorithm is used to provide side channel attack resistance.

The PKA engine supports hardware zeroization logic for memories that contain sensitive data. The hardware zeroization logic operates independently of the reset of the PKA IP and is controlled by PRCM:SECDMACLKGR.PKA_ZERIOZE_RESET_N.

Note

The system must make sure that zeroization logic is not enabled during an active operation of the module. Therefore, TI recommends activating the zeroization logic after putting the module into hardware reset mode.

For power efficiency, the module uses dynamically controlled clock switches that are activated only when required. Some of the RAMs are clocked with these dynamically controlled clocks and before enabling the zeroization logic, the system must enable these clocks.

The PKA RAM is 2 KB, which is sufficient to support ECC operations on curves of size 521 bits or smaller.

13.2 Functional Description

13.2.1 Module Architecture

The PKA engine (see Figure 13-1) contains the following internal modules:

- PKCP module: Can perform a suite of large number (vector) operations typically encountered in public key cryptography applications. Both arguments and results are stored PKA RAM (a memory block shared between the PKA engine and its host). The PKCP in this PKA Engine is a 16-bit PKCP.
- Sequencer module: Controlling modular exponentiation, elliptic curve cryptography, and modular inversion operations on large numbers in PKA RAM. One of the main tasks of the sequencer module is to hide that most of these operations are completed with numbers in Montgomery form. This module requires a program ROM or RAM as code store.
- Register interface: Used to control the PKCP and Sequencer modules.

The PKA engine is a fully synchronous design with a single clock and has a single active low asynchronous reset input. The figure below shows a block diagram of the PKA engine. Note that the Large Number Multiplier and Exponentiator (LNME) and LNME FIFO RAM shown in the diagram are not present in this microcontroller.

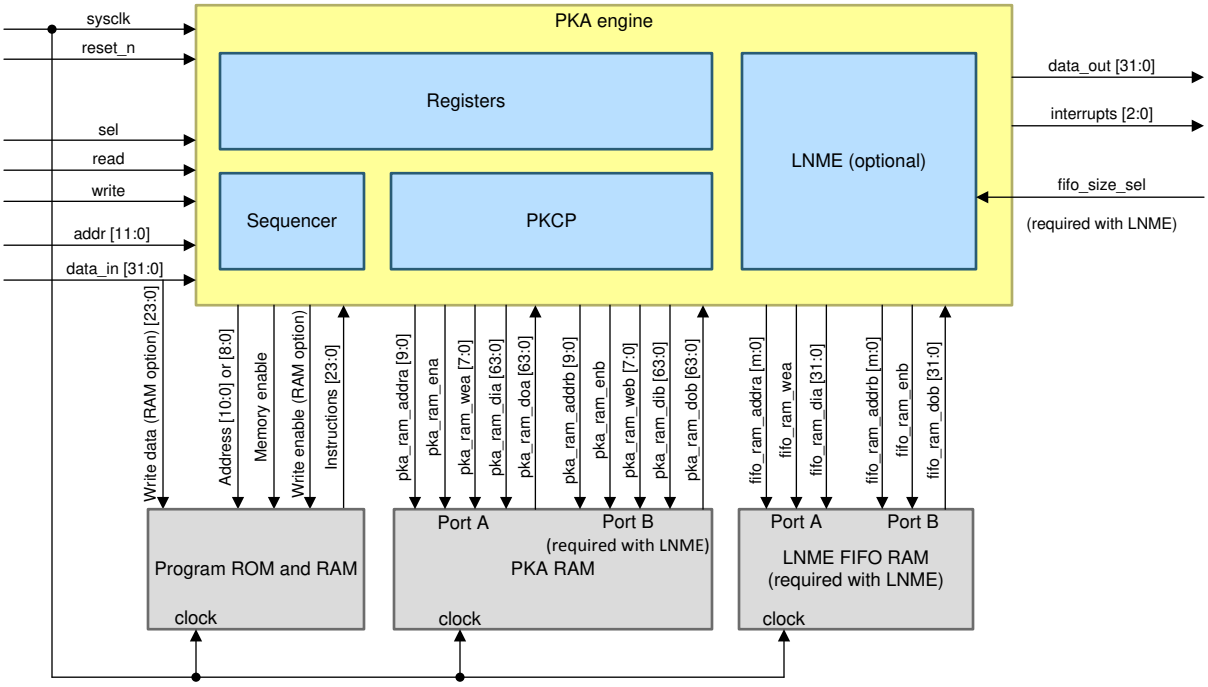


Figure 13-1. PKA Engine

13.2.2 PKA RAM

The vectors (large numbers) that are the input and output of the PKA operations are stored in the PKA RAM. Each vector consists of a sequence of 32-bit words that are stored in a contiguous block of memory with the LSB at the lowest address of that memory block. All input and output vectors must start at an even-numbered 32-bit word address.

The PKA RAM is 2 KB (256 × 64 bits). This size was chosen based on the required performance and vector length requirements. ECC curves up to 521 bits in size can be handled with the chosen RAM size.

Note

The PKA RAM is always 32-bit accessible from the host interface (all input and output vectors must start at an even-numbered 32-bit word address; that is, they must be aligned to an 8-byte boundary in PKA RAM).

13.2.3 PKCP Operations

Table 13-1 lists the arguments and results for each PKCP operation.

Table 13-1. PKCP Operations(1)

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
Multiply	A × B → C	Multiplicand	Multiplier	Product	N/A
Add	A + B → C	Addend	Addend	Sum	N/A
Subtract	A - B → C	Minuend	Subtrahend	Difference	N/A
AddSub	A + C - B → D	Addend	Subtrahend	Addend	Result
Right Shift	A >> Shift → C	Input	N/A	Result	N/A
Left Shift	A << Shift → C	Input	N/A	Result	N/A
Divide	A mod B → C, A div B → D	Dividend	Divisor	Remainder	Quotient
Modulo	A mod B → C	Dividend	Divisor	Remainder	N/A
Compare	A = B, A < B, A > B	Input1	Input2	NA	N/A
Copy	A → C	Input	N/A	Result	N/A

(1) N/A = not available

To obtain correct results, the input vector must meet the requirements presented in Table 13-2.

Note

- Input restrictions are not checked by the PKCP.
- A_Len and B_Len indicate the size of vectors A and B in 32-bit words.
- Max_{Len} equals 128 (32-bit) words (that is, the standard maximum vector size is 4096 bits).

Table 13-2. Restrictions on Input Vectors for PKCP Operations

Operational Restrictions	
Function	**Requirements**
Multiply	0 < A_Len B_Len ≤ Max_Len
Add	0 < A_Len B_Len ≤ Max_Len
Subtract	0 < A_Len B_Len ≤ Max_Len Result must be positive (A ≥ B)
AddSub	0 < A_Len ≤ Max_Len (B and C operands have A_Len as length, B_Len is ignored) Result must be positive [(A + C) ≥ B]
Right shift	0 < A_Len ≤ Max_Len
Left shift	0 < A_Len ≤ Max_Len
Divide, modulo	1 < B_Len ≤ A_Len ≤ Max_Len Most significant 32-bit word of B operand cannot be 0
Compare	0 < A_Len ≤ Max_Len (B operand has A_Len as length, B_Len is ignored)
Copy	0 < A_Len ≤ Max_Len

The host is responsible for allocating a block of contiguous memory in PKA RAM for the result vector. Table 13-3 indicates how much memory should be allocated for the result vectors.

Table 13-3. Result Vector Memory Allocation

Function	Result Vector	Result Vector Length (in 32-Bit Words)
Multiply	C	A_Len + B_Len + 6 (the six scratchpad words should be discarded)
Add	C	Max(A_Len, B_Len) + 1
Subtract	C	Max(A_Len, B_Len)
AddSub	D	A_Len + 1
Right shift	C	A_Len
Left shift	C	A_Len + 1 (when shift value is nonzero)
		A_Len (when shift value is zero)
Divide	C	Remainder → B_Len + 1 (one scratchpad word should be discarded)
	D	Quotient → A_Len – B_Len + 1
Modulo	C	Remainder → B_Len + 1 (one scratchpad word should be discarded)
Compare	None	Updates the PKA_COMPARE register
Copy	C	A_Len

Input vectors for an operation are always allowed to overlap in memory (partially or completely). Table 13-4 lists restrictions for the overlap of output and input vectors of the operations.

Table 13-4. PKCP Result Vector and Vector Overlap Restrictions

Function	Result Vector	Restriction
Multiply	C	No overlap with A or B vectors allowed.
Add, subtract	C	May overlap with A and (or) B vectors if the start address of the C vector does not lie above the start address of the vector or vectors that it overlaps.
AddSub	D	May overlap with A, B, and (or) C vectors if the start address of the D vector does not lie above the start address of the vector or vectors that it overlaps.
Right shift, left shift	C	May overlap with A vector if the start address of the C vector does not lie above the start address of the A vector.
Divide	C	No overlap with A, B, or D vectors allowed.
	D	No overlap with A, B, or C vectors allowed.
Modulo	C	No overlap with A or B vectors allowed.
Compare	None	Compare does not write a result vector.
Copy	C	Restrictions for Copy are the same as right and left shift restrictions; copy of a vector to a lower address is always allowed, even if a source and destination overlap(1).

(1) The *Copy* operation can be used to fill memory by breaking the overlap restrictions, but this requires TWO initial (32-bit) words to be set up:

- To zero a block of memory
- Set A vector pointer to the block start
- Set C vector pointer two words higher
- Set A vector length to the block length minus two (words)
- Fill the first two words of the block with constant zero and perform a PKCP Copy operation to zero the remainder of the block.

13.2.4 Sequencer Operations

13.2.4.1 Modular Exponentiation Operations

The sequencer controls modular exponentiation operations. Table 13-5 lists a summary of Modular Exponentiation (ExpMod) operations.

Table 13-5. ExpMod Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ExpMod-ACT2	$C^A \bmod B \rightarrow D$	Exponent, length = A_Len	Modulus, length = B_Len	Base, length = B_Len	Result and workspace
ExpMod-ACT4					
ExpMod-variable					
ExpMod-CRT	— (See the computation steps in the following list.)	Exp P followed by Exp Q at the next highest even-word address(1), both A_Len long	Mod P + buffer word followed by Mod Q at next highest even-word address(2), both B_Len long	Q inverse, length = B_Len	Input, result (both 2 × B_Len long), and workspace

(1) If A_Len is even, Exp Q follows Exp P immediately—if A_Len is odd, there is one empty word between Exp Q and Exp P.
(2) If B_Len is even, there are two empty words between Mod P and Mod Q—if B_Len is odd, there is one empty (buffer) word between Mod Q and Mod P. Note that the words following Mod P and Mod Q can be zeroed by *Sequencer* firmware.

The ExpMod-CRT operation performs the following computation steps:

1. $X \leftarrow (\text{Input mod Mod P})^{\text{Exp P}} \bmod \text{Mod P}$
2. $Y \leftarrow (\text{Input mod Mod Q})^{\text{Exp Q}} \bmod \text{Mod Q}$
3. $Z \leftarrow (\{[(X - Y) \bmod \text{Mod P}] \times \text{Q inverse}\} \bmod \text{Mod P}) \times \text{Mod Q}$
4. $\text{Result} \leftarrow Y + Z$

The ExpMod-ACT2, ExpMod-ACT4, and ExpMod-variable functions implement the same mathematical operation but with a differently sized table with precalculated *odd-numbered powers*. The ExpMod-ACT2 function uses a table with two entries, and the ExpMod-ACT4 function uses a table with eight entries. The ACT4 version provides better performance but requires more memory.

ExpMod-variable and ExpMod-CRT allow a variable amount (from 1 up to and including 16) of odd powers to be selected through the register that is normally used to specify the number of bits to shift for shift operations.

For a user of the PKA engine, the exponentiation functions appear to be extensions of the set of PKCP functions. Input and result vectors are passed the same as for basic PKCP operations.

Table 13-6 lists the restrictions on the input vector for ExpMod operations.

Table 13-6. Operational Restrictions

Function	Requirements
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	1. $0 < A\_Len \leq Max\_Len$ 2. $1 < B\_Len \leq Max\_Len$ 3. Modulus B must be odd-numbered (that is, the LSB must be 1). 4. Modulus B > 2^{32} 5. Base C < Modulus B 6. Vectors B and C must be followed by an empty 32-bit *buffer* word.

Table 13-6. Operational Restrictions (continued)

Function	Requirements
ExpMod-CRT	1. 0 < A_Len ≤ Max_Len 2. 1 < B_Len ≤ Max_Len 3. Mod P and Mod Q must be odd-numbered (that is, the LSBs must be 1). 4. Mod P > Mod Q > 2^{32} (1) 5. Mod P and Mod Q must be coprime (their GCD must be 1). 6. 0 < Exp P < (Mod P – 1) 7. 0 < Exp Q < (Mod Q – 1) 8. (Q inverse × Mod Q) = 1 (modulo Mod P) 9. Input < (Mod P × Mod Q) 10. Mod P and Mod Q must be followed by an empty 32-bit *buffer* word.

(1) mod P must be larger than Mod Q

Table 13-7 lists the required scratchpad sizes for the exponentiation operations. The M_{Len} in the table is the real modulus length (for Mod P in an ExpMod-CRT operation, for modulus B in other operations) in 32-bit words (that is, without trailing zero words at the end). If the last word of the modulus vector as given is nonzero, then M_{Len} equals B_{Len}.

Table 13-7. Result Vector and Scratchpad Area Memory Allocation (Starting at PKA_DPTR)

Function	Scratchpad Area Size (32-Bit Words)(1)
ExpMod-ACT2	5 × (M_Len + 2)
ExpMod-ACT4	11 × (M_Len + 2)
ExpMod-variable	(odd-numbered powers + 3) × (M_Len + 2)
ExpMod-CRT	(odd-numbered powers + 3) × (M_Len + 2) + [M_Len + 2 – (M_Len MOD 2)]

(1) The result vector is M_Len or 2 × M_Len 32-bit words long.

Table 13-8 lists the result vector and input vector overlap restrictions.

Table 13-8. Overlap Restrictions of Result and Input Vectors

Function	Result Vector	Restrictions
ExpMod-ACT2	D	Scratchpad area starting at D cannot overlap with any of the other vectors, except that Base C can be colocated with result vector D to save space (that is, PKA_CPTR = PKA_CPTR is allowed).
ExpMod-ACT4		
ExpMod-variable		
ExpMod-CRT	D	Scratchpad area starting at D cannot overlap with any of the other vectors; this is also the location of the main input vector (with length 2 × B_Len).

For exponentiation operations, the minimum size of the PKA RAM depends on the maximum modulus length and the number of odd-numbered powers. In addition, a fixed number of bytes is required as scratchpad for the sequencer firmware during execution of the exponentiation; this scratchpad must be at the end of the PKA RAM.

- The PKA RAM must be sized so the most often performed exponentiations can be done with four odd-numbered powers.
- A 2 KB PKA RAM suffices when the most frequently used modulus lengths are 1 KB and 2 KB operations, which do not need to run fast (4 KB operations are not possible with a 2 KB RAM size).

Table 13-9 depicts the RAM sizes required for exponentiation operations.

Table 13-9. Required RAM Sizes

Modulus Size (Non-CRT)	One Odd-Numbered Power		> One Odd-Numbered Power(1)
	PKA_CPTR = PKA_DPTR	PKA_CPTR ≠ PKA_DPTR	
1024 bits	808 bytes	944 bytes	+ 136 bytes per extra odd-numbered power
2048 bits	1576 bytes	1840 bytes	+ 256 bytes per extra odd-numbered power

Table 13-9. Required RAM Sizes (continued)

Modulus Size (Non-CRT)	One Odd-Numbered Power		> One Odd-Numbered Power[1]
	PKA_CPTR = PKA_DPTR	PKA_CPTR ≠ PKA_DPTR	
4096 bits[2]	3112 bytes	3632 bytes	+ 520 bytes per extra odd-numbered power
Scratchpad	+ 34 bytes (fixed, at end of PKA RAM)		
CRT Moduli	**One Odd-Numbered Power**		**> One odd-numbered Power [1]**
2 × 512 bits	696 bytes		+ 72 bytes per extra odd-numbered power
2 × 1024 bits	1336 bytes		+ 136 bytes per extra odd-numbered power
2 × 2048 bits [2]	2616 bytes		+ 264 bytes per extra odd-numbered power
Scratchpad	+ 72 bytes (fixed, at end of PKA RAM)		

(1) Add to one odd-numbered power sizes
(2) Not possible on this device because the PKA RAM is 2 KB

Table 13-10 lists the maximum number of odd-numbered powers that can be used for different standard PKA RAM sizes and PKA engine types (non-CRT operations using PKA_CPTR = PKA_DPTR).

Table 13-10. Maximum Number of Odd-Numbered Powers

Operation	Modulus and Exponent Sizes	Maximum Number of Odd-Numbered Powers for PKA RAM Sizes
		2 KB
Non-CRT	1024 bits	9
	2048 bits	2
	4096 bits	Not Possible
CRT	2 × 512 bits	16
	2 × 1024 bits	5
	2 × 2048 bits	Not Possible

Table 13-11 lists example PKA RAM vector allocations for modular exponentiation operations with and without using CRT. The free space start address is the first free byte following the vector workspace. The sequencer execution scratchpad of 34 bytes (non-CRT) or 72 bytes (using CRT) must fit between the free space start address and the end of the PKA RAM.

Note

The non-CRT operations use PKA_CPTR = PKA_DPTR to save space.

Table 13-11. Example PKA RAM Vector Allocations

Operation	(sub-)Vector	Start Address Byte Offset		Size (Words)	Buffer (Words)
non-CRT (ALENGTH = 0x040 BLENGTH = 0x040 four odd-numbered powers)	Exponent	0x000	(APTR = 0x000)	64	0
	Modulus	0x100	(BPTR = 0x040)	64	2
	Base	0x208	(CPTR = 0x082)	64	2
	Result	0x208	(DPTR = 0x082)	64	2
	Vector workspace	0x208	(= result)	7 × (64 + 2) = 462	0
	Free space	0x940	(2368 bytes used)	–	–

Table 13-11. Example PKA RAM Vector Allocations (continued)

Operation	(sub-)Vector	Start Address Byte Offset		Size (Words)	Buffer (Words)
using CRT (ALENGTH = 0x020 BLENGTH = 0x020 four odd-numbered powers)	Exp P	0x000	(APTR = 0x000)	32	0
	Exp Q	0x080		32	0
	Mod P	0x100	(BPTR = 0x040)	32	2
	Mod Q	0x188		32	2
	Q inverse	0x210	(CPTR = 0x084)	32	0
	Input, result	0x290	(DPTR = 0x0A4)	64	0
	Vector workspace	0x290	(= result)	7 × (32 + 2) + 32 + 2 – 0 = 272	0
	Free space	0x6D0	(1744 bytes used)	–	–

13.2.4.2 Modular Inversion Operation

The sequencer controls modular inversion operation. Table 13-12 lists the function that is an extension of basic PKCP functions with the following exceptions:

- Vector D addresses the result and a workspace.
- The PKA_SHIFT register field is used to return information on the operation result.

Table 13-12. Extension of Basic PKCP Functions

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ModInv	A^{-1} mod B → D	NumToInvert length = A_Len	Modulus length = B_Len	Not used	Result and workspace

Table 13-13 lists the PKA_SHIFT result values.

Table 13-13. PKA_SHIFT Result Values

Function	PKA Shift Register Field Value at Conclusion
ModInv	0 → success; Vector D holds the result 7 → no inverse exists [GCD (A, B) ≠ 1 (that is, A and B have common factors); result undefined 31 → error; modulus even-numbered, result undefined Other values are reserved.

Table 13-14 and Table 13-15 list the restrictions on the input and result vectors for ModInv operations.

Table 13-14. Operational Restrictions

Function	Requirements
ModInv	0 < A_Len ≤ Max_Len 0 < B_Len ≤ Max_Len Modulus B must be odd-numbered (that is, the LSB must be 1). Modulus B may not have a value of 1 (result is undefined, no error indicated). The highest word of the modulus vector (indicated by B_Len) may not be 0.

Table 13-15. Overlap Restrictions of Result and Input Vectors

Function	Result Vector	Restrictions
ModInv	D	Scratchpad area starting at D may not overlap with any other vectors.

Table 13-16 lists the required scratchpad sizes for the ModInv operation.

Table 13-16. Result and Vector Scratchpad Area Memory Allocation(1)

Function	Scratchpad Area Size (in 32-Bit Words), Result Vector is B_Len 32-Bit-Words Long
ModInv	5 × [M + ε (M)], with M = Max(A_Len, B_Len)

(1) Both starting at PKA_DPTR) ε (*n*) = 2 + (*n* MOD 2); that is, 2 (for *n* even-numbered) or 3 (for *n* odd-numbered.

The ModInv operation requires the modulus to be odd-numbered; this appears to make the operation useless with RSA key generation, where the private key exponent *d* is derived from a chosen public exponent *e* as in Equation 2:

$$d = \text{ModInv}(e, \varphi) \quad (2)$$

where:

- $\varphi = (p - 1) \times (q - 1)$
- *p* and *q* are boh prime

Note

In Equation 2, φ is even-numbered. However, because *e* must be odd-numbered (or no inverse exists), *d* can be calculated as in Equation 3.

$$d = 1 + \{\varphi \times [e - \text{ModInv}(\varphi, e)]\} / e \quad (3)$$

With four basic PKCP operations, ModInv can be used to find inverse values if the modulus is even.

Modular inversion can be performed with a modular exponentiation using the modulus value minus 2 as exponent, provided the modulus value is prime; this is because (under the constraint that M is prime):

- $(A^M) \bmod M = A \rightarrow$
- $(A^{M-1}) \bmod M = 1 \rightarrow$
- $(A^{M-2}) \bmod M = A^{-1} \pmod M$

The modulus values for the ECC curves supported by this PKA engine must be prime so this method can be used in ECDSA operations.

13.2.4.3 ECC Operations

The sequencer also controls ECC operations (for a summary, see Table 13-17).

Table 13-17. ECC Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ECC-ADD	Point addition/doubling(1) on elliptic curve: $y^2 = x^3 + ax + b \pmod p$ pntA + pntC → pntD	pntA.x followed(2) by pntA.y both B_Len long (A_Len is not used)	Curve parameter *p* followed(2) by *a* (*b* is not required) all B_Len long	pntC.x followed(2) by pntC.y both B_Len long	Result (that is, pntD.x followed(2) by pntD.y and workspace)
ECC-MUL	Point multiplication on elliptic curve: $y^2 = x^3 + ax + b \pmod p$ $k \times$ pntC → pntD	Scalar *k* A_Len long	Curve parameter *p* followed(2) by *a* and *b* all B_Len long	pntC.x followed(2) by pntC.y both B_Len long	Result (that is, pntD.x followed(2) by pntD.y and workspace)

(1) If pntA = pntC, a point doubling operation is performed automatically.
(2) All input components must be located on a 64-bit boundary and must have a **ε** extra buffer words (of 32 bits each) after their most significant word.
ε must be 3 (B_Len odd) or 2 (B_Len even). Each result component (that is, pntD.x, pntD.y) will also be followed by **ε** buffer (zero) words.

For users of the PKA engine, the functions in Table 13-17 appear to be extensions of the set of PKCP functions described in Section 13.2.3 with the following exceptions:

- Input and result vectors can be composite (that is, they may consist of two or three equal-sized subvectors).
- Vector D addresses the result and a workspace.

The PKA_SHIFT register returns information on the result of the operation.

Table 13-18 lists the PKA_SHIFT result values.

Table 13-18. PKA_SHIFT Result Values

Function	PKA_SHIFT Register Field Value at Conclusion
ECC-ADD	0 → success; vector D holds the result point. 7 → result is *point-at-infinity*; vector D result point is undefined. 31 → error (*p* is not odd-numbered, too short, and so on); vector D result point is undefined. Other values are reserved.
ECC-MUL	

Table 13-19 lists the operational restrictions.

Table 13-19. Operational Restrictions

Function	Requirements
ECC-ADD	1 < B_Len ≤ 24 (maximum vector length is 768 bits) Modulus *p* must be a prime > 2^{63}. Effective modulus size (in bits) must be a multiple of 32.(1) The highest word of the modulus vector (as indicated by B_Len) may not be 0. *a* < *p* and *b* < *p* pntA and pntC must be on the curve (this is not checked). Neither pntA nor pntC can be the *point-at-infinity* (although ECC-ADD can return this point as a result).
ECC-MUL	0 < A_Len ≤ 24 (maximum vector length is 768 bits) 1 < B_Len ≤ 24 (maximum vector length is 768 bits) Modulus *p* must be a prime > 2^{63}. Effective modulus size (in bits) must be a multiple of 32.(1) The highest word of the modulus vector (as indicated by B _Len) may not be 0. *a* < *p* and *b* < *p* pntC must be on the curve (this is not checked). pntC cannot be the *point-at-infinity* (although ECC-MUL can return this point as a result).

(1) Depending on the engine type, a few modulus lengths that do not adhere to this rule will lead to incorrect results—the *standard* modulus lengths of 112 and 521 bits will work on all engines, so these lengths are the exceptions to this rule.

Table 13-20 and Table 13-21 list the overlap restrictions of the input vector and the memory allocation details of the scratchpad area, respectively.

Table 13-20. Overlap Restrictions of Input and Output Vectors

Function	Result Vector	Restrictions
ECC-ADD ECC-MUL	D	Scratchpad area starting at D may not overlap with any other vectors.

Table 13-21. Memory Allocation of Result Vector and Scratchpad Area(1) (2)

Function	Scratchpad Area Size Result Vector is 2 × (B_Len + ε (B_Len)) 32-Bit Words Long
ECC-ADD	2 × L + 5 × M where: • L = B_Len + ε (B_Len) • M = B_Len + 1 + ε (B_Len + 1)
ECC-MUL	18 × L + Max(8, L) where: • L = B_Len + ε (B_Len)

(1) Both result vectors and scratchpad memory allocation start at PKA_DPTR.
(2) ε (*n*) = 2 + (*n* MOD 2, that is 2 (for *n* even or 3 (for *n* odd).

During the execution of an ECC-ADD or ECC-MUL operation, the last 72 bytes of the PKA RAM are used as a general scratchpad for the program execution of the sequencer. These areas must not overlap with any of the input vectors or the D vector scratchpad area during execution.

Table 13-22 lists example PKA RAM vector allocations for ECC point multiplication operations. The *free space* start address is the first free byte following the vector scratchpad (the sequencer execution scratchpad of 72 bytes must fit between this address and the end of PKA RAM); because of this, a 521-bit ECC point multiplication cannot be performed with a 2 KB PKA RAM.

Table 13-22. PKA RAM Vector Allocations

Modulus Length	(sub-) Vector	Start Address Byte Offset		Size (Words)	Buffer (Words)
192 bits (= 6 words, ALENGTH = 0x006, BLENGTH = 0x006)	Scalar *k*	0x000	(APTR = 0x000)	6	0
	p	0x018	(BPTR = 0x006)	6	2
	a	0x038		6	2
	b	0x058		6	2
	pntC.x (base)	0x078	(CPTR = 0x01E)	6	2
	pntC.y (base)	0x098		6	2
	pntD.x (result)	0x0B8	(DPTR = 0x02E)	6	2
	pntD.y (result)	0x0D8		6	0
	Vector scratchpad	0x0B8	(= pntD.x)	(18 × 8) + 8 = 152	0
	Free space	0x318	(792 bytes used)	–	–
384 bits (= 12 words, ALENGTH = 0x00C, BLENGTH = 0x00C)	Scalar *k*	0x000	(APTR = 0x000)	12	0
	p	0x030	(BPTR = 0x00C)	12	2
	a	0x068		12	2
	b	0x0A0		12	2
	pntC.x (base)	0x0D8	(CPTR = 0x036)	12	2
	pntC.y (base)	0x110		12	2
	pntD.x (result)	0x148	(DPTR = 0x052)	12	2
	pntD.y (result)	0x180		12	0
	Vector scratchpad	0x148	(= pntD.x)	(18 × 14) + 14 = 266	0
	Free space	0x570	(1392 bytes used)	–	–
521 bits (= 17 words, ALENGTH = 0x011, BLENGTH = 0x011)	Scalar *k*	0x000	(APTR = 0x000)	17	1 (to align *p*)
	p	0x048	(BPTR = 0x012)	17	3
	a	0x098		17	3
	b	0x0E8		17	3
	pntC.x (base)	0x138	(CPTR = 0x04E)	17	3
	pntC.y (base)	0x188		17	3
	pntD.x (result)	0x1D8	(DPTR = 0x076)	17	3
	pntD.y (result)	0x228		17	0
	Vector scratchpad	0x1D8	(= pntD.x)	(18 × 20) + 20 = 380	0
	Free space	0x7C8	(1992 bytes used)	–	–

13.2.5 Operation Sequence

Figure 13-2, Figure 13-3, and Figure 13-4 show several operation sequences that can be used to operate the PKA engine. Interface bus transactions and the behavior of the main interrupt output (bit [1] of the interrupt output bus) are described in a stylized way.

The left side of the sequence shown in Figure 13-2 shows the start-up behavior. The interrupt is inactive (low) during module reset and active (high) within 10 module clock cycles after starting the sequencer program. In the case where a program ROM is used, the sequencer is started immediately when the module reset is released. For program RAM-equipped engines, the sequencer firmware must first be loaded, and then the sequencer must be taken out of reset (write *0b* to bit [31] of the PKA_SEQ_CTRL register) to start the sequencer program.

The right side of Figure 13-2 shows the normal operation sequence. A normal operation sequence begins by writing input vectors in the PKA data RAM and vector pointers and length values to the PKA engine control registers (can be completed in any order). The operation is started with a write to the PKA_FUNCTION register, which results in dropping the main interrupt output inactive within two clock cycles after setting the RUN bit. When the PKA engine has finished executing the requested operation, the main interrupt is activated again and the result status can be read from the status registers (if needed). The result vector or vectors can be read from the PKA data RAM.

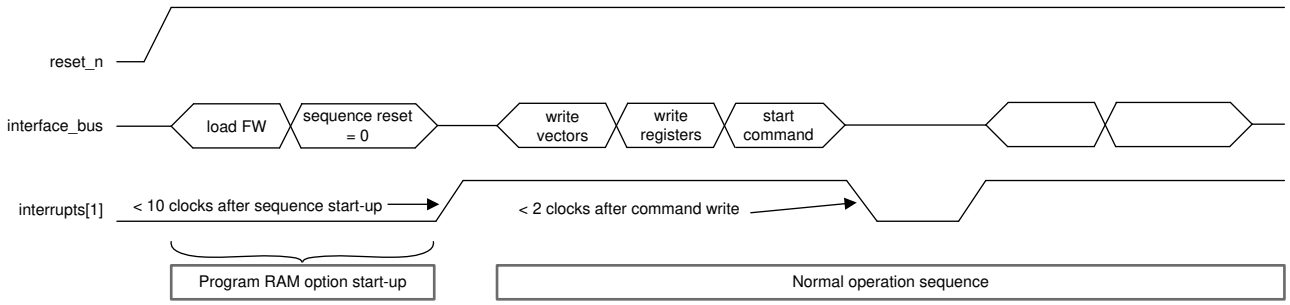


Figure 13-2. Operation Sequence

Figure 13-3 shows a more optimized, interleaved operation sequence. When enough PKA data RAM is available, separate areas in the RAM can be used for interleaving the input vector writes and result vector reads. The input vectors (for the second operation) are written while the first operation is in execution. Writing of the pointer and length registers and actual starting of the second command is done before the result vectors of the first command are read from the PKA data RAM. Writing the input vectors for a third operation can be done immediately following the reading of the first result, all while the second operation is in execution.

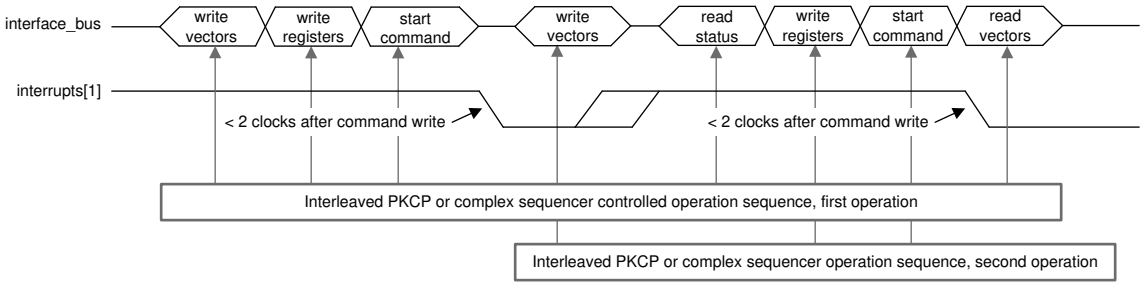


Figure 13-3. Interleaved Operation Sequence

Figure 13-4 shows a highly optimized, basic PKCP operation sequence. For basic PKCP operations, the vector pointer and length registers are double buffered; they may be written while an operation is in progress. This ability allows a more optimized interleaving of bus accesses with writing the vectors and pointer and length register for the second operation while the first operation is in execution. When the interrupt activation occurs, only the command of the second operation is required for start-up.

Figure 13-4 also shows that the status of the first operation is read after the second command is started; to make sure this status is not changed by an early completion of the second operation, the second operation must be started with the Stall Result bit (bit [24] of the PKA_FUNCTION register) written as 1b. After reading the status, the Stall Result bit must be reset to allow updating of the status. If the first operation has no status to check, setting the Stall Result bit is not required.

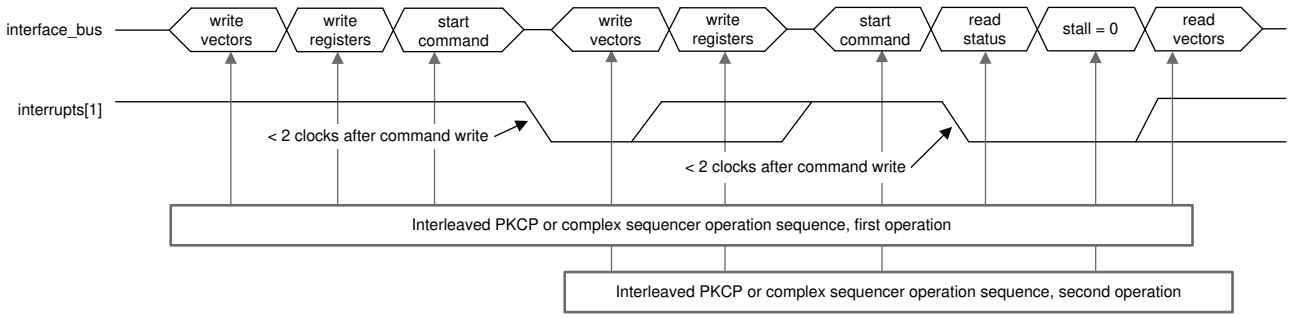


Figure 13-4. Basic PKCP Operation Sequence

13.3 PKA Engine Performance

13.3.1 Basic Operations Performance

Table 13-23 lists information on the number of clocks required to perform basic PKCP operations.

Table 13-23. Clocks for PKCP Operations

Operation (Vector Lengths in Bits)	Number of Clocks	Scaling to Other Vector Sizes
1K + 1K addition	195	Max (A_Len, B_Len)
1K – 1K subtract	194	Max (A_Len, B_Len)
1K × 1K multiply	4210	A_Len × B_Len
2K / 1K divide or modulo	19241(1)	(A_Len – B_Len) × B _Len
1K + 1K – 1K add or subtract	259	A_Len
1K shift left or right	133	A_Len
1K copy	128	A_Len
1K = 1K compare	133(2)	A_Len

(1) Slight variability in timing due to the possibility of correction cycles being required.
(2) Data dependent—the maximum time given is required only when vectors have the same value or differ only in the least significant 16- or 32-bit word.

13.3.2 ExpMod Performance

Using one odd-numbered power as the baseline performance at 100%:

- Two odd-numbered powers (used by ExpMod-ACT2) delivers approximately 112% performance.
- Four odd-numbered powers delivers approximately 121% performance.
- Eight odd-numbered powers (used by ExpMod-ACT4) delivers approximately 125% performance.

The performance figures assume that preprocessing takes negligible time (true for longer exponent lengths) and the use of random exponent vectors with 50% of the bits in the vector set to '1'. Using more than eight odd-numbered powers does not provide significant speed gains and is not recommended.

Table 13-24. ExpMod Performance Values

Exponent Length in Bits	Modulus Length in Bits	Number of Odd Powers	Number of Clocks	Operations per Second at 48 MHz
160	512	1	697 280	68
160	512	4	634 414	75
256	256	1	352 203	136
256	256	4	304 490	157

13.3.3 Modular Inversion Performance

Modular inversion uses the PKCP engine for its calculation. Performance depends on the type of PKCP engine and is slightly dependent on the data values used (a few percent variability is expected). Table 13-25 lists average performance values over five different simulations.

Table 13-25. Performance Values of Five Simulations

Vector Length for Modulus and Input		Number of Clocks	ops/sec at 48 MHz
128	bits	29 226	1 624
256		75 483	635
512		220 647	217
1024		719 587	66
2048		2 594 516	18

13.3.4 ECC Operation Performance

ECC-ADD uses the PKCP engine for calculations. Performance depends on the type of PKCP engine and is slightly dependent on the data values used (a few percent variability is expected). Table 13-26 lists the performance values.

Table 13-26. ECC Operation Performance Values

Vector Length for Modulus and Input			Number of Clocks	ops/sec at 48 MHz
128	Point addition	bits	33 714	1 422
	Point doubling		34 749	1 381
160	Point addition		43 266	1 109
	Point doubling		46 246	1 037
192	Point addition		56 403	850
	Point doubling		58 229	824
224	Point addition		68 969	695
	Point doubling		70 169	684
256	Point addition		84 140	570
	Point doubling		82 935	578
320	Point addition		110 963	432
	Point doubling		115 581	415
384	Point addition		145 655	329
	Point doubling		154 698	310
512	Point addition		236 331	203
	Point doubling		241 818	198
521	Point addition		243 138	197
	Point doubling		257 409	186

The ECC-MUL operation uses the PKCP. Because of the usage of a version of the ECC ladder algorithm, performance is independent of the scalar multiplication vector *k*. A slight variability (less than 2 percent) is expected because of the three modular inversions performed at the end of the algorithm (see Table 13-27).

Table 13-27. ECC-MUL Operation Performance Values

Vector Length in Bits for All Input Values	Number of Clocks	ops/sec at 48 MHz
128	1.36×10^6	35.2
160	1.90×10^6	25.2
192	3.00×10^6	16.0
224	4.67×10^6	10.2
256	5.51×10^6	8.7
320	10.3×10^6	4.6
384	15.5×10^6	3
512	33.2×10^6	1.4
521	35.5×10^6	1.3

13.4 PKA Registers

Table 13-28 lists the memory-mapped registers for the PKA registers. All register offset addresses not listed in Table 13-28 should be considered as reserved locations and the register contents should not be modified.

Table 13-28. PKA Registers

Offset	Acronym	Register Name	Section
0h	APTR	PKA Vector A Address	Section 13.4.1
4h	BPTR	PKA Vector B Address	Section 13.4.2
8h	CPTR	PKA Vector C Address	Section 13.4.3
Ch	DPTR	PKA Vector D Address	Section 13.4.4
10h	ALENGTH	PKA Vector A Length	Section 13.4.5
14h	BLENGTH	PKA Vector B Length	Section 13.4.6
18h	SHIFT	PKA Bit Shift Value	Section 13.4.7
1Ch	FUNCTION	PKA Function	Section 13.4.8
20h	COMPARE	PKA compare result	Section 13.4.9
24h	MSW	PKA most-significant-word of result vector	Section 13.4.10
28h	DIVMSW	PKA most-significant-word of divide remainder	Section 13.4.11
C8h	SEQCTRL	PKA sequencer control and status register	Section 13.4.12
F4h	OPTIONS	PKA hardware options register	Section 13.4.13
F8h	FWREV	PKA firmware revision and capabilities register	Section 13.4.14
FCh	HWREV	PKA hardware revision register	Section 13.4.15

Complex bit access types are encoded to fit into small table cells. Table 13-29 shows the codes that are used for access types in this section.

Table 13-29. PKA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

13.4.1 APTR Register (Offset = 0h) [Reset = 00000000h]

APTR is shown in Table 13-30.

Return to the Summary Table.

PKA Vector A Address
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-30. APTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R/W	0h	Set to zero on write, ignore on read
10-0	APTR	R/W	0h	This register specifies the location of vector A within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

13.4.2 BPTR Register (Offset = 4h) [Reset = 00000000h]

BPTR is shown in Table 13-31.

Return to the Summary Table.

PKA Vector B Address
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-31. BPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R/W	0h	Set to zero on write, ignore on read
10-0	BPTR	R/W	0h	This register specifies the location of vector B within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

13.4.3 CPTR Register (Offset = 8h) [Reset = 00000000h]

CPTR is shown in Table 13-32.

Return to the Summary Table.

PKA Vector C Address
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-32. CPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R/W	0h	Set to zero on write, ignore on read
10-0	CPTR	R/W	0h	This register specifies the location of vector C within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

13.4.4 DPTR Register (Offset = Ch) [Reset = 00000000h]

DPTR is shown in Table 13-33.

Return to the Summary Table.

PKA Vector D Address
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-33. DPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R/W	0h	Set to zero on write, ignore on read
10-0	DPTR	R/W	0h	This register specifies the location of vector D within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

13.4.5 ALENGTH Register (Offset = 10h) [Reset = 00000000h]

ALENGTH is shown in Table 13-34.

Return to the Summary Table.

PKA Vector A Length
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-34. ALENGTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R/W	0h	Set to zero on write, ignore on read
8-0	ALENGTH	R/W	0h	This register specifies the length (in 32-bit words) of Vector A.

13.4.6 BLENGTH Register (Offset = 14h) [Reset = 00000000h]

BLENGTH is shown in Table 13-35.

Return to the Summary Table.

PKA Vector B Length
During execution of basic PKCP operations, this register is double buffered and can be written with a new value for the next operation
when not written, the value remains intact. During the execution of sequencer-controlled complex operations, this register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

Table 13-35. BLENGTH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R/W	0h	Set to zero on write, ignore on read
8-0	BLENGTH	R/W	0h	This register specifies the length (in 32-bit words) of Vector B.

13.4.7 SHIFT Register (Offset = 18h) [Reset = 00000000h]

SHIFT is shown in Table 13-36.

Return to the Summary Table.

PKA Bit Shift Value
For basic PKCP operations, modifying the contents of this register is made impossible while the operation is being performed. For the ExpMod-variable and ExpMod-CRT operations, this register is used to indicate the number of odd powers to use (directly as a value in the range 1-16). For the ModInv and ECC operations, this register is used to hold a completion code.

Table 13-36. SHIFT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	0h	Set to zero on write, ignore on read
4-0	NUM_BITS_TO_SHIFT	R/W	0h	This register specifies the number of bits to shift the input vector (in the range 0-31) during a Rshift or Lshift operation.

13.4.8 FUNCTION Register (Offset = 1Ch) [Reset = 00008000h]

FUNCTION is shown in Table 13-37.

Return to the Summary Table.

PKA Function
This register contains the control bits to start basic PKCP as well as complex sequencer operations. The run bit can be used to poll for the completion of the operation. Modifying bits [11:0] is made impossible during the execution of a basic PKCP operation.
During the execution of sequencer-controlled complex operations, this register is modified, the run and stall result bits are set to zero at the conclusion, but other bits are undefined.
NOTE: Continuously reading this register to poll the run bit is not allowed when executing complex sequencer operations (the sequencer cannot access the PKCP when this is done). Leave at least one sysclk cycle between poll operations.

Table 13-37. FUNCTION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	Set to zero on write, ignore on read
24	STALL_RESULT	R/W	0h	When written with a 1b, updating of the COMPARE bit, MSW and DIVMSW registers, as well as resetting the run bit is stalled beyond the point that a running operation is actually finished. Use this to allow software enough time to read results from a previous operation when the newly started operation is known to take only a short amount of time. If a result is waiting, the result registers is updated and the run bit is reset in the clock cycle following writing the stall result bit back to 0b. The Stall result function may only be used for basic PKCP operations.
23-16	RESERVED	R/W	0h	Set to zero on write, ignore on read
15	RUN	R/W	1h	The host sets this bit to instruct the PKA module to begin processing the basic PKCP or complex sequencer operation. This bit is reset low automatically when the operation is complete. After a reset, the run bit is always set to 1b. Depending on the option, program ROM or program RAM, the following applies: Program ROM - The first sequencer instruction sets the bit to 0b. This is done immediately after the hardware reset is released. Program RAM - The sequencer must set the bit to 0b. As a valid firmware may not have been loaded, the sequencer is held in software reset after the hardware reset is released (the SEQCTRL.RESET bit is set to 1b). After the FW image is loaded and the Reset bit is cleared, the sequencer starts to execute the FW. The first instruction clears the run bit. In both cases a few clock cycles are needed before the first instruction is executed and the run bit state has been propagated.
14-12	SEQUENCER_OPERATIONS	R/W	0h	These bits select the complex sequencer operation to perform: 0x0: None 0x1: ExpMod-CRT 0x2: ECmontMUL 0x3: ECC-ADD (if available in firmware, otherwise reserved) 0x4: ExpMod-ACT2 0x5: ECC-MUL (if available in firmware, otherwise reserved) 0x6: ExpMod-variable 0x7: ModInv (if available in firmware, otherwise reserved) The encoding of these operations is determined by sequencer firmware.
11	COPY	R/W	0h	Perform copy operation
10	COMPARE	R/W	0h	Perform compare operation
9	MODULO	R/W	0h	Perform modulo operation
8	DIVIDE	R/W	0h	Perform divide operation
7	LSHIFT	R/W	0h	Perform left shift operation
6	RSHIFT	R/W	0h	Perform right shift operation

Table 13-37. FUNCTION Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	SUBTRACT	R/W	0h	Perform subtract operation
4	ADD	R/W	0h	Perform add operation
3	MS_ONE	R/W	0h	Loads the location of the Most Significant one bit within the result word indicated in the MSW register into bits [4:0] of the DIVMSW.MSW_ADDRESS register - can only be used with basic PKCP operations, except for Divide, Modulo and Compare.
2	RESERVED	R/W	0h	Set to zero on write, ignore on read
1	ADDSUB	R/W	0h	Perform combined add/subtract operation
0	MULTIPLY	R/W	0h	Perform multiply operation

13.4.9 COMPARE Register (Offset = 20h) [Reset = 00000001h]

COMPARE is shown in Table 13-38.

Return to the Summary Table.

PKA compare result
This register provides the result of a basic PKCP compare operation. It is updated when the FUNCTION.RUN bit is reset at the end of that operation.
Status after a complex sequencer operation is unknown

Table 13-38. COMPARE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Ignore on read
2	A_GREATER_THAN_B	R	0h	Vector_A is greater than Vector_B
1	A_LESS_THAN_B	R	0h	Vector_A is less than Vector_B
0	A_EQUALS_B	R	1h	Vector_A is equal to Vector_B

13.4.10 MSW Register (Offset = 24h) [Reset = 00008000h]

MSW is shown in Table 13-39.

Return to the Summary Table.

PKA most-significant-word of result vector
This register indicates the (word) address in the PKA RAM where the most significant nonzero 32-bit word of the result is stored. Should be ignored for modulo operations. For basic PKCP operations, this register is updated FUNCTION.RUN bit is reset at the end of the operation. For the complex-sequencer controlled operations, updating of the final value matching the actual result is done near the end of the operation
note that the result is only meaningful if no errors were detected and that for ECC operations, this register will provide information for the x-coordinate of the result point only.

Table 13-39. MSW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Ignore on read
15	RESULT_IS_ZERO	R	1h	The result vector is all zeroes, ignore the address returned in bits [10:0]
14-11	RESERVED	R	0h	Ignore on read
10-0	MSW_ADDRESS	R	0h	Address of the most-significant nonzero 32-bit word of the result vector in PKA RAM

13.4.11 DIVMSW Register (Offset = 28h) [Reset = 00008000h]

DIVMSW is shown in Table 13-40.

Return to the Summary Table.

PKA most-significant-word of divide remainder
This register indicates the (32-bit word) address in the PKA RAM where the most significant nonzero 32-bit word of the remainder result for the basic divide and modulo operations is stored. Bits [4:0] are loaded with the bit number of the most-significant nonzero bit in the most-significant nonzero word when MS one control bit is set. For divide, modulo, and MS one reporting, this register is updated when FUNCTION.RUN bit is reset at the end of the operation. For the complex sequencer controlled operations, updating of bits [4:0] of this register with the most-significant bit location of the actual result is done near the end of the operation. The result is meaningful only if no errors were detected and that for ECC operations
this register provides information for the x-coordinate of the result point only.

Table 13-40. DIVMSW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Ignore on read
15	RESULT_IS_ZERO	R	1h	The result vector is all zeroes, ignore the address returned in bits [10:0]
14-11	RESERVED	R	0h	Ignore on read
10-0	MSW_ADDRESS	R	0h	Address of the most significant nonzero 32-bit word of the remainder result vector in PKA RAM

13.4.12 SEQCTRL Register (Offset = C8h) [Reset = 00000100h]

SEQCTRL is shown in Table 13-41.

Return to the Summary Table.

PKA sequencer control and status register
The sequencer is interfaced with the outside world through a single control and status register. With the exception of bit [31], the actual use of bits in the separate sub-fields of this register is determined by the sequencer firmware. This register need only be accessed when the sequencer program is stored in RAM. The reset value of the RESET bit depends upon the option chosen for sequencer program storage. NOTE: For Agama the sequencer firmware is executed from ROM.

Table 13-41. SEQCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESET	R/W	0h	Option program ROM: Reset value = 0. Read/Write, reset value 0b (ZERO). Writing 1b resets the sequencer, write to 0b to restart operations again. As the reset value is 0b, the sequencer will automatically start operations executing from program ROM. This bit should always be written with zero and ignored when reading this register. Option Program RAM: Reset value =1. Read/Write, reset value 1b (ONE). When 1b, the sequencer is held in a reset state and the PKA_PROGRAM area is accessible for loading the sequencer program (while the PKA_DATA_RAM is inaccessible), write to 0b to (re)start sequencer operations and disable PKA_PROGRAM area accessibility (also enables the PKA_DATA_RAM accesses). Resetting the sequencer (in order to load other firmware) should only be done when the PKA Engine is not performing any operations (i.e. the FUNCTION.RUN bit should be zero).
30-16	RESERVED	R/W	0h	Set to zero on write, ignore on read
15-8	SEQUENCER_STAT	R	1h	These read-only bits can be used by the sequencer to communicate status to the outside world. Bit [8] is also used as sequencer interrupt, with the complement of this bit ORed into the FUNCTION.RUN bit. This field should always be written with zeroes and ignored when reading this register.
7-0	SW_CONTROL_STAT	R/W	0h	These bits can be used by software to trigger sequencer operations. External logic can set these bits by writing 1b, cannot reset them by writing 0b. The sequencer can reset these bits by writing 0b, cannot set them by writing 1b. Setting the FUNCTION.RUN bit together with a nonzero sequencer operations field automatically sets bit [0] here. This field should always be written with zeroes and ignored when reading this register.

13.4.13 OPTIONS Register (Offset = F4h) [Reset = 00000020h]

OPTIONS is shown in Table 13-42.

Return to the Summary Table.

PKA hardware options register
This register provides the host with a means to determine the hardware configuration implemented in this PKA engine, focused on options that have an effect on software interacting with the module.

Table 13-42. OPTIONS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Ignore on read
11	INT_MASKING	R	0h	Interrupt Masking 0x0: indicates that the main interrupt output (bit [1] of the interrupts output bus) is the direct complement of the run bit in the PKA_CONTROL register 0x1 : indicates that interrupt masking logic is present for this output. Note: Reset value is undefined
10-8	PROTECTION_OPTION	R	0h	Protection Option 0x0: indicates no additional protection against side channel attacks 0x1: indicates the SCAP option 0x2: Reserved 0x3: indicates the PROT option Note: Reset value is undefined
7	PROGRAM_RAM	R	0h	Program RAM 0x1: indicates sequencer program storage in RAM 0x0: indicates sequencer program storage in ROM. Note: Reset value is undefined
6-5	SEQUENCER_ CONFIGURATION	R	1h	Sequencer Configuration 0x0: Reserved 0x1 : Indicates a standard sequencer 0x2: Reserved 0x3: Reserved
4-2	RESERVED	R	0h	Ignore on read
1-0	PKCP_CONFIGURATION	R	0h	PKCP Configuration 0x0 : Reserved 0x1 : Indicates a PKCP with a 16x16 multiplier 0x2: indicates a PKCP with a 32x32 multiplier 0x3 : Reserved Note: Reset value is undefined.

13.4.14 FWREV Register (Offset = F8h) [Reset = 21500000h]

FWREV is shown in Table 13-43.

Return to the Summary Table.

PKA firmware revision and capabilities register
This register allows the host access to the internal firmware revision number of the PKA Engine for software driver matching and diagnostic purposes. This register also contains a field that encodes the capabilities of the embedded firmware.
This register is written by the firmware within a few clock cycles after starting up that firmware. The hardware reset value is zero, indicating that the information has not been written yet.

Table 13-43. FWREV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	FW_CAPABILITIES	R	2h	Firmware Capabilities 4-bit binary encoding for the functionality implemented in the firmware. 0x0: indicates basic ModExp with/without CRT. 0x1: adds Modular Inversion 0x2: value 2 adds Modular Inversion and ECC operations. 0x3-0xF : Reserved.
27-24	MAJOR_FW_REVISION	R	1h	4-bit binary encoding of the major firmware revision number
23-20	MINOR_FW_REVISION	R	5h	4-bit binary encoding of the minor firmware revision number
19-16	FW_PATCH_LEVEL	R	0h	4-bit binary encoding of the firmware patch level, initial release will carry value zero Patches are used to remove bugs without changing the functionality or interface of a module.
15-0	RESERVED	R	0h	Ignore on read

13.4.15 HWREV Register (Offset = FCh) [Reset = 0151E31Ch]

HWREV is shown in Table 13-44.

Return to the Summary Table.

PKA hardware revision register
This register allows the host access to the hardware revision number of the PKA engine for software driver matching and diagnostic purposes. It is always located at the highest address in the access space of the module and contains an encoding of the EIP number (with its complement as signature) for recognition of the hardware module.

Table 13-44. HWREV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Ignore on read
27-24	MAJOR_HW_REVISION	R	1h	4-bit binary encoding of the major hardware revision number
23-20	MINOR_HW_REVISION	R	5h	4-bit binary encoding of the minor hardware revision number
19-16	HW_PATCH_LEVEL	R	1h	4-bit binary encoding of the hardware patch level, initial release will carry value zero Patches are used to remove bugs without changing the functionality or interface of a module.
15-8	COMPLEMENT_OF_BASIC_ EIP_NUMBER	R	E3h	Bit-by-bit logic complement of bits [7:0], EIP-28 gives 0xE3
7-0	BASIC_EIP_NUMBER	R	1Ch	8-bit binary encoding of the EIP number, EIP-28 gives 0x1C

This page intentionally left blank.

Chapter 14

True Random Number Generator (TRNG)

The true random number generator (TRNG) module provides a true, non-deterministic noise source for the purpose of generating keys, initialization vectors (IVs), and other random number requirements. The TRNG is built on 24 ring oscillators that create unpredictable output. To obtain cryptographically secure random data, the output is then post-processed by being fed to a complex nonlinear combinatorial circuit. Typical applications might be (but are not limited to) the following:

- Generation of cryptographic key material
- Generation of initialization vectors
- Generation of cookies and nonces
- Statistical sampling
- Retry timers in communication protocols
- Noise generation

The TRNG registers are all mapped to Secure address space.

14.1 Introduction 1048
14.2 Block Diagram 1048
14.3 TRNG Software Reset 1049
14.4 Interrupt Requests 1049
14.5 TRNG Operation Description 1050
14.6 TRNG Low-Level Programming Guide 1052
14.7 TRNG Registers 1055

14.1 Introduction

The TRNG has the following features:

- The TRNG is based on 24 ring oscillators (shot noise) that create entropy. To generate this entropy, the system needs a minimum of 2^8 system clock cycles (for reference) to produce the first random output. Then the TRNG takes a minimum of 2^6 system clock cycles to produce each subsequent 64-bit random number.
- Startup time and entropy regeneration time can be controlled between 2^8 and 2^{24} sampling clock cycles, and entropy regeneration time can be controlled between 2^6 and 2^{24} sampling clock cycles to adapt entropy accumulation time to basic entropy generation rate. Entropy regeneration time can be tailored in a trade-off between speed of random number generation and amount of entropy in each of those random numbers.
- The TRNG architecture is based on linear-feedback shift register (LFSR) in association with a nonlinear entropic hasher.
- The random numbers are accessible to the applications in a 64-bit read-only register. When the register is read, the TRNG immediately generates a new value, which is then shifted into the output register when ready.
- If the ready value is not read within a maximum time-out window, the TRNG is set to idle mode.
- The TRNG provides a built-in self-test that checks the number of consecutive bits sampled to provide the statistical robustness required by FIPS 140. System alarms are generated based on feedback from this test.
- The internal power-saving mode is built to carefully manage the entropy previously generated.
- The interrupt channel allows the transfer of 32-bit data blocks.

14.2 Block Diagram

The TRNG core uses dual-shot noise generators that create unpredictable jittering output when asynchronously sampled by the system clock provided to the TRNG. The outputs from the shot noise generators feed a complex nonlinear combinatorial circuit (mixer) that produces the final TRNG output (see Figure 14-1).

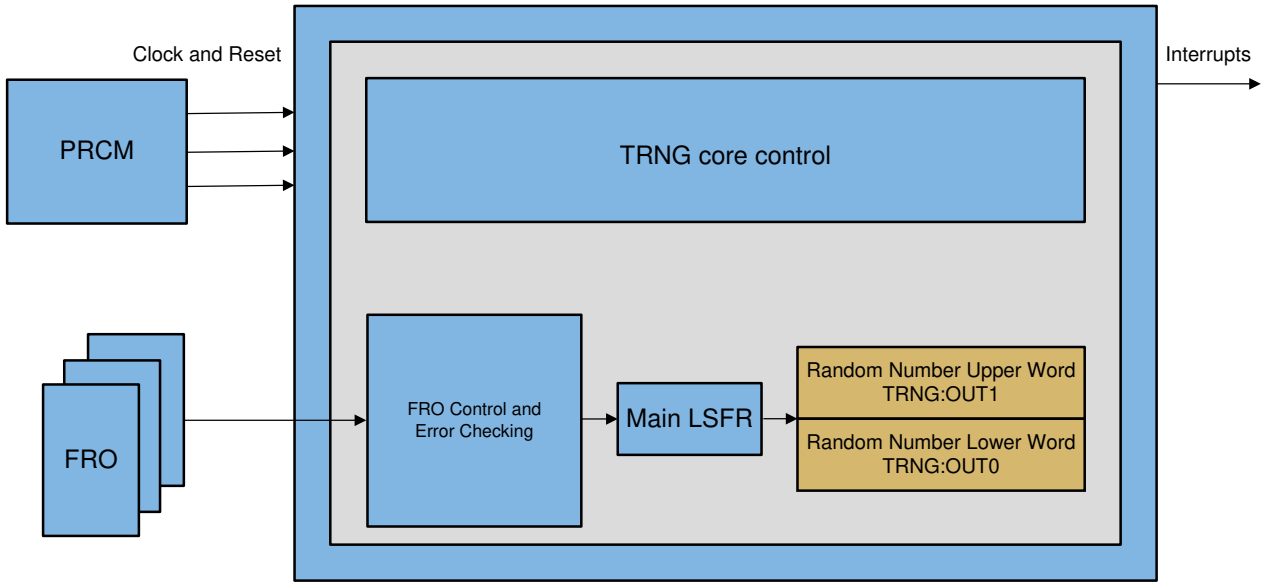


Figure 14-1. Random Number Generator Block Diagram

The TRNG core consists of two parts:

- The first part contains the Free-Running Oscillators (FROs), whose output signals are sampled at regular intervals. The FROs are asynchronous to one another and asynchronous to the sampling clock to make their behavior truly nondeterministic. Each FRO has an error detection circuit that checks for repeating patterns coming out of the FRO. If a repeating pattern is detected, the FRO is suspected of having locked onto the sampling clock, which drastically reduces the amount of entropy generated by that FRO (this is signaled as a FRO error event).
- The second part is the entropy accumulation circuit that uses an XOR tree to combine the sampled FRO clock outputs and an 81-bit LFSR to accumulate entropy (TRNG:LFSR0, TRNG:LFSR1, and TRNG:LFSR2 registers give the 81 bits main entropy accumulation LFSR).

The true entropy source is based upon a predetermined number of FROs. The accumulation of timing jitter, caused (for the largest part) by shot noise, creates uncertainty intervals for the output transitions of each FRO. Sampling within the uncertainly interval generates a small amount of entropy, which is accumulated in an LFSR. Entropy generation with multiple FROs in parallel allows the entropy accumulation to be done far more rapidly than is possible with one FRO.

14.3 TRNG Software Reset

A software reset of the module can be done by writing 1 to the TRNG:SWRESET.RESET register. When a software reset completes the TRNG:SWRESET.RESET register is automatically reset to 0. By polling the TRNG:SWRESET.RESET register for 0, the software can ensure that the reset is completed. The software reset must be completed before doing any TRNG operations.

14.4 Interrupt Requests

An interrupt request, TRNG_IRQ, is generated when data is ready for transmission (or an alarm was triggered). Table 14-1 lists the event flags, and their masks, that can cause module interrupts.

Table 14-1. Events

Event Flag	Event Mask	Description
TRNG:IRQSTAT.STAT	TRNG:IRQFLAGMASK.RDY and TRNG:IRQFLAGMASK. SHUTDOWN_OVF	Not used, but can be read for combined status of the two available interrupts
TRNG:IRQFLAGSTAT.RDY	TRNG:IRQFLAGMASK.RDY	When 1, data is available in the TRNG:OUT1 and the TRNG:OUT0 registers. Use TRNG:IRQFLAGCLR.RDY to clear it.
TRNG:IRQFLAGSTAT. SHUTDOWN_OVF	TRNG:IRQFLAGMASK. SHUTDOWN_OFV	When 1, the number of FROs shut down after a second error event (the number of 1 bits in the TRNG:ALARMSTOP register) has exceeded the threshold set by the TRNG:ALARMCNT.SHUTDOWN_THR register. Use the TRNG:IRQFLAGCLR.SHUTDOWN_OVF register to clear it.

The TRNG:ALARMCNT register, together with the TRNG:ALARMMASK and TRNG:ALARMSTOP registers, can be used by the host to determine if the FRO or sample cycle locking is a problem.

Lock detection in functional mode is performed using the sampled outputs of the individual FROs. A FRO alarm event is declared when a repeating pattern (of up to four samples length) is detected continuously for the number of samples defined by the TRNG:ALARMCNT:ALARM_THR register. The alarm event is logged by setting a bit to pinpoint the FRO in the TRNG:ALARMMASK register. If that bit in the TRNG:ALARMMASK register was already set, the corresponding bit in the TRNG:ALARMSTOP register is set and the FRO is switched off to prevent further alarm events from that FRO. If the TRNG:ALARMMASK register bit was not yet set, the FRO is restarted automatically in an attempt to break the locking. If a FRO is locked again after detune and re-enable, software must leave the FRO deactivated.

The TRNG:ALARMCNT.SHUTDOWN_CNT register bit field keeps track of the number of FROs switched off (actually, is a count of the number of 1 bits in the TRNG:ALARMSTOP register). The TRNG:ALARMCNT.SHUTDOWN_THR register bit field allows a configurable threshold to be set to generate the SHUTDOWN_OVF interrupt. When the TRNG:ALARMCNT.SHUTDOWN_CNT register exceeds the TRNG:ALARMCNT.SHUTDOWN_THR register, the TRNG:IRQFLAGSTAT.SHUTDOWN_OVF register bit is set to 1, which can be used to generate an interrupt.

14.5 TRNG Operation Description

Before the first random number generation, the TRNG:CTL and the TRNG:CFG0 registers must be written to start accumulating entropy. The entropy is a measure of the uncertainty associated with a random value. The random numbers are accessible to the application in a 64-bit read-only register TRNG:OUT0 and TRNG:OUT1. When the register is read, the TRNG generates a new value, which is available after the TRNG:CFG0.MIN_REFILL_CYCLES register system clock cycles and is then shifted into the output register. Software can use two strategies for operating the TRNG:

- **Monitored mode**: Software checks the TRNG:ALARMMASK register at regular intervals (in the order of seconds). If a bit is set there, the TRNG:ALARMSTOP register must also be checked to see if a FRO was shut down due to multiple alarm events—if none were shut down, the TRNG:ALARMMASK register can be cleared to get rid of the spurious alarm events. If one or more FROs were shut down, software can modify the delay selection of those FROs in the TRNG:FRODETUNE register in an attempt to prevent further locking. For this type of operation, the TRNG:ALARMCNT.SHUTDOWN_THR register would normally be set to a low value (2 for instance) and the SHUTDOWN_OVF interrupt can then be used to signal abnormal operation conditions and/or breakdowns of FROs.
- **Unmonitored mode**: Software sets the TRNG:ALARMCNT.SHUTDOWN_THR register to the number of FROs that are allowed to be shut down before corrective actions must be taken, and then uses the SHUTDOWN_OVF interrupt to initiate those corrective actions (clearing the TRNG:ALARMMASK and the TRNG:ALARMSTOP registers, toggling bits in the TRNG:FRODETUNE register). Software must keep track of the time interval between these interrupts. If they happen too often, this indicates abnormal operating conditions and/or breakdown of FROs.

14.5.1 TRNG Shutdown

The TRNG can be shutdown in many ways, but not all of them result in storing of entropy. The different modes are discussed here.

The best way is to not read the last generated random number. After the MAX_REFILL time (maximum of 2^{24} cycles) defined in the TRNG:CFG0 register, the TRNG enters idle mode where all FROs are turned off. When the generated value is read, the TRNG starts up again and generates a new random seed, which is ready after the time TRNG:CFG0.MIN_REFILL_CYCLES register. When the TRNG is in idle mode, the module clock can be turned off. Entropy is kept in between random number creations, so no reset (SW) of module is needed.

Another approach to shut down the TRNG is to just stop the module clock. By shutting down the TRNG by stopping the module clock, the entropy is also kept (that is, does not affect randomness), but the FROs might still be running. The clock can be enabled at any time, and the generation of a random seed is continued. There is no need for a soft reset of the module.

If the clock is stopped, the TRNG cannot be accessed and a bus fault is generated (within the Interconnect).

If an application that no longer needs the TRNG must go into deep sleep mode without waiting, the application can write 0 in the TRNG:CTL.TRNG_EN register bit, and the input system clock can be switched off. After such a shutdown, a soft reset of the TRNG module (see the register description in Section 14.7) should be performed before generating a new random number because randomness cannot be ensured. The penalty of this shutdown method is that entropy accumulation time is required before the next random number is ready.

14.5.2 TRNG Alarms

TRNG alarms happen and are most likely caused by FRO clock to sampling clock frequency locking. The sampling clock is the same as the system clock for the TRNG, and when the FRO oscillating frequency gets too close to a multiple of this clock, frequency lock might result in sampling the FRO clock at the same phase too

many times so a repeated pattern is detected. When such a repeated pattern is detected it is counted, and when this count exceeds the limit set by the TRNG:ALARMCNT.ALARM_THR register, the alarm event is triggered. Keeping this value high limits the number of alarms, and default is 255 alarm indications before an alarm event is enabled.

When an alarm event is triggered, the associated FRO is automatically shut down and not allowed to contribute to entropy accumulation. The user must then decide what to do with this event. Two options exist:

- Change the FRO oscillating frequency
- Leave the FRO off

For the first option, a bit in the TRNG:FRODETUNE.FRO_MASK register set to 1 allows the associated FRO run approximately 5 percent faster. The value of one of these bits may only be changed while the corresponding FRO is turned off (by temporary writing 0 in the corresponding bits of the TRNG:FROEN register—in case of an alarm this bit is already set to 0). When the value is updated, the corresponding FRO must be enabled again.

For the second option, the detune probably had no effect, or the FRO is not oscillating. This state must be stored so the corresponding bit in the TRNG:FROEN register is kept in off state to eliminate new alarm triggers caused by the particular FRO.

14.5.3 TRNG Entropy

Entropy is defined as a result of:

- How many FROs are enabled—with more, entropy is achieved faster
- How many samples are accumulated—longer running times yield higher entropy

The more FROs are enabled and the longer they run (that is, how many samples have been stored in the LSFR), the higher the entropy becomes.

The TRNG module must be running at maximum frequency when creating random values.

Creation time for a random value is defined by the values set in the TRNG:CTL.STARTUP_CYCLES register, the TRNG:CFG0.MIN_REFILL_CYCLES or the TRNG:CFG0.MAX_REFILL_CYCLES register and the TRNG:CFG0.SMPL_DIV register; modifications of all these registers can only be done when the TRNG:CTL.TRNG_EN register is 0.

The TRNG:CFG0.SMPL_DIV register defines how often a sample is collected from the FRO, default value 0 indicates that samples are taken every clock cycle, maximum value 0xF takes one sample every 16 clock cycles. All values of SAMPLE_DIV can be used on this device and it must be set as small as possible.

To have the same amount of entropy in each created seed, the startup and minimum refill times must be identical. By using minimum startup and minimum refill time, the entropy per bit is very low. When all FROs are enabled, a start-up time of 5 ms generates a word with 64-bit entropy.

Low values in the TRNG:CTL.STARTUP_CYCLES register and the TRNG:CFG0.MIN_REFILL_CYCLES or TRNG:CFG0.MAX_REFILL_CYCLES registers must only be used to generate random values for Non-secure use like synchronization words, CRC initialization, and so forth. For more secure usages the minimum of 64-bit entropy and beyond must be defined.

14.6 TRNG Low-Level Programming Guide

This section covers the low-level hardware programming sequences for configuration and usage of the module.

14.6.1 Initialization

14.6.1.1 Interfacing Modules

This section identifies the requirements of initializing the interfacing modules when the TRNG is to be used for the first time after a device reset. Table 14-2 lists the Initialization of interfacing modules.

Table 14-2. Initialization of Surrounding Modules

Interfacing Module	Comment
PRCM	TRNG module interface clock must be enabled. See PRCM:PDCTL0.PERIPH_ON and PRCM:SECDMACLKGR.TRNG_CLK_EN in Section 7.8.1
Arm® Cortex®-M33	NVIC configuration must be done to enable the interrupt from the TRNG. Only needed for interrupt based communication.
Interconnect	Interconnect must be enabled for communication with TRNG, which is handled in the PRCM as a consequence of many settings like CPU in run, sleep, or deep sleep mode, usage of DMA, I^2S, RFCORE, and Crypto engine.

14.6.1.2 TRNG Main Sequence

This procedure initializes the TRNG after a power-on reset (POR). Table 14-3 lists the TRNG main initialization sequence.

Table 14-3. TRNG Initialization Sequence

Step	Register or Bit Field
Execute a SW reset	TRNG:SWRESET.RESET
Wait for SW completion by polling	TRNG:SWRESET.RESET
Select the number of FRO clock input cycles between two samples	TRNG:CFG0.SMPL_DIV
Select the number of samples taken to gather enough entropy in the FROs of the module and to generate the first random value	TRNG:CTL.STARTUP_CYCLES
Select the minimum number of samples taken regenerate entropy in the FROs of the module and to generate subsequent random values	TRNG:CFG0.MIN_REFILL_CYCLES
Select the maximum number of samples taken regenerate entropy in the FROs of the module and to generate subsequent random values Also defines timeout period for shutting down the FROs after inactivity	TRNG:CFG0.MAX_REFILL_CYCLES
Configure the desired FROs to run 5% faster	TRNG:FRODETUNE.FRO_MASK
Enable all FROs	TRNG:FROEN.FRO_MASK
Select the maximum number of samples after which a detected repeated pattern an alarm event is generated	TRNG:ALARMCNT.ALARM_THR
Set the shutdown threshold to the number of FROs allowed to be shut down(1)	TRNG:ALARMCNT.SHUTDOWN_THR
Enable and start	TRNG:CTL.TRNG_EN

(1) This step is only required if unmonitored mode is used.

14.6.1.3 TRNG Operating Modes

This section presents the flow for different operating modes of the TRNG module.

14.6.1.3.1 Polling Mode

In polling mode, both monitored and unmonitored modes are covered. Figure 14-2 shows the TRNG polling mode.

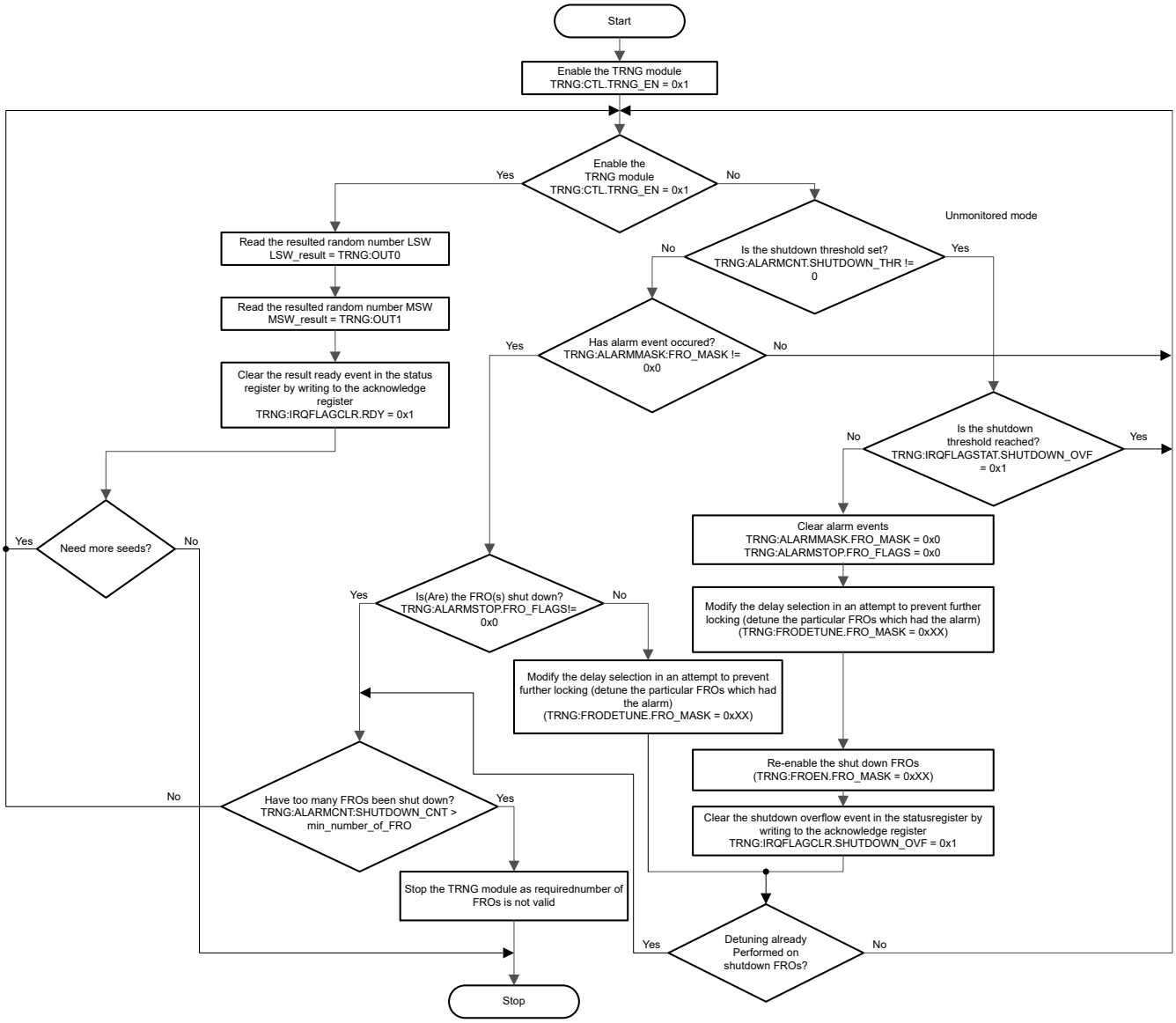


Figure 14-2. TRNG Polling Mode

14.6.1.3.2 Interrupt Mode

This section describes the event servicing of the module. Only the unmonitored mode is covered. Table 14-4 lists the TRNG interrupt mode steps, while Figure 14-3 shows the interrupt service routine flow.

Table 14-4. TRNG Interrupt Mode

Step	Register or Bit Field	Value
Enable interrupt generation when data is ready (available) in the output registers.	TRNG:IRQFLAGMASK.RDY	0x1
Enable the shutdown overflow interrupt generation when the maximum possible FRO shutdowns reach the selected shutdown threshold	TRNG:IRQFLAGMASK.SHUTDOWN_OVF	0x1
Enable the TRNG	TRNG:CTL.TRNG_EN	0x1

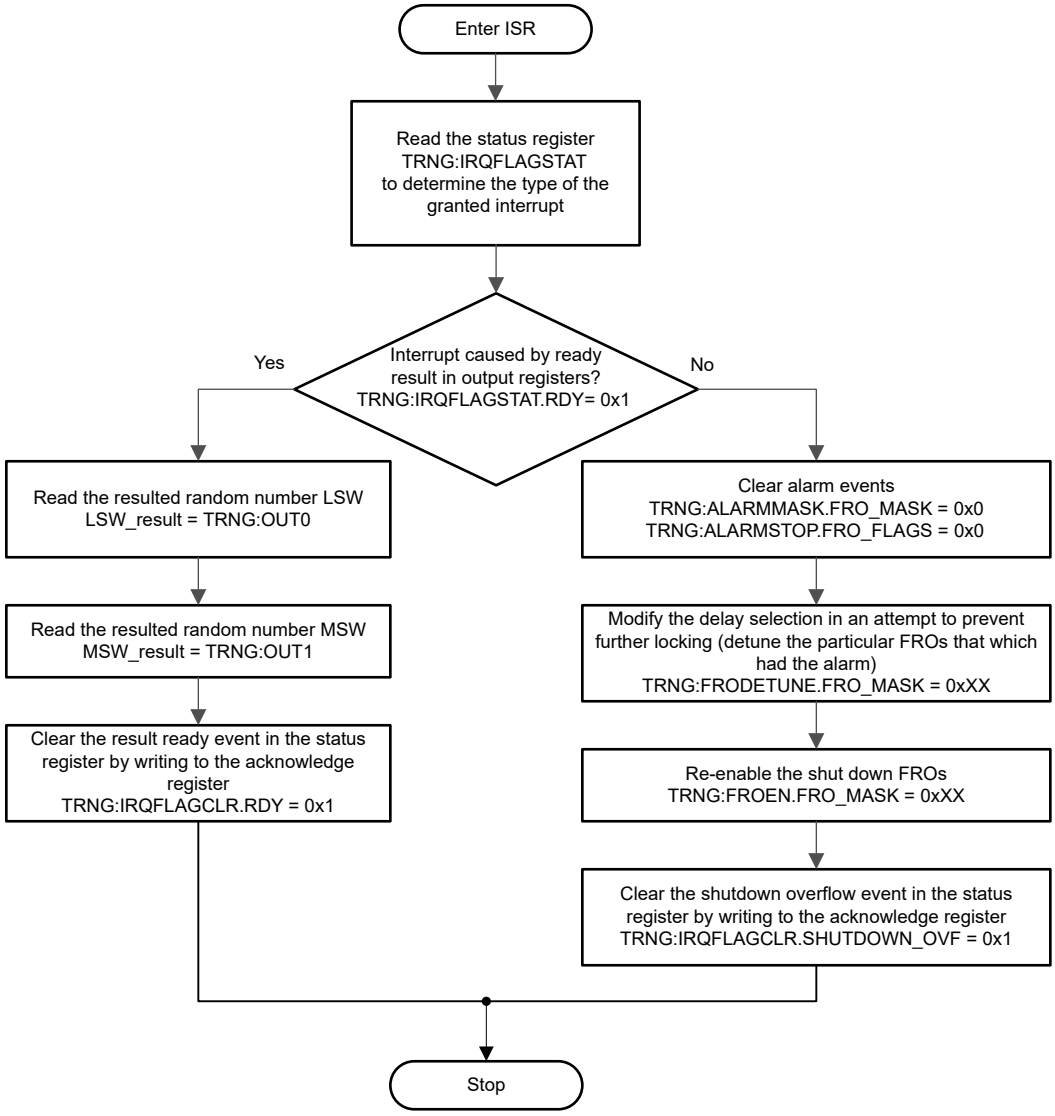


Figure 14-3. Interrupt Service Routine

14.7 TRNG Registers

Table 14-5 lists the memory-mapped registers for the TRNG registers. All register offset addresses not listed in Table 14-5 should be considered as reserved locations and the register contents should not be modified.

Table 14-5. TRNG Registers

Offset	Acronym	Register Name	Section
0h	OUT0	Random Number Lower Word Readout Value	Section 14.7.1
4h	OUT1	Random Number Upper Word Readout Value	Section 14.7.2
8h	IRQFLAGSTAT	Interrupt Status	Section 14.7.3
Ch	IRQFLAGMASK	Interrupt Mask	Section 14.7.4
10h	IRQFLAGCLR	Interrupt Flag Clear	Section 14.7.5
14h	CTL	Control	Section 14.7.6
18h	CFG0	Configuration 0	Section 14.7.7
1Ch	ALARMCNT	Alarm Control	Section 14.7.8
20h	FROEN	FRO Enable	Section 14.7.9
24h	FRODETUNE	FRO De-tune Bit	Section 14.7.10
28h	ALARMMASK	Alarm Event	Section 14.7.11
2Ch	ALARMSTOP	Alarm Shutdown	Section 14.7.12
30h	LFSR0	LFSR Readout Value	Section 14.7.13
34h	LFSR1	LFSR Readout Value	Section 14.7.14
38h	LFSR2	LFSR Readout Value	Section 14.7.15
78h	HWOPT	TRNG Engine Options Information	Section 14.7.16
7Ch	HWVER0	HW Version 0	Section 14.7.17
1FD8h	IRQSTATMASK	Interrupt Status After Masking	Section 14.7.18
1FE0h	HWVER1	HW Version 1	Section 14.7.19
1FECh	IRQSET	Interrupt Set	Section 14.7.20
1FF0h	SWRESET	SW Reset Control	Section 14.7.21
1FF8h	IRQSTAT	Interrupt Status	Section 14.7.22

Complex bit access types are encoded to fit into small table cells. Table 14-6 shows the codes that are used for access types in this section.

Table 14-6. TRNG Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

14.7.1 OUT0 Register (Offset = 0h) [Reset = 00000000h]

OUT0 is shown in Table 14-7.

Return to the Summary Table.

Random Number Lower Word Readout Value

Table 14-7. OUT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE_31_0	R	0h	LSW of 64- bit random value. New value ready when IRQFLAGSTAT.RDY = 1.

14.7.2 OUT1 Register (Offset = 4h) [Reset = 00000000h]

OUT1 is shown in Table 14-8.

Return to the Summary Table.

Random Number Upper Word Readout Value

Table 14-8. OUT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE_63_32	R	0h	MSW of 64-bit random value. New value ready when IRQFLAGSTAT.RDY = 1.

14.7.3 IRQFLAGSTAT Register (Offset = 8h) [Reset = 00000000h]

IRQFLAGSTAT is shown in Table 14-9.

Return to the Summary Table.

Interrupt Status

Table 14-9. IRQFLAGSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NEED_CLOCK	R	0h	1: Indicates that the TRNG is busy generating entropy or is in one of its test modes - clocks may not be turned off and the power supply voltage must be kept stable. 0: TRNG is idle and can be shut down
30-2	RESERVED	R	0h	Reserved
1	SHUTDOWN_OVF	R	0h	1: The number of FROs shut down (i.e. the number of '1' bits in the ALARMSTOP register) has exceeded the threshold set by ALARMCNT.SHUTDOWN_THR Writing '1' to IRQFLAGCLR.SHUTDOWN_OVF clears this bit to '0' again.
0	RDY	R	0h	1: Data are available in OUT0 and OUT1. Acknowledging this state by writing '1' to IRQFLAGCLR.RDY clears this bit to '0'. If a new number is already available in the internal register of the TRNG, the number is directly clocked into the result register. In this case the status bit is asserted again, after one clock cycle.

14.7.4 IRQFLAGMASK Register (Offset = Ch) [Reset = 00000000h]

IRQFLAGMASK is shown in Table 14-10.

Return to the Summary Table.

Interrupt Mask

Table 14-10. IRQFLAGMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	SHUTDOWN_OVF	R/W	0h	1: Allow IRQFLAGSTAT.SHUTDOWN_OVF to activate the interrupt from this module.
0	RDY	R/W	0h	1: Allow IRQFLAGSTAT.RDY to activate the interrupt from this module.

14.7.5 IRQFLAGCLR Register (Offset = 10h) [Reset = 00000000h]

IRQFLAGCLR is shown in Table 14-11.

Return to the Summary Table.

Interrupt Flag Clear

Table 14-11. IRQFLAGCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	SHUTDOWN_OVF	W	0h	1: Clear IRQFLAGSTAT.SHUTDOWN_OVF.
0	RDY	W	0h	1: Clear IRQFLAGSTAT.RDY.

14.7.6 CTL Register (Offset = 14h) [Reset = 00000000h]

CTL is shown in Table 14-12.

Return to the Summary Table.

Control

Table 14-12. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	STARTUP_CYCLES	R/W	0h	This field determines the number of samples (between 2^8 and 2^{24}) taken to gather entropy from the FROs during startup. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8. 0x0000: 2^{24} samples 0x0001: $1*2^8$ samples 0x0002: $2*2^8$ samples 0x0003: $3*2^8$ samples ... 0x8000: $32768*2^8$ samples 0xC000: $49152*2^8$ samples ... 0xFFFF: $65535*2^8$ samples This field can only be modified while TRNG_EN is 0. If 1 an update will be ignored.
15-11	RESERVED	R	0h	Reserved
10	TRNG_EN	R/W	0h	0: Forces all TRNG logic back into the idle state immediately. 1: Starts TRNG, gathering entropy from the FROs for the number of samples determined by STARTUP_CYCLES.
9-3	RESERVED	R	0h	Reserved
2	NO_LFSR_FB	R/W	0h	1: Remove XNOR feedback from the main LFSR, converting it into a normal shift register for the XOR-ed outputs of the FROs (shifting data in on the LSB side). A '1' also forces the LFSR to sample continuously. This bit can only be set to '1' when TEST_MODE is also set to '1' and should not be used for other than test purposes
1	TEST_MODE	R/W	0h	1: Enables access to the TESTCNT and LFSR0/LFSR1/LFSR2 registers (the latter are automatically cleared before enabling access) and keeps IRQFLAGSTAT.NEED_CLOCK at '1'. This bit shall not be used unless you need to change the LFSR seed prior to creating a new random value. All other testing is done external to register control.
0	RESERVED	R	0h	Reserved

14.7.7 CFG0 Register (Offset = 18h) [Reset = 00000000h]

CFG0 is shown in Table 14-13.

Return to the Summary Table.

Configuration 0

Table 14-13. CFG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MAX_REFILL_CYCLES	R/W	0h	This field determines the maximum number of samples (between 2^8 and 2^{24}) taken to re-generate entropy from the FROs after reading out a 64 bits random number. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8. 0x0000: 2^{24} samples 0x0001: $1*2^8$ samples 0x0002: $2*2^8$ samples 0x0003: $3*2^8$ samples ... 0x8000: $32768*2^8$ samples 0xC000: $49152*2^8$ samples ... 0xFFFF: $65535*2^8$ samples This field can only be modified while CTL.TRNG_EN is 0.
15-12	RESERVED	R	0h	Reserved
11-8	SMPL_DIV	R/W	0h	This field directly controls the number of clock cycles between samples taken from the FROs. Default value 0 indicates that samples are taken every clock cycle, maximum value 0xF takes one sample every 16 clock cycles. This field must be set to a value such that the slowest FRO (even under worst-case conditions) has a cycle time less than twice the sample period. This field can only be modified while CTL.TRNG_EN is '0'.
7-0	MIN_REFILL_CYCLES	R/W	0h	This field determines the minimum number of samples (between 2^6 and 2^{14}) taken to re-generate entropy from the FROs after reading out a 64 bits random number. If the value of this field is zero, the number of samples is fixed to the value determined by the MAX_REFILL_CYCLES field, otherwise the minimum number of samples equals the written value times 64 (which can be up to 2^{14}). To ensure same entropy in all generated random numbers the value 0 should be used. Then MAX_REFILL_CYCLES controls the minimum refill interval. The number of samples defined here cannot be higher than the number defined by the 'max_refill_cycles' field (i.e. that field takes precedence). No random value will be created if min refill > max refill. This field can only be modified while CTL.TRNG_EN = 0. 0x00: Minimum samples = MAX_REFILL_CYCLES (all numbers have same entropy) 0x01: $1*2^6$ samples 0x02: $2*2^6$ samples ... 0xFF: $255*2^6$ samples

14.7.8 ALARMCNT Register (Offset = 1Ch) [Reset = 000000FFh]

ALARMCNT is shown in Table 14-14.

Return to the Summary Table.

Alarm Control

Table 14-14. ALARMCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-24	SHUTDOWN_CNT	R/W	0h	Read-only, indicates the number of '1' bits in ALARMSTOP register. The maximum value equals the number of FROs.
23-21	RESERVED	R	0h	Reserved
20-16	SHUTDOWN_THR	R/W	0h	Threshold setting for generating IRQFLAGSTAT.SHUTDOWN_OVF interrupt. The interrupt is triggered when SHUTDOWN_CNT value exceeds this bit field.
15-8	RESERVED	R	0h	Reserved
7-0	ALARM_THR	R/W	FFh	Alarm detection threshold for the repeating pattern detectors on each FRO. An FRO 'alarm event' is declared when a repeating pattern (of up to four samples length) is detected continuously for the number of samples defined by this field's value. Reset value 0xFF should keep the number of 'alarm events' to a manageable level.

14.7.9 FROEN Register (Offset = 20h) [Reset = 00FFFFFFh]

FROEN is shown in Table 14-15.

Return to the Summary Table.

FRO Enable

Table 14-15. FROEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FRO_MASK	R/W	00FFFFFFh	Enable bits for the individual FROs. A '1' in bit [n] enables FRO 'n'. Default state is all '1's to enable all FROs after power-up. Note that they are not actually started up before the CTL.TRNG_EN bit is set to '1'. Bits are automatically forced to '0' here (and cannot be written to '1') while the corresponding bit in ALARMSTOP.FRO_FLAGS has value '1'.

14.7.10 FRODETUNE Register (Offset = 24h) [Reset = 00000000h]

FRODETUNE is shown in Table 14-16.

Return to the Summary Table.

FRO De-tune Bit

Table 14-16. FRODETUNE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FRO_MASK	R/W	0h	De-tune bits for the individual FROs. A '1' in bit [n] lets FRO 'n' run approximately 5% faster. The value of one of these bits may only be changed while the corresponding FRO is turned off (by temporarily writing a '0' in the corresponding bit of the FROEN.FRO_MASK register).

14.7.11 ALARMMASK Register (Offset = 28h) [Reset = 00000000h]

ALARMMASK is shown in Table 14-17.

Return to the Summary Table.

Alarm Event

Table 14-17. ALARMMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FRO_MASK	R/W	0h	Logging bits for the 'alarm events' of individual FROs. A '1' in bit [n] indicates FRO 'n' experienced an 'alarm event'.

14.7.12 ALARMSTOP Register (Offset = 2Ch) [Reset = 00000000h]

ALARMSTOP is shown in Table 14-18.

Return to the Summary Table.

Alarm Shutdown

Table 14-18. ALARMSTOP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FRO_FLAGS	R/W	0h	Logging bits for the 'alarm events' of individual FROs. A '1' in bit [n] indicates FRO 'n' experienced more than one 'alarm event' in quick succession and has been turned off. A '1' in this field forces the corresponding bit in FROEN.FRO_MASK to '0'.

14.7.13 LFSR0 Register (Offset = 30h) [Reset = 00000000h]

LFSR0 is shown in Table 14-19.

Return to the Summary Table.

LFSR Readout Value

Table 14-19. LFSR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LFSR_31_0	R/W	0h	Bits [31:0] of the main entropy accumulation LFSR. Register can only be accessed when CTL.TEST_MODE = 1. Register contents will be cleared to zero before access is enabled.

14.7.14 LFSR1 Register (Offset = 34h) [Reset = 00000000h]

LFSR1 is shown in Table 14-20.

Return to the Summary Table.

LFSR Readout Value

Table 14-20. LFSR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LFSR_63_32	R/W	0h	Bits [63:32] of the main entropy accumulation LFSR. Register can only be accessed when CTL.TEST_MODE = 1. Register contents will be cleared to zero before access is enabled.

14.7.15 LFSR2 Register (Offset = 38h) [Reset = 00000000h]

LFSR2 is shown in Table 14-21.

Return to the Summary Table.

LFSR Readout Value

Table 14-21. LFSR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-0	LFSR_80_64	R/W	0h	Bits [80:64] of the main entropy accumulation LFSR. Register can only be accessed when CTL.TEST_MODE = 1. Register contents will be cleared to zero before access is enabled.

14.7.16 HWOPT Register (Offset = 78h) [Reset = 00000600h]

HWOPT is shown in Table 14-22.

Return to the Summary Table.

TRNG Engine Options Information

Table 14-22. HWOPT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11-6	NR_OF_FROS	R	18h	Number of FROs implemented in this TRNG, value 24 (decimal).
5-0	RESERVED	R	0h	Reserved

14.7.17 HWVER0 Register (Offset = 7Ch) [Reset = 0200B44Bh]

HWVER0 is shown in Table 14-23.

Return to the Summary Table.

HW Version 0
EIP Number And Core Revision

Table 14-23. HWVER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	HW_MAJOR_VER	R	2h	4 bits binary encoding of the major hardware revision number.
23-20	HW_MINOR_VER	R	0h	4 bits binary encoding of the minor hardware revision number.
19-16	HW_PATCH_LVL	R	0h	4 bits binary encoding of the hardware patch level, initial release will carry value zero.
15-8	EIP_NUM_COMPL	R	B4h	Bit-by-bit logic complement of bits [7:0]. This TRNG gives 0xB4.
7-0	EIP_NUM	R	4Bh	8 bits binary encoding of the module number. This TRNG gives 0x4B.

14.7.18 IRQSTATMASK Register (Offset = 1FD8h) [Reset = 00000000h]

IRQSTATMASK is shown in Table 14-24.

Return to the Summary Table.

Interrupt Status After Masking

Table 14-24. IRQSTATMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	SHUTDOWN_OVF	R	0h	Shutdown Overflow (result of IRQFLAGSTAT.SHUTDOWN_OVF AND'ed with IRQFLAGMASK.SHUTDOWN_OVF)
0	RDY	R	0h	New random value available (result of IRQFLAGSTAT.RDY AND'ed with IRQFLAGMASK.RDY)

14.7.19 HWVER1 Register (Offset = 1FE0h) [Reset = 00000020h]

HWVER1 is shown in Table 14-25.

Return to the Summary Table.

HW Version 1
TRNG Revision Number

Table 14-25. HWVER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	REV	R	20h	The revision number of this module is Rev 2.0.

14.7.20 IRQSET Register (Offset = 1FECh) [Reset = 00000000h]

IRQSET is shown in Table 14-26.

Return to the Summary Table.

Interrupt Set

Table 14-26. IRQSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

14.7.21 SWRESET Register (Offset = 1FF0h) [Reset = 00000000h]

SWRESET is shown in Table 14-27.

Return to the Summary Table.

SW Reset Control

Table 14-27. SWRESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RESET	R/W	0h	Write '1' to soft reset , reset will be low for 4-5 clock cycles. Poll to 0 for reset to be completed.

14.7.22 IRQSTAT Register (Offset = 1FF8h) [Reset = 00000000h]

IRQSTAT is shown in Table 14-28.

Return to the Summary Table.

Interrupt Status

Table 14-28. IRQSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R	0h	TRNG Interrupt status. OR'ed version of IRQFLAGSTAT.SHUTDOWN_OVF and IRQFLAGSTAT.RDY

This page intentionally left blank.

Chapter 15

I/O Controller (IOC)

This chapter describes the input/output controller (IOC) and the general-purpose input/output (GPIO) module. The IOC provides a flexible configuration, as most of the peripheral ports can be mapped to any of the physical I/O pins. The CC13x4x10 and CC26x4x10 device platform has up to 46 I/O pins that are configurable as GPIO or peripheral I/O function.

15.1 Introduction 1080
15.2 IOC Overview 1080
15.3 I/O Mapping and Configuration 1081
15.4 Edge Detection on DIO Pins 1082
15.5 Unused I/O Pins 1083
15.6 GPIO 1083
15.7 I/O Pin Capability 1084
15.8 Peripheral PORT_IDs 1086
15.9 I/O Pins 1088
15.10 IOC Registers 1090

15.1 Introduction

The I/O controller configures I/O pins and maps peripheral signals to physical pins (DIOx) on the CC13x4x10 and CC26x4x10 device packages. This chapter explains the I/O controller function and gives a few examples on how to map peripheral functions to the pins chosen by the user.

Several similar terms are defined as follows:

- PORT_ID is the number for a peripheral function.
- GPIO is a peripheral function with a PORT_ID of 0x0.
- DIOn (DIO0 to DIO47) are the logical names of the different I/O pins on the specific package.
- Eight of these DIOs also have analog capabilities. The device-specific data sheet provides the mapping between DIO and pins for the different packages.

15.2 IOC Overview

Figure 15-1 shows a general overview of the IOC.

The IOC module consists of two main submodules:

- Microcontroller unit IOC (MCU IOC) configures the peripheral ports to the user-defined pins.
- Always-on IOC (AON IOC) module handles JTAG, 32 kHz clock, AON peripherals, and AUX Domain signals.

The always-on peripherals (RTC, Battery Monitor and internal temperature sensor) are clocked from the 32 kHz Low Frequency Clock, SCLK_LF (see Section 7.5 for more details). This allows the device to operate at very low power levels while still maintaining active operation of these peripheral functions. When configured correctly, the AON IOC ensures that output levels of all I/Os remain unchanged when the AUX power domain is powered down.

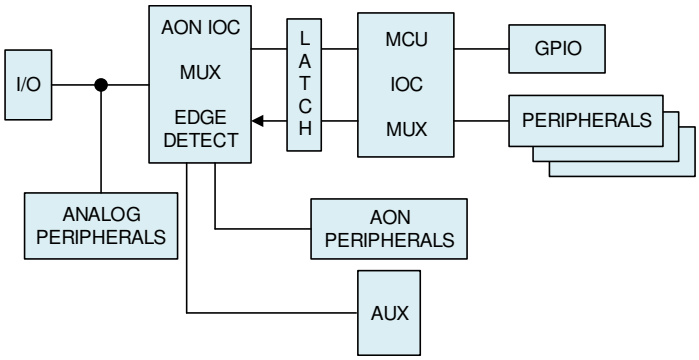


Figure 15-1. IOC Overview (Simplified)

15.3 I/O Mapping and Configuration

The MCU IOC can map a number of peripheral modules such as GPIO, SPI, UART, I^2C, and I^2S to any of the available I/Os. JTAG functions are limited to specific I/O pins. Each type of peripheral signal has a unique PORT_ID that can be assigned to selected I/O pins (referenced as DIOs). Table 15-3 lists the different PORT_ID signals.

15.3.1 Basic I/O Mapping

To map a peripheral function to DIOn, where n can range from 0 to a maximum of 47, the PORT_ID and pin configuration must be set in the corresponding IOC:IOCFGn register. To select what kind of function the pin must be routed, choose the PORT_ID number for the desired peripheral function and write the PORT_ID number to the IOC:IOCFGn.PORT_ID bit field.

15.3.2 Mapping AUXIOs to DIO Pins

There are up to 32 signals (AUXIO0 to AUXIO31) in the sensor controller domain (AUX Domain). These signals can be routed to specific DIO pins given in Table 15-2. The signals AUXIO19 to AUXIO26 have analog capability, but can also be used as digital I/Os. All the other AUXIOn signals are digital only. The signals routed from the AUX Domain are configured differently than GPIO and other peripheral functions. This section does not cover the use of all the capabilities of the sensor controller (for more details, see Chapter 20).

In this example, AUXIO1 is mapped to DIO17 and set up as a digital input.

1. Set the IOC:IOCFG17.PORT_ID bit field to 0x08 (AUX_I/O) to route AUXIO1 to DIO17.
2. The I/O signals in the AUX domain have their own open-source or open-drain configuration, which must be set in the AUX_AIODIO0:IOMODE register in the AUX domain. Set AUX_AIODIO0:IOMODE.IO1 to 0x01 to enable AUXIO1 as a digital input.
3. Enable the digital input buffer for AUXIO1 by setting the IO7_0 bit field to 0x02 in the AUX_AIODIO0:GPIODIE register.

15.3.3 Control External LNA/PA (Range Extender) with I/Os

There are four logic RF-Core internal output signals called RF Core Data Out n, where n goes from 0 to 3. These signals can be mapped to DIOs. By default, RF Core Data Out 0 is set to go high when the LNA must be enabled, and RF Core Data Out 1 is set high when the PA must be enabled. Table 15-1 describes the signals. The signals can be mapped to any DIO by setting the relevant PORT_ID in the designated IOCFGn register.

```
#include <ti/drivers/GPIO.h>
// Map RFC_GPOn to CONFIG_GPIO_x
GPIO_setConfig(CONFIG_GPIO_x, GPIO_CFG_OUT_STD | GPIO_CFG_OUT_LOW);
GPIO_setMux(CONFIG_GPIO_x, IOC_PORT_RFC_GPOn);
```

Table 15-1. RF Core Data Signals for PA and LNA

Port Name	PORT_ID	RF Core Signal	Description
RFC_GPO0	0x2F	RF Core Data Out 0	LNA enable
RFC_GPO1	0x30	RF Core Data Out 1	PA enable
RFC_GPO2	0x31	RF Core Data Out 2	Synthesizer calibration running
RFC_GPO3	0x32	RF Core Data Out 3	TX start

15.3.4 Map the 32 kHz System Clock (SCLK_LF Clock) to DIO

The AON IOC contains the output enable control for the 32 kHz LF system clock (SCLK_LF) output, and the clock signal has its own PORT_ID called AON_CLK32K (0x7). This makes it easy to output the clock signal to a pin. Map the clock to a chosen DIO, and enable the clock output by setting the AON_IOC:CLK32KCTL.OE_N register field to 0x0. The following code snippets show how this can be done:

```
#include <ti/devices/DeviceFamily.h>
#include DeviceFamily_constructPath(driverlib/aon_ioc.h)
GPIO_setMux(CONFIG_GPIO_x, IOC_PORT_AON_CLK32K); // CONFIG_GPIO_x is assigned
                                                 // DIO through SysConfig
AONIOC32kHzOutputEnable();
```

This outputs the LF system clock signal in all power modes except for Shutdown.

The AON_CLK32K PORT_ID value is also chosen when using a DIO as the input source for the 32 kHz LF system clock. For a description of how to use a DIO as the clock source, see Chapter 7.

Note

The AON_CLK32K PORT_ID must be used as only a single clock output or as only a clock input.

15.4 Edge Detection on DIO Pins

The AON IOC supports detection of positive and (or) negative edges on the digital I/Os and provides the resulting events to the AON event fabric as the following events:

- IOEV_AON_PROG0
- IOEV_AON_PROG1
- IOEV_AON_PROG2
- IOEV_RTC
- IOEV_MCU_WU

The edge-detect event can be cleared by both the MCU GPIO and the AUX Domain. The edge detect event can also be cleared from MCU IOC by doing a disable or enable cycle of the edge configuration. The MCU GPIO has a separate clear line to each edge detection cell, while the AUX Domain uses a single line to clear all events on pins connected to the AUX Domain. When clearing from AUX Domain, all events related to AUX Domain I/Os are cleared.

The edge detect block uses an edge-detect cell for each I/O. Each detection cell can flag edge-detected and trigger an interrupt signal. The interrupt signals from all cells are ORed together to form a single interrupt line toward the AON event fabric.

The AON IOC can also generate an interrupt event when any programmable subset of the input I/Os generates an event. The registers controlling the edge-detect circuit reside in the MCU IOC.

15.4.1 Configure DIO as GPIO Input to Generate Interrupt on Edge Detect

Interrupt and edge detect event generation from DIOs is configured through the IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET bit fields. The DIO must have input enable set in order to perform edge detection. A GPIO edge-detect event is sent to the CPU interrupt IRQ0 (vector number 16). This interrupt must be enabled to call the GPIO interrupt handler. In this interrupt handler, the event source must be cleared by clearing the relevant GPIO:EVFLAGS31 event register DIOn bit. Reading this register returns 1 for triggered events and 0 for non-triggered events. The event is cleared from the MCU IOC by toggling the enabled EDGE_DET configuration. The event is cleared when the active-edge configuration is disabled and IOC:IOCFGn.EDGE_DET set to 0.

15.5 Unused I/O Pins

By default, the I/O driver (output) and input buffer (input) are disabled (tri-state mode) at power on or reset, and thus the I/O pin can safely be left unconnected (floating).

If the I/O pin is in a tri-state condition and connected to a node with a different voltage potential, a small leakage current can go through the pin. The same applies to an I/O pin configured as input, where the pin is connected to a voltage source (for example VDD/2). The input is then an undefined value of either 0 or 1.

15.6 GPIO

The MCU GPIO is a general-purpose input/output module that allows software to write to and read from the DIOs. GPIO supports up to 46 programmable I/O pins. These pins are configured by the IOC module. To modify a single GPIO output value, use the GPIO:DOUTn registers (see Section 15.10.2). The following describes the necessary steps to set up DIO1 as a GPIO output and toggle the bit.

TI recommends using the GPIO driver in the SimpleLink™ CC13xx and CC26xx software development kit (SDK) when managing general purpose I/Os.

1. Map DIO1 as a GPIO output by setting the IOC:IOCFG1.PORT_ID register to 0 (GPIO PORT_ID)
2. Set DIO1 as output by clearing the IOC:IOCFG1.IE bit. More port configurations can also be set in the IOC:IOCFG1 register (for more details, see Section 15.9.1.2)
3. Set the data output enable bit for DIO1 in GPIO:DOE31_0.DIO1 by issuing a read-modify-write operation
4. Toggle the DIO1 output by issuing an XOR operation on the GPIO:DOUT3_0:DIO1 bit with 0x100

15.7 I/O Pin Capability

Table 15-2 shows the I/O capabilities for DIOs. Refer to the device-specific data sheet for DIO mapping to package pins.

Table 15-2. CC13x4x10 and CC26x4x10 Pin Mapping

Package Type		Sensor Controller		Drive Strength	JTAG
8 x 8 QFN (RSK)	7 × 7 QFN (RGZ)	Analog Capable	AUX Domain I/O		
DIO[1]	DIO[1]				
47			N/A	2 mA, 4 mA	
46			N/A	2 mA, 4 mA	
45			N/A	2 mA, 4 mA	
44			N/A	2 mA, 4 mA	
43			N/A	2 mA, 4 mA	
42			N/A	2 mA, 4 mA	
41			N/A	2 mA, 4 mA	
40			N/A	2 mA, 4 mA	
39			N/A	2 mA, 4 mA	
38			N/A	2 mA, 4 mA	
37			N/A	2 mA, 4 mA	
36			N/A	2 mA, 4 mA	
35			N/A	2 mA, 4 mA	
34			N/A	2 mA, 4 mA	
33			N/A	2 mA, 4 mA	
32			N/A	2 mA, 4 mA	
30	30	Yes	19	2 mA, 4 mA	
29	29	Yes	20	2 mA, 4 mA	
28	28	Yes	21	2 mA, 4 mA	
27	27	Yes	22	2 mA, 4 mA	
26	26	Yes	23	2 mA, 4 mA	
25	25	Yes	24	2 mA , 4 mA	
24	24	Yes	25	2 mA , 4 mA	
23	23	Yes	26	2 mA , 4 mA	
22	22		27	2 mA , 4 mA	
21	21		28	2 mA, 4 mA	
20	20		29	2 mA, 4 mA	
19	19		30	2 mA, 4 mA	
18	18		31	2 mA, 4 mA	
17	17		1	2 mA, 4 mA, 8 mA	TDI
16	16		2	2 mA, 4 mA, 8 mA	TDO
15	15		3	2 mA, 4 mA	
14	14		4	2 mA, 4 mA	

Table 15-2. CC13x4x10 and CC26x4x10 Pin Mapping (continued)

Package Type		Sensor Controller		Drive Strength	JTAG
8 x 8 QFN (RSK)	7 × 7 QFN (RGZ)	Analog Capable	AUX Domain I/O		
DIO[1]	DIO[1]				
13	13		5	2 mA, 4 mA	
12	12		6	2 mA, 4 mA	
11	11		7	2 mA, 4 mA	
10	10		8	2 mA, 4 mA	
9	9		9	2 mA, 4 mA	
8	8		10	2 mA, 4 mA	
7	7		11	2 mA, 4 mA, 8 mA	
6	6		12	2 mA, 4 mA, 8 mA	
5	5		13	2 mA, 4 mA, 8 mA	
4	4		14	2 mA, 4 mA	
3	3		15	2 mA, 4 mA	
2	2		16	2 mA, 4 mA	
1	1		17	2 mA, 4 mA	
0	0		18	2 mA, 4 mA	

(1) Please see the data sheet for detailed information regarding which DIOs are supported by a specific device

15.8 Peripheral PORT_IDs

Table 15-3 lists the different PORT_ID signals.

Table 15-3. CC13x4x10 and CC26x4x10 PORT_IDs

ID	Port Name	Port Description		ID	Port Name	Port Description
0	GPIO	Default GPIO usage		39	I2S_WCLK	I^2S WCLK pin
1–6		Reserved		40	I2S_BCLK	I^2S BCLK pin
7	AON_CLK32K	AON 32 kHz clock pin (SCLK_LF)		41	I2S_MCLK	I^2S MCLK pin
8	AUX_IO(1)	AUX Domain I/O pin		42–45		Reserved
9	SPI0_RX	SPI 0 RX pin		46	RFC_TRC	RF Core trace
10	SPI0_TX	SPI 0 TX pin		47	RFC_GPO0	RF Core general-purpose output 0
11	SPI0_CS	SPI 0 CS pin		48	RFC_GPO1	RF Core general-purpose output 1
12	SPI0_CLK	SPI 0 CLK pin		49	RFC_GPO2	RF Core general-purpose output 2
13	I2C0_MSSDA	I^2C 0 data		50	RFC_GPO3	RF Core general-purpose output 3
14	I2C0_MSSCL	I^2C 0 clock		51	RFC_GPI0	RF Core Data In 0
5	UART0_RX	UART 0 RX pin		52	RFC_GPI1	RF Core Data In 1
16	UART0_TX	UART 0 TX pin		53	RFC_SMI_DL_OUT	RF Core SMI Data Link Out
17	UART0_CTS	UART 0 CTS pin		54	RFC_SMI_DL_IN	RF Core SMI Data Link In
18	UART0_RTS	UART 0 RTS pin		55	RFC_SMI_CL_OUT	RF Core SMI Command Link Out
19	UART1_RX	UART 1 RX pin		56	RFC_SMI_CL_IN	RF Core SMI Command Link In
20	UART1_TX	UART 1 TX pin		57	SPI2_RX	SPI 2 RX pin
21	UART1_CTS	UART 1 CTS pin		58	SPI2_TX	SPI 2 TX pin
22	UART1_RTS	UART 1 RTS pin		59	SPI2_CS	SPI 2 CS pin
23	PORT_EVENT0	General-purpose I/O event 0		60	SPI2_CLK	SPI 2 CLK pin
24	PORT_EVENT1	General-purpose I/O event 1		61	SPI3_RX	SPI 3 RX pin
25	PORT_EVENT2	General-purpose I/O event 2		62	SPI3_TX	SPI 3 TX pin
26	PORT_EVENT3	General-purpose I/O event 3		63	SPI3_CS	SPI 3 CS pin
27	PORT_EVENT4	General-purpose I/O event 4		64	SPI3_CLK	SPI 3 CLK pin
28	PORT_EVENT5	General-purpose I/O event 5		65	UART2_RX	UART 2 RX pin
29	PORT_EVENT6	General-purpose I/O event 6		66	UART2_TX	UART 2 TX pin
30	PORT_EVENT7	General-purpose I/O event 7		67	UART2_CTS	UART 2 CTS pin
31		Reserved		68	UART2_RTS	UART 2 RTS pin
32	CPU_SWV	CPU SWV		69	UART3_RX	UART 3 RX pin
33	SPI1_RX	SPI 1 RX pin		70	UART3_TX	UART 3 TX pin
34	SPI1_TX	SPI 1 TX pin		71	UART3_CTS	UART 3 CTS pin
35	SPI1_CS	SPI 1 CS pin		72	UART3_RTS	UART 3 RTS pin
36	SPI1_CLK	SPI 1 CLK pin		73	I2C1_MSSDA	I^2C 1 data
37	I2S_AD0	I^2S data 0 pin		74	I2C1_MSSCL	I^2C 1 clock

Table 15-3. CC13x4x10 and CC26x4x10 PORT_IDs (continued)

ID	Port Name	Port Description		ID	Port Name	Port Description
38	I2S_AD1	I^2S data 1 pin				

(1) AUX_IO is only available for DIO0 - DIO31

15.9 I/O Pins

This section discusses specific physical details and configuration possibilities for the I/O pins on the CC13x4x10 and CC26x4x10 device platform.

15.9.1 Input/Output Modes

Each I/O pin has separate input and output buffers which can be configured independently. The main configurations for input and output are:

- Input mode (detached, hysteresis, pullup, pulldown)
- Output mode (tri-state condition, push-pull, open drain, open source)

Both the input and the output buffer can be enabled or disabled at the same time. By disabling the output buffer the corresponding I/O pin will be in the tri-state condition (high impedance). If nothing is driving the I/O to a valid logical level when the output buffer is disabled then disable the input buffer to avoid excessive current draw through the I/O input buffer. Section 15.9.1.2 describes the I/O pin configuration in more detail.

15.9.1.1 Physical Pin

The digital I/O driver and receiver is a wide-supply voltage range, bidirectional buffer combining an output buffer, an input buffer with optional hysteresis, and optional pullup and pulldown circuitry. The I/O has limited power-management features, including support for wakeup from sleep with core power gated. The sink and source capability of the pins are symmetrical, as shown in Figure 15-2, which gives a rough overview of the I/O pin.

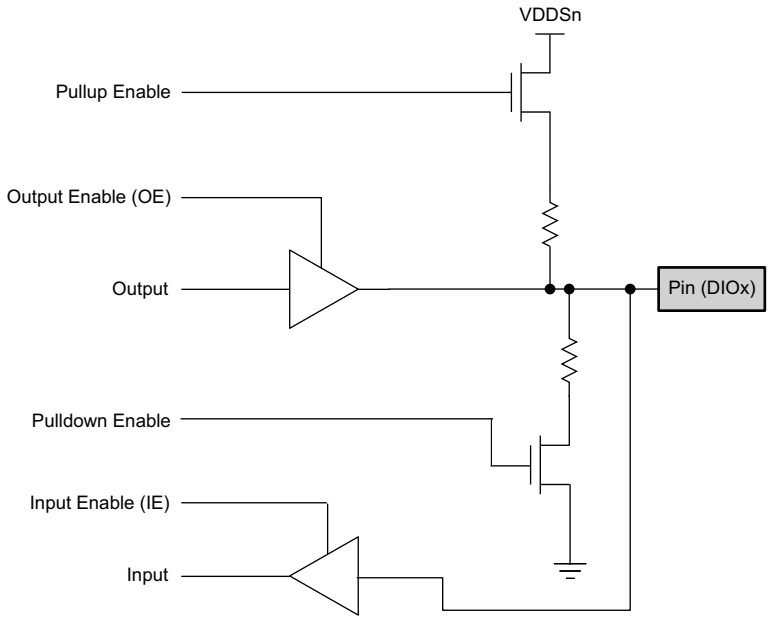


Figure 15-2. Generic I/O Pin (Simplified)

15.9.1.2 Pin Configuration

The IOC allows software to configure the pins based on the requirements of the application. The software can configure different characteristic settings for any or all of the I/O pins. All of the following features, except for output driver (output enable set in the GPIO:DOE31_0 or GPIO:DOE47_32 registers), are controlled in the IOC:IOCFGn registers:

- **Drive Strength** (IOC:IOCFGn.IOSTR)
  - Configures the drive strength and maximum current of an I/O pin. All I/O pins support 2 mA and 4 mA, while five pins support up to 8 mA. By setting IOC:IOCFGn.IOSTR to 0x0, the drive strength is automatically updated based upon inputs from the battery monitor, BATMON, to maintain the set drive strength level at different battery voltages.
- **Pull** (IOC:IOCFGn.PULL_CTL)
  - Configures a weak pull on an I/O pin. The following can be set: pullup, pulldown, or no pull. See the data sheet for specific pullup and pulldown resistance.
- **Slew Control** (IOC:IOCFGn.SLEW_RED)
  - Sets normal or reduced slew rate on an I/O pin.
- **Hysteresis** (IOC:IOCFGn.HYST_EN)
  - Enables or disables input hysteresis on an I/O pin.
- **Open-Source or Open-Drain Configuration** (IOC:IOCFGn.IOMODE)
  - Configures the pin as normal, open source, or open drain; all of these can be set to either inverted or normal (noninverted).
- **Interrupt and Edge Detection** (IOC:IOCFGn.EDGE_IRQ_EN and IOC:IOCFGn.EDGE_DET)
  - Enables interrupt triggered by edge detection on I/O pin. The following modes are supported:
    - Rising edge
    - Falling edge
    - Trigger on both rising and falling
    - No edge detection
- **Input Driver** (IOC:IOCFGn.IE)
  - Enables or disables the I/O input driver.
- **Output Driver** (Depends on specific peripheral mapped to pin)

15.10 IOC Registers

15.10.1 AON_IOC Registers

Table 15-4 lists the memory-mapped registers for the AON_IOC registers. All register offset addresses not listed in Table 15-4 should be considered as reserved locations and the register contents should not be modified.

Table 15-4. AON_IOC Registers

Offset	Acronym	Register Name	Section
0h	IOSTRMIN	Internal	Section 15.10.1.1
4h	IOSTRMED	Internal	Section 15.10.1.2
8h	IOSTRMAX	Internal	Section 15.10.1.3
Ch	IOCLATCH	IO Latch Control	Section 15.10.1.4
10h	CLK32KCTL	SCLK_LF External Output Control	Section 15.10.1.5
14h	TCKCTL	TCK IO Pin Control	Section 15.10.1.6

Complex bit access types are encoded to fit into small table cells. Table 15-5 shows the codes that are used for access types in this section.

Table 15-5. AON_IOC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

15.10.1.1 IOSTRMIN Register (Offset = 0h) [Reset = 00000003h]

IOSTRMIN is shown in Table 15-6.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 15-6. IOSTRMIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	GRAY_CODE	R/W	3h	Internal. Only to be used through TI provided API.

15.10.1.2 IOSTRMED Register (Offset = 4h) [Reset = 00000006h]

IOSTRMED is shown in Table 15-7.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 15-7. IOSTRMED Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	GRAY_CODE	R/W	6h	Internal. Only to be used through TI provided API.

15.10.1.3 IOSTRMAX Register (Offset = 8h) [Reset = 00000005h]

IOSTRMAX is shown in Table 15-8.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 15-8. IOSTRMAX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	GRAY_CODE	R/W	5h	Internal. Only to be used through TI provided API.

15.10.1.4 IOCLATCH Register (Offset = Ch) [Reset = 00000001h]

IOCLATCH is shown in Table 15-9.

Return to the Summary Table.

IO Latch Control
Controls transparency of all latches holding I/O or configuration state from the MCU IOC

Table 15-9. IOCLATCH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	1h	Controls latches between MCU IOC and AON_IOC. The latches are transparent by default. They must be closed prior to power off the domain(s) controlling the IOs in order to preserve IO values on external pins. 0h = Latches are static, meaning the current value on the IO pin is frozen by latches and kept even if GPIO module or a peripheral module is turned off 1h = Latches are transparent, meaning the value of the IO is directly controlled by the GPIO or peripheral value

15.10.1.5 CLK32KCTL Register (Offset = 10h) [Reset = 00000001h]

CLK32KCTL is shown in Table 15-10.

Return to the Summary Table.

SCLK_LF External Output Control

Table 15-10. CLK32KCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	OE_N	R/W	1h	0: Output enable active. SCLK_LF output on IO pin that has PORT_ID (for example IOC:IOCFG0.PORT_ID) set to AON_CLK32K. 1: Output enable not active

15.10.1.6 TCKCTL Register (Offset = 14h) [Reset = 00000001h]

TCKCTL is shown in Table 15-11.

Return to the Summary Table.

TCK IO Pin Control

Table 15-11. TCKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	1h	0: Input driver for TCK disabled. 1: Input driver for TCK enabled.

15.10.2 GPIO Registers

Table 15-12 lists the memory-mapped registers for the GPIO registers. All register offset addresses not listed in Table 15-12 should be considered as reserved locations and the register contents should not be modified.

Table 15-12. GPIO Registers

Offset	Acronym	Register Name	Section
0h	DOUT3_0	Data Out 0 to 3	Section 15.10.2.1
4h	DOUT7_4	Data Out 4 to 7	Section 15.10.2.2
8h	DOUT11_8	Data Out 8 to 11	Section 15.10.2.3
Ch	DOUT15_12	Data Out 12 to 15	Section 15.10.2.4
10h	DOUT19_16	Data Out 16 to 19	Section 15.10.2.5
14h	DOUT23_20	Data Out 20 to 23	Section 15.10.2.6
18h	DOUT27_24	Data Out 24 to 27	Section 15.10.2.7
1Ch	DOUT31_28	Data Out 28 to 31	Section 15.10.2.8
20h	DOUT35_32	Data Out 35 to 32	Section 15.10.2.9
24h	DOUT39_36	Data Out 39 to 36	Section 15.10.2.10
28h	DOUT43_40	Data Out 43 to 40	Section 15.10.2.11
2Ch	DOUT47_44	Data Out 47 to 44	Section 15.10.2.12
80h	DOUT31_0	Data Output for DIO 0 to 31	Section 15.10.2.13
84h	DOUT47_32	Data Output for DIO 0 to 31	Section 15.10.2.14
90h	DOUTSET31_0	Data Out Set	Section 15.10.2.15
94h	DOUTSET47_32	Data Out Set	Section 15.10.2.16
A0h	DOUTCLR31_0	Data Out Clear	Section 15.10.2.17
A4h	DOUTCLR47_32	Data Out Clear	Section 15.10.2.18
B0h	DOUTTGL31_0	Data Out Toggle	Section 15.10.2.19
B4h	DOUTTGL47_32	Data Out Toggle	Section 15.10.2.20
C0h	DIN31_0	Data Input from DIO 0 to 31	Section 15.10.2.21
C4h	DIN47_32	Data Input from DIO 32 to 47	Section 15.10.2.22
D0h	DOE31_0	Data Output Enable for DIO 0 to 31	Section 15.10.2.23
D4h	DOE47_32	Data Output Enable for DIO 32 to 47	Section 15.10.2.24
E0h	EVFLAGS31_0	Event Register for DIO 0 to 31	Section 15.10.2.25
E4h	EVFLAGS47_32	Event Register for DIO 32 to 47	Section 15.10.2.26

Complex bit access types are encoded to fit into small table cells. Table 15-13 shows the codes that are used for access types in this section.

Table 15-13. GPIO Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

15.10.2.1 DOUT3_0 Register (Offset = 0h) [Reset = 00000000h]

DOUT3_0 is shown in Table 15-14.

Return to the Summary Table.

Data Out 0 to 3
Alias register for byte access to each bit in DOUT47_0

Table 15-14. DOUT3_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO3	R/W	0h	Sets the state of the pin that is configured as DIO#3, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO2	R/W	0h	Sets the state of the pin that is configured as DIO#2, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO1	R/W	0h	Sets the state of the pin that is configured as DIO#1, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO0	R/W	0h	Sets the state of the pin that is configured as DIO#0, if the corresponding DOE47_0 bitfield is set.

15.10.2.2 DOUT7_4 Register (Offset = 4h) [Reset = 00000000h]

DOUT7_4 is shown in Table 15-15.

Return to the Summary Table.

Data Out 4 to 7
Alias register for byte access to each bit in DOUT47_0

Table 15-15. DOUT7_4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO7	R/W	0h	Sets the state of the pin that is configured as DIO#7, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO6	R/W	0h	Sets the state of the pin that is configured as DIO#6, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO5	R/W	0h	Sets the state of the pin that is configured as DIO#5, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO4	R/W	0h	Sets the state of the pin that is configured as DIO#4, if the corresponding DOE47_0 bitfield is set.

15.10.2.3 DOUT11_8 Register (Offset = 8h) [Reset = 00000000h]

DOUT11_8 is shown in Table 15-16.

Return to the Summary Table.

Data Out 8 to 11
Alias register for byte access to each bit in DOUT47_0

Table 15-16. DOUT11_8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO11	R/W	0h	Sets the state of the pin that is configured as DIO#11, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO10	R/W	0h	Sets the state of the pin that is configured as DIO#10, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO9	R/W	0h	Sets the state of the pin that is configured as DIO#9, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO8	R/W	0h	Sets the state of the pin that is configured as DIO#8, if the corresponding DOE47_0 bitfield is set.

15.10.2.4 DOUT15_12 Register (Offset = Ch) [Reset = 00000000h]

DOUT15_12 is shown in Table 15-17.

Return to the Summary Table.

Data Out 12 to 15
Alias register for byte access to each bit in DOUT47_0

Table 15-17. DOUT15_12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO15	R/W	0h	Sets the state of the pin that is configured as DIO#15, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO14	R/W	0h	Sets the state of the pin that is configured as DIO#14, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO13	R/W	0h	Sets the state of the pin that is configured as DIO#13, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO12	R/W	0h	Sets the state of the pin that is configured as DIO#12, if the corresponding DOE47_0 bitfield is set.

15.10.2.5 DOUT19_16 Register (Offset = 10h) [Reset = 00000000h]

DOUT19_16 is shown in Table 15-18.

Return to the Summary Table.

Data Out 16 to 19
Alias register for byte access to each bit in DOUT47_0

Table 15-18. DOUT19_16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO19	R/W	0h	Sets the state of the pin that is configured as DIO#19, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO18	R/W	0h	Sets the state of the pin that is configured as DIO#18, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO17	R/W	0h	Sets the state of the pin that is configured as DIO#17, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO16	R/W	0h	Sets the state of the pin that is configured as DIO#16, if the corresponding DOE47_0 bitfield is set.

15.10.2.6 DOUT23_20 Register (Offset = 14h) [Reset = 00000000h]

DOUT23_20 is shown in Table 15-19.

Return to the Summary Table.

Data Out 20 to 23
Alias register for byte access to each bit in DOUT47_0

Table 15-19. DOUT23_20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO23	R/W	0h	Sets the state of the pin that is configured as DIO#23, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO22	R/W	0h	Sets the state of the pin that is configured as DIO#22, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO21	R/W	0h	Sets the state of the pin that is configured as DIO#21, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO20	R/W	0h	Sets the state of the pin that is configured as DIO#20, if the corresponding DOE47_0 bitfield is set.

15.10.2.7 DOUT27_24 Register (Offset = 18h) [Reset = 00000000h]

DOUT27_24 is shown in Table 15-20.

Return to the Summary Table.

Data Out 24 to 27
Alias register for byte access to each bit in DOUT47_0

Table 15-20. DOUT27_24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO27	R/W	0h	Sets the state of the pin that is configured as DIO#27, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO26	R/W	0h	Sets the state of the pin that is configured as DIO#26, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO25	R/W	0h	Sets the state of the pin that is configured as DIO#25, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO24	R/W	0h	Sets the state of the pin that is configured as DIO#24, if the corresponding DOE47_0 bitfield is set.

15.10.2.8 DOUT31_28 Register (Offset = 1Ch) [Reset = 00000000h]

DOUT31_28 is shown in Table 15-21.

Return to the Summary Table.

Data Out 28 to 31
Alias register for byte access to each bit in DOUT47_0

Table 15-21. DOUT31_28 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO31	R/W	0h	Sets the state of the pin that is configured as DIO#31, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO30	R/W	0h	Sets the state of the pin that is configured as DIO#30, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO29	R/W	0h	Sets the state of the pin that is configured as DIO#29, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO28	R/W	0h	Sets the state of the pin that is configured as DIO#28, if the corresponding DOE47_0 bitfield is set.

15.10.2.9 DOUT35_32 Register (Offset = 20h) [Reset = 00000000h]

DOUT35_32 is shown in Table 15-22.

Return to the Summary Table.

Data Out 35 to 32
Alias register for byte access to each bit in DOUT47_0

Table 15-22. DOUT35_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO35	R/W	0h	Sets the state of the pin that is configured as DIO#35, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO34	R/W	0h	Sets the state of the pin that is configured as DIO#34, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO33	R/W	0h	Sets the state of the pin that is configured as DIO#33, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO32	R/W	0h	Sets the state of the pin that is configured as DIO#32, if the corresponding DOE47_0 bitfield is set.

15.10.2.10 DOUT39_36 Register (Offset = 24h) [Reset = 00000000h]

DOUT39_36 is shown in Table 15-23.

Return to the Summary Table.

Data Out 39 to 36
Alias register for byte access to each bit in DOUT47_0

Table 15-23. DOUT39_36 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO39	R/W	0h	Sets the state of the pin that is configured as DIO#39, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO38	R/W	0h	Sets the state of the pin that is configured as DIO#38, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO37	R/W	0h	Sets the state of the pin that is configured as DIO#37, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO36	R/W	0h	Sets the state of the pin that is configured as DIO#36, if the corresponding DOE47_0 bitfield is set.

15.10.2.11 DOUT43_40 Register (Offset = 28h) [Reset = 00000000h]

DOUT43_40 is shown in Table 15-24.

Return to the Summary Table.

Data Out 43 to 40
Alias register for byte access to each bit in DOUT47_0

Table 15-24. DOUT43_40 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO43	R/W	0h	Sets the state of the pin that is configured as DIO#43, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO42	R/W	0h	Sets the state of the pin that is configured as DIO#42, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO41	R/W	0h	Sets the state of the pin that is configured as DIO#41, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO40	R/W	0h	Sets the state of the pin that is configured as DIO#40, if the corresponding DOE47_0 bitfield is set.

15.10.2.12 DOUT47_44 Register (Offset = 2Ch) [Reset = 00000000h]

DOUT47_44 is shown in Table 15-25.

Return to the Summary Table.

Data Out 47 to 44
Alias register for byte access to each bit in DOUT47_0

Table 15-25. DOUT47_44 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24	DIO47	R/W	0h	Sets the state of the pin that is configured as DIO#47, if the corresponding DOE47_0 bitfield is set.
23-17	RESERVED	R	0h	Reserved
16	DIO46	R/W	0h	Sets the state of the pin that is configured as DIO#46, if the corresponding DOE47_0 bitfield is set.
15-9	RESERVED	R	0h	Reserved
8	DIO45	R/W	0h	Sets the state of the pin that is configured as DIO#45, if the corresponding DOE47_0 bitfield is set.
7-1	RESERVED	R	0h	Reserved
0	DIO44	R/W	0h	Sets the state of the pin that is configured as DIO#44, if the corresponding DOE47_0 bitfield is set.

15.10.2.13 DOUT31_0 Register (Offset = 80h) [Reset = 00000000h]

DOUT31_0 is shown in Table 15-26.

Return to the Summary Table.

Data Output for DIO 0 to 31

Table 15-26. DOUT31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W	0h	Data output for DIO 31
30	DIO30	R/W	0h	Data output for DIO 30
29	DIO29	R/W	0h	Data output for DIO 29
28	DIO28	R/W	0h	Data output for DIO 28
27	DIO27	R/W	0h	Data output for DIO 27
26	DIO26	R/W	0h	Data output for DIO 26
25	DIO25	R/W	0h	Data output for DIO 25
24	DIO24	R/W	0h	Data output for DIO 24
23	DIO23	R/W	0h	Data output for DIO 23
22	DIO22	R/W	0h	Data output for DIO 22
21	DIO21	R/W	0h	Data output for DIO 21
20	DIO20	R/W	0h	Data output for DIO 20
19	DIO19	R/W	0h	Data output for DIO 19
18	DIO18	R/W	0h	Data output for DIO 18
17	DIO17	R/W	0h	Data output for DIO 17
16	DIO16	R/W	0h	Data output for DIO 16
15	DIO15	R/W	0h	Data output for DIO 15
14	DIO14	R/W	0h	Data output for DIO 14
13	DIO13	R/W	0h	Data output for DIO 13
12	DIO12	R/W	0h	Data output for DIO 12
11	DIO11	R/W	0h	Data output for DIO 11
10	DIO10	R/W	0h	Data output for DIO 10
9	DIO9	R/W	0h	Data output for DIO 9
8	DIO8	R/W	0h	Data output for DIO 8
7	DIO7	R/W	0h	Data output for DIO 7
6	DIO6	R/W	0h	Data output for DIO 6
5	DIO5	R/W	0h	Data output for DIO 5
4	DIO4	R/W	0h	Data output for DIO 4
3	DIO3	R/W	0h	Data output for DIO 3
2	DIO2	R/W	0h	Data output for DIO 2
1	DIO1	R/W	0h	Data output for DIO 1
0	DIO0	R/W	0h	Data output for DIO 0

15.10.2.14 DOUT47_32 Register (Offset = 84h) [Reset = 00000000h]

DOUT47_32 is shown in Table 15-27.

Return to the Summary Table.

Data Output for DIO 0 to 31

Table 15-27. DOUT47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W	0h	Data output for DIO 47
14	DIO46	R/W	0h	Data output for DIO 46
13	DIO45	R/W	0h	Data output for DIO 45
12	DIO44	R/W	0h	Data output for DIO 44
11	DIO43	R/W	0h	Data output for DIO 43
10	DIO42	R/W	0h	Data output for DIO 42
9	DIO41	R/W	0h	Data output for DIO 41
8	DIO40	R/W	0h	Data output for DIO 40
7	DIO39	R/W	0h	Data output for DIO 39
6	DIO38	R/W	0h	Data output for DIO 38
5	DIO37	R/W	0h	Data output for DIO 37
4	DIO36	R/W	0h	Data output for DIO 36
3	DIO35	R/W	0h	Data output for DIO 35
2	DIO34	R/W	0h	Data output for DIO 34
1	DIO33	R/W	0h	Data output for DIO 33
0	DIO32	R/W	0h	Data output for DIO 32

15.10.2.15 DOUTSET31_0 Register (Offset = 90h) [Reset = 00000000h]

DOUTSET31_0 is shown in Table 15-28.

Return to the Summary Table.

Data Out Set
Writing 1 to a bit position sets the corresponding bit in the DOUT47_0 register

Table 15-28. DOUTSET31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W	0h	Set bit 31
30	DIO30	R/W	0h	Set bit 30
29	DIO29	R/W	0h	Set bit 29
28	DIO28	R/W	0h	Set bit 28
27	DIO27	R/W	0h	Set bit 27
26	DIO26	R/W	0h	Set bit 26
25	DIO25	R/W	0h	Set bit 25
24	DIO24	R/W	0h	Set bit 24
23	DIO23	R/W	0h	Set bit 23
22	DIO22	R/W	0h	Set bit 22
21	DIO21	R/W	0h	Set bit 21
20	DIO20	R/W	0h	Set bit 20
19	DIO19	R/W	0h	Set bit 19
18	DIO18	R/W	0h	Set bit 18
17	DIO17	R/W	0h	Set bit 17
16	DIO16	R/W	0h	Set bit 16
15	DIO15	R/W	0h	Set bit 15
14	DIO14	R/W	0h	Set bit 14
13	DIO13	R/W	0h	Set bit 13
12	DIO12	R/W	0h	Set bit 12
11	DIO11	R/W	0h	Set bit 11
10	DIO10	R/W	0h	Set bit 10
9	DIO9	R/W	0h	Set bit 9
8	DIO8	R/W	0h	Set bit 8
7	DIO7	R/W	0h	Set bit 7
6	DIO6	R/W	0h	Set bit 6
5	DIO5	R/W	0h	Set bit 5
4	DIO4	R/W	0h	Set bit 4
3	DIO3	R/W	0h	Set bit 3
2	DIO2	R/W	0h	Set bit 2
1	DIO1	R/W	0h	Set bit 1
0	DIO0	R/W	0h	Set bit 0

15.10.2.16 DOUTSET47_32 Register (Offset = 94h) [Reset = 00000000h]

DOUTSET47_32 is shown in Table 15-29.

Return to the Summary Table.

Data Out Set
Writing 1 to a bit position sets the corresponding bit in the DOUT47_0 register

Table 15-29. DOUTSET47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W	0h	Set bit 47
14	DIO46	R/W	0h	Set bit 46
13	DIO45	R/W	0h	Set bit 45
12	DIO44	R/W	0h	Set bit 44
11	DIO43	R/W	0h	Set bit 43
10	DIO42	R/W	0h	Set bit 42
9	DIO41	R/W	0h	Set bit 41
8	DIO40	R/W	0h	Set bit 40
7	DIO39	R/W	0h	Set bit 39
6	DIO38	R/W	0h	Set bit 38
5	DIO37	R/W	0h	Set bit 37
4	DIO36	R/W	0h	Set bit 36
3	DIO35	R/W	0h	Set bit 35
2	DIO34	R/W	0h	Set bit 34
1	DIO33	R/W	0h	Set bit 33
0	DIO32	R/W	0h	Set bit 32

15.10.2.17 DOUTCLR31_0 Register (Offset = A0h) [Reset = 00000000h]

DOUTCLR31_0 is shown in Table 15-30.

Return to the Summary Table.

Data Out Clear
Writing 1 to a bit position clears the corresponding bit in the DOUT47_0 register

Table 15-30. DOUTCLR31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W	0h	Clears bit 31
30	DIO30	R/W	0h	Clears bit 30
29	DIO29	R/W	0h	Clears bit 29
28	DIO28	R/W	0h	Clears bit 28
27	DIO27	R/W	0h	Clears bit 27
26	DIO26	R/W	0h	Clears bit 26
25	DIO25	R/W	0h	Clears bit 25
24	DIO24	R/W	0h	Clears bit 24
23	DIO23	R/W	0h	Clears bit 23
22	DIO22	R/W	0h	Clears bit 22
21	DIO21	R/W	0h	Clears bit 21
20	DIO20	R/W	0h	Clears bit 20
19	DIO19	R/W	0h	Clears bit 19
18	DIO18	R/W	0h	Clears bit 18
17	DIO17	R/W	0h	Clears bit 17
16	DIO16	R/W	0h	Clears bit 16
15	DIO15	R/W	0h	Clears bit 15
14	DIO14	R/W	0h	Clears bit 14
13	DIO13	R/W	0h	Clears bit 13
12	DIO12	R/W	0h	Clears bit 12
11	DIO11	R/W	0h	Clears bit 11
10	DIO10	R/W	0h	Clears bit 10
9	DIO9	R/W	0h	Clears bit 9
8	DIO8	R/W	0h	Clears bit 8
7	DIO7	R/W	0h	Clears bit 7
6	DIO6	R/W	0h	Clears bit 6
5	DIO5	R/W	0h	Clears bit 5
4	DIO4	R/W	0h	Clears bit 4
3	DIO3	R/W	0h	Clears bit 3
2	DIO2	R/W	0h	Clears bit 2
1	DIO1	R/W	0h	Clears bit 1
0	DIO0	R/W	0h	Clears bit 0

15.10.2.18 DOUTCLR47_32 Register (Offset = A4h) [Reset = 00000000h]

DOUTCLR47_32 is shown in Table 15-31.

Return to the Summary Table.

Data Out Clear
Writing 1 to a bit position clears the corresponding bit in the DOUT47_0 register

Table 15-31. DOUTCLR47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W	0h	Clears bit 47
14	DIO46	R/W	0h	Clears bit 46
13	DIO45	R/W	0h	Clears bit 45
12	DIO44	R/W	0h	Clears bit 44
11	DIO43	R/W	0h	Clears bit 43
10	DIO42	R/W	0h	Clears bit 42
9	DIO41	R/W	0h	Clears bit 41
8	DIO40	R/W	0h	Clears bit 40
7	DIO39	R/W	0h	Clears bit 39
6	DIO38	R/W	0h	Clears bit 38
5	DIO37	R/W	0h	Clears bit 37
4	DIO36	R/W	0h	Clears bit 36
3	DIO35	R/W	0h	Clears bit 35
2	DIO34	R/W	0h	Clears bit 34
1	DIO33	R/W	0h	Clears bit 33
0	DIO32	R/W	0h	Clears bit 32

15.10.2.19 DOUTTGL31_0 Register (Offset = B0h) [Reset = 00000000h]

DOUTTGL31_0 is shown in Table 15-32.

Return to the Summary Table.

Data Out Toggle
Writing 1 to a bit position will invert the corresponding DIO output.

Table 15-32. DOUTTGL31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W	0h	Toggles bit 31
30	DIO30	R/W	0h	Toggles bit 30
29	DIO29	R/W	0h	Toggles bit 29
28	DIO28	R/W	0h	Toggles bit 28
27	DIO27	R/W	0h	Toggles bit 27
26	DIO26	R/W	0h	Toggles bit 26
25	DIO25	R/W	0h	Toggles bit 25
24	DIO24	R/W	0h	Toggles bit 24
23	DIO23	R/W	0h	Toggles bit 23
22	DIO22	R/W	0h	Toggles bit 22
21	DIO21	R/W	0h	Toggles bit 21
20	DIO20	R/W	0h	Toggles bit 20
19	DIO19	R/W	0h	Toggles bit 19
18	DIO18	R/W	0h	Toggles bit 18
17	DIO17	R/W	0h	Toggles bit 17
16	DIO16	R/W	0h	Toggles bit 16
15	DIO15	R/W	0h	Toggles bit 15
14	DIO14	R/W	0h	Toggles bit 14
13	DIO13	R/W	0h	Toggles bit 13
12	DIO12	R/W	0h	Toggles bit 12
11	DIO11	R/W	0h	Toggles bit 11
10	DIO10	R/W	0h	Toggles bit 10
9	DIO9	R/W	0h	Toggles bit 9
8	DIO8	R/W	0h	Toggles bit 8
7	DIO7	R/W	0h	Toggles bit 7
6	DIO6	R/W	0h	Toggles bit 6
5	DIO5	R/W	0h	Toggles bit 5
4	DIO4	R/W	0h	Toggles bit 4
3	DIO3	R/W	0h	Toggles bit 3
2	DIO2	R/W	0h	Toggles bit 2
1	DIO1	R/W	0h	Toggles bit 1
0	DIO0	R/W	0h	Toggles bit 0

15.10.2.20 DOUTTGL47_32 Register (Offset = B4h) [Reset = 00000000h]

DOUTTGL47_32 is shown in Table 15-33.

Return to the Summary Table.

Data Out Toggle
Writing 1 to a bit position will invert the corresponding DIO output.

Table 15-33. DOUTTGL47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W	0h	Toggles bit 47
14	DIO46	R/W	0h	Toggles bit 46
13	DIO45	R/W	0h	Toggles bit 45
12	DIO44	R/W	0h	Toggles bit 44
11	DIO43	R/W	0h	Toggles bit 43
10	DIO42	R/W	0h	Toggles bit 42
9	DIO41	R/W	0h	Toggles bit 41
8	DIO40	R/W	0h	Toggles bit 40
7	DIO39	R/W	0h	Toggles bit 39
6	DIO38	R/W	0h	Toggles bit 38
5	DIO37	R/W	0h	Toggles bit 37
4	DIO36	R/W	0h	Toggles bit 36
3	DIO35	R/W	0h	Toggles bit 35
2	DIO34	R/W	0h	Toggles bit 34
1	DIO33	R/W	0h	Toggles bit 33
0	DIO32	R/W	0h	Toggles bit 32

15.10.2.21 DIN31_0 Register (Offset = C0h) [Reset = 00000000h]

DIN31_0 is shown in Table 15-34.

Return to the Summary Table.

Data Input from DIO 0 to 31

Table 15-34. DIN31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R	0h	Data input from DIO 31
30	DIO30	R	0h	Data input from DIO 30
29	DIO29	R	0h	Data input from DIO 29
28	DIO28	R	0h	Data input from DIO 28
27	DIO27	R	0h	Data input from DIO 27
26	DIO26	R	0h	Data input from DIO 26
25	DIO25	R	0h	Data input from DIO 25
24	DIO24	R	0h	Data input from DIO 24
23	DIO23	R	0h	Data input from DIO 23
22	DIO22	R	0h	Data input from DIO 22
21	DIO21	R	0h	Data input from DIO 21
20	DIO20	R	0h	Data input from DIO 20
19	DIO19	R	0h	Data input from DIO 19
18	DIO18	R	0h	Data input from DIO 18
17	DIO17	R	0h	Data input from DIO 17
16	DIO16	R	0h	Data input from DIO 16
15	DIO15	R	0h	Data input from DIO 15
14	DIO14	R	0h	Data input from DIO 14
13	DIO13	R	0h	Data input from DIO 13
12	DIO12	R	0h	Data input from DIO 12
11	DIO11	R	0h	Data input from DIO 11
10	DIO10	R	0h	Data input from DIO 10
9	DIO9	R	0h	Data input from DIO 9
8	DIO8	R	0h	Data input from DIO 8
7	DIO7	R	0h	Data input from DIO 7
6	DIO6	R	0h	Data input from DIO 6
5	DIO5	R	0h	Data input from DIO 5
4	DIO4	R	0h	Data input from DIO 4
3	DIO3	R	0h	Data input from DIO 3
2	DIO2	R	0h	Data input from DIO 2
1	DIO1	R	0h	Data input from DIO 1
0	DIO0	R	0h	Data input from DIO 0

15.10.2.22 DIN47_32 Register (Offset = C4h) [Reset = 00000000h]

DIN47_32 is shown in Table 15-35.

Return to the Summary Table.

Data Input from DIO 32 to 47

Table 15-35. DIN47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R	0h	Data input from DIO 47
14	DIO46	R	0h	Data input from DIO 46
13	DIO45	R	0h	Data input from DIO 45
12	DIO44	R	0h	Data input from DIO 44
11	DIO43	R	0h	Data input from DIO 43
10	DIO42	R	0h	Data input from DIO 42
9	DIO41	R	0h	Data input from DIO 41
8	DIO40	R	0h	Data input from DIO 40
7	DIO39	R	0h	Data input from DIO 39
6	DIO38	R	0h	Data input from DIO 38
5	DIO37	R	0h	Data input from DIO 37
4	DIO36	R	0h	Data input from DIO 36
3	DIO35	R	0h	Data input from DIO 35
2	DIO34	R	0h	Data input from DIO 34
1	DIO33	R	0h	Data input from DIO 33
0	DIO32	R	0h	Data input from DIO 32

15.10.2.23 DOE31_0 Register (Offset = D0h) [Reset = 00000000h]

DOE31_0 is shown in Table 15-36.

Return to the Summary Table.

Data Output Enable for DIO 0 to 31

Table 15-36. DOE31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W	0h	Data output enable for DIO 31
30	DIO30	R/W	0h	Data output enable for DIO 30
29	DIO29	R/W	0h	Data output enable for DIO 29
28	DIO28	R/W	0h	Data output enable for DIO 28
27	DIO27	R/W	0h	Data output enable for DIO 27
26	DIO26	R/W	0h	Data output enable for DIO 26
25	DIO25	R/W	0h	Data output enable for DIO 25
24	DIO24	R/W	0h	Data output enable for DIO 24
23	DIO23	R/W	0h	Data output enable for DIO 23
22	DIO22	R/W	0h	Data output enable for DIO 22
21	DIO21	R/W	0h	Data output enable for DIO 21
20	DIO20	R/W	0h	Data output enable for DIO 20
19	DIO19	R/W	0h	Data output enable for DIO 19
18	DIO18	R/W	0h	Data output enable for DIO 18
17	DIO17	R/W	0h	Data output enable for DIO 17
16	DIO16	R/W	0h	Data output enable for DIO 16
15	DIO15	R/W	0h	Data output enable for DIO 15
14	DIO14	R/W	0h	Data output enable for DIO 14
13	DIO13	R/W	0h	Data output enable for DIO 13
12	DIO12	R/W	0h	Data output enable for DIO 12
11	DIO11	R/W	0h	Data output enable for DIO 11
10	DIO10	R/W	0h	Data output enable for DIO 10
9	DIO9	R/W	0h	Data output enable for DIO 9
8	DIO8	R/W	0h	Data output enable for DIO 8
7	DIO7	R/W	0h	Data output enable for DIO 7
6	DIO6	R/W	0h	Data output enable for DIO 6
5	DIO5	R/W	0h	Data output enable for DIO 5
4	DIO4	R/W	0h	Data output enable for DIO 4
3	DIO3	R/W	0h	Data output enable for DIO 3
2	DIO2	R/W	0h	Data output enable for DIO 2
1	DIO1	R/W	0h	Data output enable for DIO 1
0	DIO0	R/W	0h	Data output enable for DIO 0

15.10.2.24 DOE47_32 Register (Offset = D4h) [Reset = 00000000h]

DOE47_32 is shown in Table 15-37.

Return to the Summary Table.

Data Output Enable for DIO 32 to 47

Table 15-37. DOE47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W	0h	Data output enable for DIO 47
14	DIO46	R/W	0h	Data output enable for DIO 46
13	DIO45	R/W	0h	Data output enable for DIO 45
12	DIO44	R/W	0h	Data output enable for DIO 44
11	DIO43	R/W	0h	Data output enable for DIO 43
10	DIO42	R/W	0h	Data output enable for DIO 42
9	DIO41	R/W	0h	Data output enable for DIO 41
8	DIO40	R/W	0h	Data output enable for DIO 40
7	DIO39	R/W	0h	Data output enable for DIO 39
6	DIO38	R/W	0h	Data output enable for DIO 38
5	DIO37	R/W	0h	Data output enable for DIO 37
4	DIO36	R/W	0h	Data output enable for DIO 36
3	DIO35	R/W	0h	Data output enable for DIO 35
2	DIO34	R/W	0h	Data output enable for DIO 34
1	DIO33	R/W	0h	Data output enable for DIO 33
0	DIO32	R/W	0h	Data output enable for DIO 32

15.10.2.25 EVFLAGS31_0 Register (Offset = E0h) [Reset = 00000000h]

EVFLAGS31_0 is shown in Table 15-38.

Return to the Summary Table.

Event Register for DIO 0 to 31
Reading this registers will return 1 for triggered event and 0 for non-triggered events.
Writing a 1 to a bit field will clear the event.
The configuration of events is done inside MCU IOC, e.g. events for DIO #0 is configured in IOC:IOCFG0.EDGE_DET and IOC:IOCFG0.EDGE_IRQ_EN.

Table 15-38. EVFLAGS31_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DIO31	R/W1C	0h	Event for DIO 31
30	DIO30	R/W1C	0h	Event for DIO 30
29	DIO29	R/W1C	0h	Event for DIO 29
28	DIO28	R/W1C	0h	Event for DIO 28
27	DIO27	R/W1C	0h	Event for DIO 27
26	DIO26	R/W1C	0h	Event for DIO 26
25	DIO25	R/W1C	0h	Event for DIO 25
24	DIO24	R/W1C	0h	Event for DIO 24
23	DIO23	R/W1C	0h	Event for DIO 23
22	DIO22	R/W1C	0h	Event for DIO 22
21	DIO21	R/W1C	0h	Event for DIO 21
20	DIO20	R/W1C	0h	Event for DIO 20
19	DIO19	R/W1C	0h	Event for DIO 19
18	DIO18	R/W1C	0h	Event for DIO 18
17	DIO17	R/W1C	0h	Event for DIO 17
16	DIO16	R/W1C	0h	Event for DIO 16
15	DIO15	R/W1C	0h	Event for DIO 15
14	DIO14	R/W1C	0h	Event for DIO 14
13	DIO13	R/W1C	0h	Event for DIO 13
12	DIO12	R/W1C	0h	Event for DIO 12
11	DIO11	R/W1C	0h	Event for DIO 11
10	DIO10	R/W1C	0h	Event for DIO 10
9	DIO9	R/W1C	0h	Event for DIO 9
8	DIO8	R/W1C	0h	Event for DIO 8
7	DIO7	R/W1C	0h	Event for DIO 7
6	DIO6	R/W1C	0h	Event for DIO 6
5	DIO5	R/W1C	0h	Event for DIO 5
4	DIO4	R/W1C	0h	Event for DIO 4
3	DIO3	R/W1C	0h	Event for DIO 3
2	DIO2	R/W1C	0h	Event for DIO 2
1	DIO1	R/W1C	0h	Event for DIO 1
0	DIO0	R/W1C	0h	Event for DIO 0

15.10.2.26 EVFLAGS47_32 Register (Offset = E4h) [Reset = 00000000h]

EVFLAGS47_32 is shown in Table 15-39.

Return to the Summary Table.

Event Register for DIO 32 to 47
Reading this registers will return 1 for triggered event and 0 for non-triggered events.
Writing a 1 to a bit field will clear the event.
The configuration of events is done inside MCU IOC, e.g. events for DIO #0 is configured in IOC:IOCFG0.EDGE_DET and IOC:IOCFG0.EDGE_IRQ_EN.

Table 15-39. EVFLAGS47_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	DIO47	R/W1C	0h	Event for DIO 47
14	DIO46	R/W1C	0h	Event for DIO 46
13	DIO45	R/W1C	0h	Event for DIO 45
12	DIO44	R/W1C	0h	Event for DIO 44
11	DIO43	R/W1C	0h	Event for DIO 43
10	DIO42	R/W1C	0h	Event for DIO 42
9	DIO41	R/W1C	0h	Event for DIO 41
8	DIO40	R/W1C	0h	Event for DIO 40
7	DIO39	R/W1C	0h	Event for DIO 39
6	DIO38	R/W1C	0h	Event for DIO 38
5	DIO37	R/W1C	0h	Event for DIO 37
4	DIO36	R/W1C	0h	Event for DIO 36
3	DIO35	R/W1C	0h	Event for DIO 35
2	DIO34	R/W1C	0h	Event for DIO 34
1	DIO33	R/W1C	0h	Event for DIO 33
0	DIO32	R/W1C	0h	Event for DIO 32

15.10.3 IOC Registers

Table 15-40 lists the memory-mapped registers for the IOC registers. All register offset addresses not listed in Table 15-40 should be considered as reserved locations and the register contents should not be modified.

Table 15-40. IOC Registers

Offset	Acronym	Register Name	Section
0h	IOCFG0	Configuration of DIO0	Section 15.10.3.1
4h	IOCFG1	Configuration of DIO1	Section 15.10.3.2
8h	IOCFG2	Configuration of DIO2	Section 15.10.3.3
Ch	IOCFG3	Configuration of DIO3	Section 15.10.3.4
10h	IOCFG4	Configuration of DIO4	Section 15.10.3.5
14h	IOCFG5	Configuration of DIO5	Section 15.10.3.6
18h	IOCFG6	Configuration of DIO6	Section 15.10.3.7
1Ch	IOCFG7	Configuration of DIO7	Section 15.10.3.8
20h	IOCFG8	Configuration of DIO8	Section 15.10.3.9
24h	IOCFG9	Configuration of DIO9	Section 15.10.3.10
28h	IOCFG10	Configuration of DIO10	Section 15.10.3.11
2Ch	IOCFG11	Configuration of DIO11	Section 15.10.3.12
30h	IOCFG12	Configuration of DIO12	Section 15.10.3.13
34h	IOCFG13	Configuration of DIO13	Section 15.10.3.14
38h	IOCFG14	Configuration of DIO14	Section 15.10.3.15
3Ch	IOCFG15	Configuration of DIO15	Section 15.10.3.16
40h	IOCFG16	Configuration of DIO16	Section 15.10.3.17
44h	IOCFG17	Configuration of DIO17	Section 15.10.3.18
48h	IOCFG18	Configuration of DIO18	Section 15.10.3.19
4Ch	IOCFG19	Configuration of DIO19	Section 15.10.3.20
50h	IOCFG20	Configuration of DIO20	Section 15.10.3.21
54h	IOCFG21	Configuration of DIO21	Section 15.10.3.22
58h	IOCFG22	Configuration of DIO22	Section 15.10.3.23
5Ch	IOCFG23	Configuration of DIO23	Section 15.10.3.24
60h	IOCFG24	Configuration of DIO24	Section 15.10.3.25
64h	IOCFG25	Configuration of DIO25	Section 15.10.3.26
68h	IOCFG26	Configuration of DIO26	Section 15.10.3.27
6Ch	IOCFG27	Configuration of DIO27	Section 15.10.3.28
70h	IOCFG28	Configuration of DIO28	Section 15.10.3.29
74h	IOCFG29	Configuration of DIO29	Section 15.10.3.30
78h	IOCFG30	Configuration of DIO30	Section 15.10.3.31
7Ch	IOCFG31	Configuration of DIO31	Section 15.10.3.32
80h	IOCFG32	Configuration of DIO32	Section 15.10.3.33
84h	IOCFG33	Configuration of DIO33	Section 15.10.3.34
88h	IOCFG34	Configuration of DIO34	Section 15.10.3.35
8Ch	IOCFG35	Configuration of DIO35	Section 15.10.3.36
90h	IOCFG36	Configuration of DIO36	Section 15.10.3.37
94h	IOCFG37	Configuration of DIO37	Section 15.10.3.38
98h	IOCFG38	Configuration of DIO38	Section 15.10.3.39
9Ch	IOCFG39	Configuration of DIO39	Section 15.10.3.40
A0h	IOCFG40	Configuration of DIO40	Section 15.10.3.41
A4h	IOCFG41	Configuration of DIO41	Section 15.10.3.42

Table 15-40. IOC Registers (continued)

Offset	Acronym	Register Name	Section
A8h	IOCFG42	Configuration of DIO42	Section 15.10.3.43
ACh	IOCFG43	Configuration of DIO43	Section 15.10.3.44
B0h	IOCFG44	Configuration of DIO44	Section 15.10.3.45
B4h	IOCFG45	Configuration of DIO45	Section 15.10.3.46
B8h	IOCFG46	Configuration of DIO46	Section 15.10.3.47
BCh	IOCFG47	Configuration of DIO47	Section 15.10.3.48

Complex bit access types are encoded to fit into small table cells. Table 15-41 shows the codes that are used for access types in this section.

Table 15-41. IOC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

15.10.3.1 IOCFG0 Register (Offset = 0h) [Reset = 0000C000h]

IOCFG0 is shown in Table 15-42.

Return to the Summary Table.

Configuration of DIO0

Table 15-42. IOCFG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / outut 7h = OPENSRC_INV : Open Source Inverted input/output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-42. IOCFG0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-42. IOCFG0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO0 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-42. IOCFG0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.2 IOCFG1 Register (Offset = 4h) [Reset = 0000C000h]

IOCFG1 is shown in Table 15-43.

Return to the Summary Table.

Configuration of DIO1

Table 15-43. IOCFG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-43. IOCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-43. IOCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO1 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-43. IOCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.3 IOCFG2 Register (Offset = 8h) [Reset = 0000C000h]

IOCFG2 is shown in Table 15-44.

Return to the Summary Table.

Configuration of DIO2

Table 15-44. IOCFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-44. IOCFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-44. IOCFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO2 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-44. IOCFG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.4 IOCFG3 Register (Offset = Ch) [Reset = 0000C000h]

IOCFG3 is shown in Table 15-45.

Return to the Summary Table.

Configuration of DIO3

Table 15-45. IOCFG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-45. IOCFG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-45. IOCFG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO3 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-45. IOCFG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.5 IOCFG4 Register (Offset = 10h) [Reset = 0000C000h]

IOCFG4 is shown in Table 15-46.

Return to the Summary Table.

Configuration of DIO4

Table 15-46. IOCFG4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-46. IOCFG4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-46. IOCFG4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO4 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-46. IOCFG4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.6 IOCFG5 Register (Offset = 14h) [Reset = 0000C000h]

IOCFG5 is shown in Table 15-47.

Return to the Summary Table.

Configuration of DIO5

Table 15-47. IOCFG5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-47. IOCFG5 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-47. IOCFG5 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO5 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-47. IOCFG5 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.7 IOCFG6 Register (Offset = 18h) [Reset = 0000C000h]

IOCFG6 is shown in Table 15-48.

Return to the Summary Table.

Configuration of DIO6

Table 15-48. IOCFG6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-48. IOCFG6 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-48. IOCFG6 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO6 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-48. IOCFG6 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.8 IOCFG7 Register (Offset = 1Ch) [Reset = 0000C000h]

IOCFG7 is shown in Table 15-49.

Return to the Summary Table.

Configuration of DIO7

Table 15-49. IOCFG7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-49. IOCFG7 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-49. IOCFG7 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO7 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-49. IOCFG7 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.9 IOCFG8 Register (Offset = 20h) [Reset = 0000C000h]

IOCFG8 is shown in Table 15-50.

Return to the Summary Table.

Configuration of DIO8

Table 15-50. IOCFG8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-50. IOCFG8 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-50. IOCFG8 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO8 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-50. IOCFG8 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.10 IOCFG9 Register (Offset = 24h) [Reset = 0000C000h]

IOCFG9 is shown in Table 15-51.

Return to the Summary Table.

Configuration of DIO9

Table 15-51. IOCFG9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-51. IOCFG9 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-51. IOCFG9 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO9 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-51. IOCFG9 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.11 IOCFG10 Register (Offset = 28h) [Reset = 0000C000h]

IOCFG10 is shown in Table 15-52.

Return to the Summary Table.

Configuration of DIO10

Table 15-52. IOCFG10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-52. IOCFG10 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-52. IOCFG10 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO10 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-52. IOCFG10 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.12 IOCFG11 Register (Offset = 2Ch) [Reset = 0000C000h]

IOCFG11 is shown in Table 15-53.

Return to the Summary Table.

Configuration of DIO11

Table 15-53. IOCFG11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-53. IOCFG11 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-53. IOCFG11 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO11 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-53. IOCFG11 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.13 IOCFG12 Register (Offset = 30h) [Reset = 0000C000h]

IOCFG12 is shown in Table 15-54.

Return to the Summary Table.

Configuration of DIO12

Table 15-54. IOCFG12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-54. IOCFG12 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-54. IOCFG12 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO12 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-54. IOCFG12 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.14 IOCFG13 Register (Offset = 34h) [Reset = 0000C000h]

IOCFG13 is shown in Table 15-55.

Return to the Summary Table.

Configuration of DIO13

Table 15-55. IOCFG13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-55. IOCFG13 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-55. IOCFG13 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO13 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-55. IOCFG13 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.15 IOCFG14 Register (Offset = 38h) [Reset = 0000C000h]

IOCFG14 is shown in Table 15-56.

Return to the Summary Table.

Configuration of DIO14

Table 15-56. IOCFG14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-56. IOCFG14 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-56. IOCFG14 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO14 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-56. IOCFG14 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.16 IOCFG15 Register (Offset = 3Ch) [Reset = 0000C000h]

IOCFG15 is shown in Table 15-57.

Return to the Summary Table.

Configuration of DIO15

Table 15-57. IOCFG15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-57. IOCFG15 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-57. IOCFG15 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO15 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-57. IOCFG15 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.17 IOCFG16 Register (Offset = 40h) [Reset = 0008C000h]

IOCFG16 is shown in Table 15-58.

Return to the Summary Table.

Configuration of DIO16

Table 15-58. IOCFG16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-58. IOCFG16 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-58. IOCFG16 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO16 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-58. IOCFG16 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.18 IOCFG17 Register (Offset = 44h) [Reset = 0010C000h]

IOCFG17 is shown in Table 15-59.

Return to the Summary Table.

Configuration of DIO17

Table 15-59. IOCFG17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-59. IOCFG17 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-59. IOCFG17 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO17 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-59. IOCFG17 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.19 IOCFG18 Register (Offset = 48h) [Reset = 0000C000h]

IOCFG18 is shown in Table 15-60.

Return to the Summary Table.

Configuration of DIO18

Table 15-60. IOCFG18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-60. IOCFG18 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-60. IOCFG18 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO18 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-60. IOCFG18 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.20 IOCFG19 Register (Offset = 4Ch) [Reset = 0000C000h]

IOCFG19 is shown in Table 15-61.

Return to the Summary Table.

Configuration of DIO19

Table 15-61. IOCFG19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-61. IOCFG19 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-61. IOCFG19 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO19 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-61. IOCFG19 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.21 IOCFG20 Register (Offset = 50h) [Reset = 0000C000h]

IOCFG20 is shown in Table 15-62.

Return to the Summary Table.

Configuration of DIO20

Table 15-62. IOCFG20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-62. IOCFG20 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-62. IOCFG20 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO20 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-62. IOCFG20 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.22 IOCFG21 Register (Offset = 54h) [Reset = 0000C000h]

IOCFG21 is shown in Table 15-63.

Return to the Summary Table.

Configuration of DIO21

Table 15-63. IOCFG21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-63. IOCFG21 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-63. IOCFG21 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO21 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-63. IOCFG21 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.23 IOCFG22 Register (Offset = 58h) [Reset = 0000C000h]

IOCFG22 is shown in Table 15-64.

Return to the Summary Table.

Configuration of DIO22

Table 15-64. IOCFG22 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-64. IOCFG22 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-64. IOCFG22 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO22 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-64. IOCFG22 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.24 IOCFG23 Register (Offset = 5Ch) [Reset = 0000C000h]

IOCFG23 is shown in Table 15-65.

Return to the Summary Table.

Configuration of DIO23

Table 15-65. IOCFG23 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-65. IOCFG23 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-65. IOCFG23 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO23 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-65. IOCFG23 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.25 IOCFG24 Register (Offset = 60h) [Reset = 0000C000h]

IOCFG24 is shown in Table 15-66.

Return to the Summary Table.

Configuration of DIO24

Table 15-66. IOCFG24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-66. IOCFG24 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-66. IOCFG24 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO24 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-66. IOCFG24 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.26 IOCFG25 Register (Offset = 64h) [Reset = 0000C000h]

IOCFG25 is shown in Table 15-67.

Return to the Summary Table.

Configuration of DIO25

Table 15-67. IOCFG25 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-67. IOCFG25 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-67. IOCFG25 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO25 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-67. IOCFG25 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.27 IOCFG26 Register (Offset = 68h) [Reset = 0000C000h]

IOCFG26 is shown in Table 15-68.

Return to the Summary Table.

Configuration of DIO26

Table 15-68. IOCFG26 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-68. IOCFG26 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-68. IOCFG26 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO26 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-68. IOCFG26 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.28 IOCFG27 Register (Offset = 6Ch) [Reset = 0000C000h]

IOCFG27 is shown in Table 15-69.

Return to the Summary Table.

Configuration of DIO27

Table 15-69. IOCFG27 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-69. IOCFG27 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-69. IOCFG27 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO27 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-69. IOCFG27 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.29 IOCFG28 Register (Offset = 70h) [Reset = 0000C000h]

IOCFG28 is shown in Table 15-70.

Return to the Summary Table.

Configuration of DIO28

Table 15-70. IOCFG28 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-70. IOCFG28 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-70. IOCFG28 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO28 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-70. IOCFG28 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.30 IOCFG29 Register (Offset = 74h) [Reset = 0000C000h]

IOCFG29 is shown in Table 15-71.

Return to the Summary Table.

Configuration of DIO29

Table 15-71. IOCFG29 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-71. IOCFG29 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-71. IOCFG29 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO29 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-71. IOCFG29 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.31 IOCFG30 Register (Offset = 78h) [Reset = 0000C000h]

IOCFG30 is shown in Table 15-72.

Return to the Summary Table.

Configuration of DIO30

Table 15-72. IOCFG30 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-72. IOCFG30 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-72. IOCFG30 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO30 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-72. IOCFG30 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.32 IOCFG31 Register (Offset = 7Ch) [Reset = 0000C000h]

IOCFG31 is shown in Table 15-73.

Return to the Summary Table.

Configuration of DIO31

Table 15-73. IOCFG31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-73. IOCFG31 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-73. IOCFG31 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO31 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 8h = AUX IO 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK

Table 15-73. IOCFG31 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				25h = I2S_AD0 : I2S Data 0 26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.33 IOCFG32 Register (Offset = 80h) [Reset = 0000C000h]

IOCFG32 is shown in Table 15-74.

Return to the Summary Table.

Configuration of DIO32

Table 15-74. IOCFG32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-74. IOCFG32 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-74. IOCFG32 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO32 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-74. IOCFG32 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.34 IOCFG33 Register (Offset = 84h) [Reset = 0000C000h]

IOCFG33 is shown in Table 15-75.

Return to the Summary Table.

Configuration of DIO33

Table 15-75. IOCFG33 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-75. IOCFG33 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-75. IOCFG33 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO33 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-75. IOCFG33 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.35 IOCFG34 Register (Offset = 88h) [Reset = 0000C000h]

IOCFG34 is shown in Table 15-76.

Return to the Summary Table.

Configuration of DIO34

Table 15-76. IOCFG34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-76. IOCFG34 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-76. IOCFG34 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO34 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-76. IOCFG34 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.36 IOCFG35 Register (Offset = 8Ch) [Reset = 0000C000h]

IOCFG35 is shown in Table 15-77.

Return to the Summary Table.

Configuration of DIO35

Table 15-77. IOCFG35 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-77. IOCFG35 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-77. IOCFG35 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO35 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-77. IOCFG35 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.37 IOCFG36 Register (Offset = 90h) [Reset = 0000C000h]

IOCFG36 is shown in Table 15-78.

Return to the Summary Table.

Configuration of DIO36

Table 15-78. IOCFG36 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-78. IOCFG36 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-78. IOCFG36 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO36 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-78. IOCFG36 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.38 IOCFG37 Register (Offset = 94h) [Reset = 0000C000h]

IOCFG37 is shown in Table 15-79.

Return to the Summary Table.

Configuration of DIO37

Table 15-79. IOCFG37 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-79. IOCFG37 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-79. IOCFG37 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO37 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-79. IOCFG37 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.39 IOCFG38 Register (Offset = 98h) [Reset = 0000C000h]

IOCFG38 is shown in Table 15-80.

Return to the Summary Table.

Configuration of DIO38

Table 15-80. IOCFG38 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-80. IOCFG38 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-80. IOCFG38 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO38 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-80. IOCFG38 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.40 IOCFG39 Register (Offset = 9Ch) [Reset = 0000C000h]

IOCFG39 is shown in Table 15-81.

Return to the Summary Table.

Configuration of DIO39

Table 15-81. IOCFG39 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-81. IOCFG39 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-81. IOCFG39 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO39 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-81. IOCFG39 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.41 IOCFG40 Register (Offset = A0h) [Reset = 0000C000h]

IOCFG40 is shown in Table 15-82.

Return to the Summary Table.

Configuration of DIO40

Table 15-82. IOCFG40 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-82. IOCFG40 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-82. IOCFG40 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO40 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-82. IOCFG40 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.42 IOCFG41 Register (Offset = A4h) [Reset = 0000C000h]

IOCFG41 is shown in Table 15-83.

Return to the Summary Table.

Configuration of DIO41

Table 15-83. IOCFG41 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-83. IOCFG41 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-83. IOCFG41 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO41 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-83. IOCFG41 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.43 IOCFG42 Register (Offset = A8h) [Reset = 0000C000h]

IOCFG42 is shown in Table 15-84.

Return to the Summary Table.

Configuration of DIO42

Table 15-84. IOCFG42 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-84. IOCFG42 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-84. IOCFG42 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO42 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-84. IOCFG42 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.44 IOCFG43 Register (Offset = ACh) [Reset = 0000C000h]

IOCFG43 is shown in Table 15-85.

Return to the Summary Table.

Configuration of DIO43

Table 15-85. IOCFG43 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-18	RESERVED	R	0h	Reserved
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection

Table 15-85. IOCFG43 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-85. IOCFG43 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO43 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-85. IOCFG43 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.45 IOCFG44 Register (Offset = B0h) [Reset = 0000C000h]

IOCFG44 is shown in Table 15-86.

Return to the Summary Table.

Configuration of DIO44

Table 15-86. IOCFG44 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-86. IOCFG44 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-86. IOCFG44 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO44 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-86. IOCFG44 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.46 IOCFG45 Register (Offset = B4h) [Reset = 0000C000h]

IOCFG45 is shown in Table 15-87.

Return to the Summary Table.

Configuration of DIO45

Table 15-87. IOCFG45 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-87. IOCFG45 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-87. IOCFG45 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO45 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-87. IOCFG45 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.47 IOCFG46 Register (Offset = B8h) [Reset = 0000C000h]

IOCFG46 is shown in Table 15-88.

Return to the Summary Table.

Configuration of DIO46

Table 15-88. IOCFG46 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-88. IOCFG46 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-88. IOCFG46 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO46 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-88. IOCFG46 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

15.10.3.48 IOCFG47 Register (Offset = BCh) [Reset = 0000C000h]

IOCFG47 is shown in Table 15-89.

Return to the Summary Table.

Configuration of DIO47

Table 15-89. IOCFG47 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOEV_MCU_WU_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert MCU_WU event 1: Input edge detection asserts MCU_WU event
30	HYST_EN	R/W	0h	0: Input hysteresis disable 1: Input hysteresis enable
29	IE	R/W	0h	0: Input disabled 1: Input enabled Note: If IO is configured for AUX PORT_ID = 0x08, the enable will be ignored.
28-27	WU_CFG	R/W	0h	If DIO is configured GPIO or non-AON peripheral signals, PORT_ID 0x00 or >0x08: 00: No wake-up 01: No wake-up 10: Wakes up from shutdown if this pad is going low. 11: Wakes up from shutdown if this pad is going high. If IO is configured for AON peripheral signals or AUX PORT_ID 0x01-0x08, this register only sets wakeup enable or not. 00, 01: Wakeup disabled 10, 11: Wakeup enabled Polarity is controlled from AON registers. Note:When the MSB is set, the IOC will deactivate the output enable for the DIO.
26-24	IOMODE	R/W	0h	IO Mode Not applicable for IO configured for AON periph. signals and AUX PORT_ID 0x01-0x08 AUX has its own open_source/drain configuration. 0x2: Reserved. Undefined behavior. 0x3: Reserved. Undefined behavior. 0h = NORMAL : Normal input / output 1h = INV : Inverted input / ouput 4h = OPENDR : Open Drain, Normal input / output 5h = OPENDR_INV : Open Drain Inverted input / output 6h = OPENSRC : Open Source Normal input / output 7h = OPENSRC_INV : Open Source Inverted input / output
23	IOEV_AON_PROG2_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG2 event 1: Input edge detection asserts AON_PROG2 event
22	IOEV_AON_PROG1_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG1 event 1: Input edge detection asserts AON_PROG1 event
21	IOEV_AON_PROG0_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert AON_PROG0 event 1: Input edge detection asserts AON_PROG0 event
20-19	RESERVED	R	0h	Reserved
18	EDGE_IRQ_EN	R/W	0h	0: No interrupt generation 1: Enable interrupt generation for this IO (Only effective if EDGE_DET is enabled)

Table 15-89. IOCFG47 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-16	EDGE_DET	R/W	0h	Enable generation of edge detection events on this IO 0h = NONE : No edge detection 1h = Negative edge detection 2h = Positive edge detection 3h = Positive and negative edge detection
15-14	PULL_CTL	R/W	3h	Pull control 1h = DWN : Pull down 2h = UP : Pull up 3h = DIS : No pull
13	SLEW_RED	R/W	0h	0: Normal slew rate 1: Enables reduced slew rate in output driver.
12-11	IOCURR	R/W	0h	Selects IO current mode of this IO. 0h = 2MA : Low-Current (LC) mode: Min 2 mA when IOSTR is set to AUTO 1h = 4MA : High-Current (HC) mode: Min 4 mA when IOSTR is set to AUTO 2h = 4_8MA : Extended-Current (EC) mode: Min 8 mA for double drive strength IOs (min 4 mA for normal IOs) when IOSTR is set to AUTO
10-9	IOSTR	R/W	0h	Select source for drive strength control of this IO. This setting controls the drive strength of the Low-Current (LC) mode. Higher drive strength can be selected in IOCURR 0h = Automatic drive strength, controlled by AON BATMON based on battery voltage. (min 2 mA @VDDS) 1h = Minimum drive strength, controlled by AON_IOC:IOSTRMIN (min 2 mA @3.3V with default values) 2h = MED : Medium drive strength, controlled by AON_IOC:IOSTRMED (min 2 mA @2.5V with default values) 3h = Maximum drive strength, controlled by AON_IOC:IOSTRMAX (min 2 mA @1.8V with default values)
8	IOEV_RTC_EN	R/W	0h	Event asserted by this IO when edge detection is enabled 0: Input edge detection does not assert RTC event 1: Input edge detection asserts RTC event
7	RESERVED	R	0h	Reserved

Table 15-89. IOCFG47 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-0	PORT_ID	R/W	0h	Selects usage for DIO47 Note: This field should not be written other than the times when PORT_ID value is specifically required to change. 0h = General Purpose IO 7h = AON 32 KHz clock (SCLK_LF) 9h = SPI0_RX : SPI0 RX Ah = SPI0_TX : SPI0 TX Bh = SPI0_CS : SPI0 CS Ch = SPI0_CLK : SPI0 CLK Dh = I2C0_MSSDA : I2C0 Data Eh = I2C0_MSSCL : I2C0 Clock Fh = UART0_RX : UART0 RX 10h = UART0_TX : UART0 TX 11h = UART0_CTS : UART0 CTS 12h = UART0_RTS : UART0 RTS 13h = UART1_RX : UART1 RX 14h = UART1_TX : UART1 TX 15h = UART1_CTS : UART1 CTS 16h = UART1_RTS : UART1 RTS 17h = PORT_EVENT0 : PORT EVENT 0 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 18h = PORT_EVENT1 : PORT EVENT 1 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 19h = PORT_EVENT2 : PORT EVENT 2 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ah = PORT_EVENT3 : PORT EVENT 3 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Bh = PORT_EVENT4 : PORT EVENT 4 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Ch = PORT_EVENT5 : PORT EVENT 5 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Dh = PORT_EVENT6 : PORT EVENT 6 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 1Eh = PORT_EVENT7 : PORT EVENT 7 Can be used as a general purpose IO event by selecting it through registers in the EVENT module, for example EVENT:GPT0ACAPTSEL.EV, EVENT:UDMACH14BSEL.EV, and so on 20h = CPU_SWV : CPU SWV 21h = SPI1_RX : SPI1 RX 22h = SPI1_TX : SPI1 TX 23h = SPI1_CS : SPI1 CS 24h = SPI1_CLK : SPI1 CLK 25h = I2S_AD0 : I2S Data 0

Table 15-89. IOCFG47 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				26h = I2S_AD1 : I2S Data 1 27h = I2S_WCLK : I2S WCLK 28h = I2S_BCLK : I2S BCLK 29h = I2S_MCLK : I2S MCLK 2Eh = RF Core Trace 2Fh = RF Core Data Out 0 30h = RF Core Data Out 1 31h = RF Core Data Out 2 32h = RF Core Data Out 3 33h = RF Core Data In 0 34h = RF Core Data In 1 35h = RF Core SMI Data Link Out 36h = RF Core SMI Data Link In 37h = RF Core SMI Command Link Out 38h = RF Core SMI Command Link In 39h = SPI2_RX : SPI2 RX 3Ah = SPI2_TX : SPI2 TX 3Bh = SPI2_CS : SPI2 CS 3Ch = SPI2_CLK : SPI2 CLK 3Dh = SPI3_RX : SPI3 RX 3Eh = SPI3_TX : SPI3 TX 3Fh = SPI3_CS : SPI3 CS 40h = SPI3_CLK : SPI3 CLK 41h = UART2_RX : UART2 RX 42h = UART2_TX : UART2 TX 43h = UART2_CTS : UART2 CTS 44h = UART2_RTS : UART2 RTS 45h = UART3_RX : UART3 RX 46h = UART3_TX : UART3 TX 47h = UART3_CTS : UART3 CTS 48h = UART3_RTS : UART3 RTS 49h = I2C1_MSSDA : I2C1 Data 4Ah = I2C1_MSSCL : I2C1 Clock

This page intentionally left blank.

Chapter 16

Micro Direct Memory Access (µDMA)

This chapter describes the direct memory access (DMA) controller, known as µDMA.

16.1 Introduction 1320
16.2 Block Diagram 1321
16.3 Functional Description 1321
16.4 Initialization and Configuration 1348
16.5 UDMA Registers 1350

16.1 Introduction

The CC13x4x10 and CC26x4x10 device platform includes a direct memory access (DMA) controller, known as µDMA. The µDMA controller provides a way to offload data transfer tasks from the Arm® Cortex®-M33 processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. The controller has dedicated channels for each supported on-chip module, and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller provides the following features:

- Arm® PrimeCell® 32-channel configurable µDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers in multiple transfer modes:
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation:
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules
  - Primary and secondary channel assignments
  - Flexible channel assignments
  - One channel each for receive and transmit paths for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, halfword, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

16.2 Block Diagram

Figure 16-1 shows the µDMA block diagram.

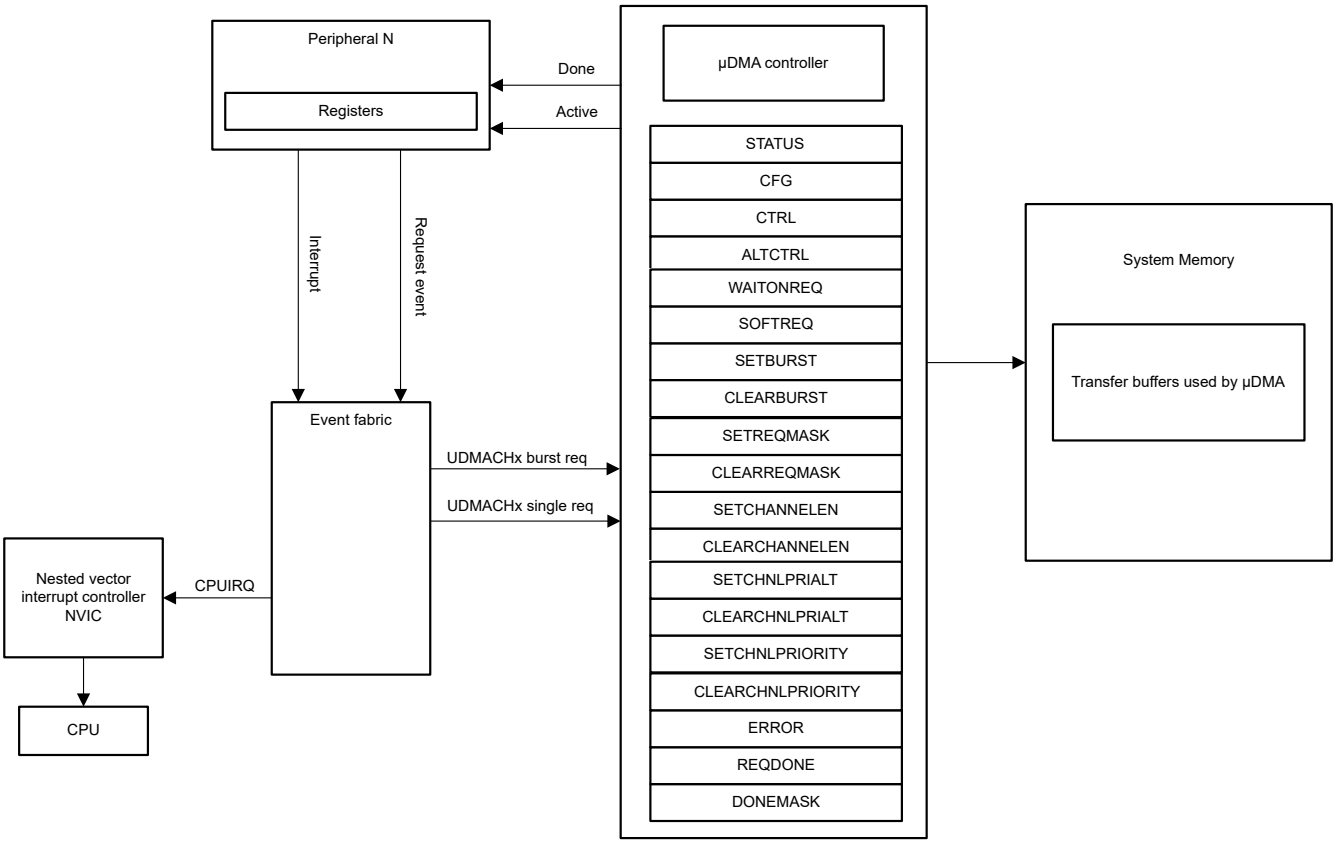


Figure 16-1. µDMA Block Diagram

16.3 Functional Description

The µDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the Arm® Cortex®-M33 processor core of the microcontroller. The controller supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers.

Each supported peripheral function has a dedicated channel on the µDMA controller that can be configured independently. The µDMA controller implements a configuration method using channel control structures maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated task lists in memory that allow the µDMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The µDMA controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the µDMA controller re-arbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral every time a µDMA service request is made.

16.3.1 Channel Assignments

Table 16-1 lists µDMA channel assignments to peripherals.

Table 16-1. Channel Assignments

Channel	Peripheral	waitonreq	stall	dma _done	dma _active	DMA_CHANNEL _WITH_2STAGE_SYNC	DMA _ACTIVE_FF	DMA_CHANNEL _ASYNC
31	UART3_TX	1		Yes	Yes	0		0
30	UART3_RX	1		Yes	Yes	0		0
29	UART2_TX	1		Yes	Yes	0		0
28	UART2_RX	1		Yes	Yes	0		0
27	Reserved	1				0		0
26	SPI3_MOSI	1		Yes	Yes	0		0
25	SPI3_MISO	1		Yes	Yes	0		0
24	Software Event 3	1		No	No	0		1
23	Software Event 2	1		No	No	0		1
22	Software Event 1	1				0		1
21	Software Event 0	1				0		1
20	SPI2_MOSI	1		Yes	Yes	0		0
19	SPI2_MISO	1		Yes	Yes	0		0
18	DMA Software 1	1		Yes		0		0
17	SPI1_MOSI	1		Yes	Yes	0		0
16	SPI1_MISO	1		Yes	Yes	0		0
15	AON_RTC	0				0		1
14	DMA_PROG	0				0		1
13	AON_PROG2	0				0		1
12	GPT1_B	1		Yes		0		0
11	GPT1_A	1		Yes		0		0
10	GPT0_B	1		Yes		0		0
9	GPT0_A	1		Yes		0		0
8	AUX_SW	0		Yes		0		1

Table 16-1. Channel Assignments (continued)

Channel	Peripheral	waitonreq	stall	dma _done	dma _active	DMA_CHANNEL _WITH_2STAGE_SYNC	DMA _ACTIVE_FF	DMA_CHANNEL _ASYNC
7	AUX_ADC	1		Yes	Yes	0	1	1
6	UART1_TX	1		Yes	Yes	0		0
5	UART1_RX	1		Yes	Yes	0		0
4	SPI0_MOSI	1		Yes	Yes	0		0
3	SPI0_MISO	1		Yes	Yes	0		0
2	UART0_TX	1		Yes	Yes	0		0
1	UART0_RX	1		Yes	Yes	0		0
0	DMA Software 0	1	Yes	Yes		0		0

16.3.2 Priority

The µDMA controller assigns priority to each channel based on the channel number and the priority-level bit for the channel. Channel 0 has the highest priority, and as the channel number increases, the priority of a channel decreases. Each channel has a priority-level bit to provide two levels of priority: default priority and high priority. If the priority-level bit is set, then that channel has a higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high-priority channels.

The priority bit for a channel can be set using the UDMA:SETCHNLPRIORITY register and cleared with the UDMA:CLEARCHNLPRIORITY register (see Section 16.5).

16.3.3 Arbitration Size

When a µDMA channel requests a transfer, the µDMA controller arbitrates among all the channels making a request, and services the µDMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before rearbitrating among the requesting channels. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the µDMA controller transfers the number of items specified by the arbitration size, the controller then checks among all the channels making a request, and services the channel with the highest priority.

If a lower-priority µDMA channel uses a large arbitration size, the latency for higher-priority channels is increased because the µDMA controller completes the lower-priority burst before checking for higher-priority requests. Therefore, lower-priority channels must not use a large arbitration size for best response on high-priority channels.

The arbitration size can also be thought of as burst size. Arbitration size is the maximum number of items that are transferred at any one time in a burst. Here, the term *arbitration* refers to the determination of the µDMA channel priority, not arbitration for the bus. When the µDMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the µDMA controller is delayed whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

16.3.4 Request Types

The µDMA controller responds to two types of requests from a peripheral: single request or burst request. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both types of requests are asserted and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. Table 16-2 lists how each peripheral supports the two request types.

Table 16-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
ADC	None (FIFO is not empty)	Sequencer IE bit (FIFO is half full)
General-purpose timer	Raw interrupt pulse	None
GPIO	Raw interrupt pulse	None
SPI MOSI	TX FIFO not full	TX FIFO level (configurable)
SPI MISO	RX FIFO not empty	RX FIFO level (configurable)
UART TX	TX FIFO not full	TX FIFO level (configurable)
UART RX	RX FIFO not empty	RX FIFO level (configurable)

16.3.4.1 Single Request

When a single request is detected (not a burst request), the µDMA controller transfers one item and then stops to wait for another request.

Note

Channels 8, 13, 14, and 15 do not respond to a single request because waitonreq is tied low.

16.3.4.2 Burst Request

When a burst request is detected, the µDMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size must be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART and SPI, which use a mix of single or burst requests, could generate a burst request based on the FIFO trigger level. In this case, the arbitration size must be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it starts and cannot be interrupted, even by a higher-priority channel. Burst transfers complete in a shorter time than the same number of nonburst transfers.

It may be desirable to use only burst transfers and not allow single transfers (for example, when the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time). The single request can be disabled in the UDMA:SETBURST register. By setting the bit for a channel in this register, the µDMA controller responds only to burst requests for that channel.

16.3.5 Channel Configuration

The µDMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each µDMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 16-3 describes the memory layout of the channel control table. Each channel may have one or two control structures in the control table—a primary control structure and an optional, alternate control structure. The table is organized with all of the primary entries in the first half of the table, and with all the alternate structures in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer completes. In this case, the alternate control structures are not used and only the first half of the table must be allocated in memory. The rest of the memory can be used for something else. If a more complex transfer mode is used, such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space must be allocated for the entire table.

Any unused memory in the control table may be used by the application, which includes the control structures for any channels that are unused by the application, as well as the unused control word for each channel.

Table 16-3. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 16-4 describes an individual control-structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four 4-byte long words: the source end pointer, the destination end pointer, the

control word, and an unused entry. The inclusive end pointers point to the ending address of the transfer. If the source or destination is nonincrementing (as for a peripheral register), then the pointer must point to the transfer address.

Table 16-4. Channel Control Structure

Offset	Description
0x000	Source end pointer
0x004	Destination end pointer
0x008	Control word
0x00C	Unused entry

The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control parameters for a channel can be set using the uDMAChannelControlSet() function from DriverLib (see SimpleLink™ CC13xx and CC26xx software development kit (SDK)). The µDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates stopped. Because the control word is modified by the µDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Before starting a transfer, a µDMA channel must be enabled by setting the appropriate bit in the UDMA:SETCHANNELEN register. A channel can be disabled by setting the channel bit in the UDMA:CLEARCHANNELEN register. At the end of a complete µDMA transfer, the controller automatically disables the channel.

16.3.6 Transfer Modes

The µDMA controller supports several transfer modes. Two of the modes support simple, one-time transfers. Several complex modes support a continuous flow of data.

16.3.6.1 Stop Mode

While stop mode is not actually a transfer mode, stop is a valid value for the *mode* field of the control word. When the mode field has the *stop* value, the µDMA controller does not perform any transfers and disables the channel if enabled. The µDMA controller updates the control word to set the mode to stop at the end of a transfer. This mode can be useful in scatter-gather operations.

16.3.6.2 Basic Mode

In basic mode, the µDMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a µDMA request signal whenever the peripheral is ready for a data transfer. Basic mode must not be used in any situation where the request is momentary, even though the entire transfer must be completed.

The µDMA controller sets the mode for that channel to stop when all of the items have been transferred using basic mode.

16.3.6.3 Auto Mode

Auto mode is similar to basic mode, except that when a transfer request is received, the transfer completes, even if the µDMA request is removed. This mode is suitable for software-triggered transfers. Generally, auto mode is not used with a peripheral.

The µDMA controller sets the mode for that channel to stop when all the items have been transferred using auto mode.

16.3.6.4 Ping-Pong Mode

Ping-pong mode is used to support a continuous data flow to or from a peripheral. Both the primary and alternate data structures must be implemented to use ping-pong mode. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure completes, the µDMA controller reads the alternate control structure for that channel to continue the transfer. Each time this occurs, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch between buffers as the data flows to or from the peripheral.

Figure 16-2 shows an example operation in ping-pong mode.

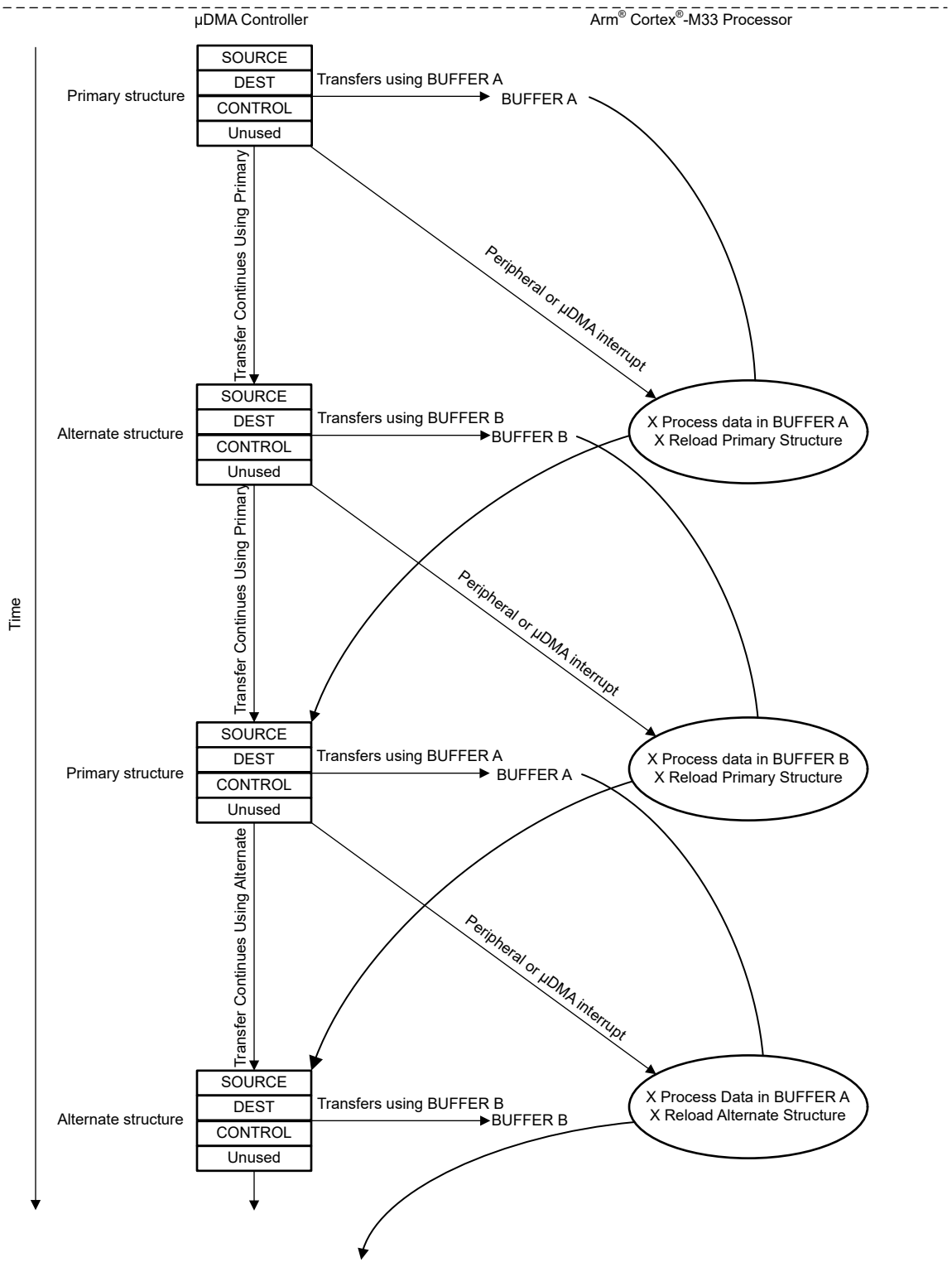


Figure 16-2. Example of Ping-Pong µDMA Transaction

16.3.6.5 Memory Scatter-Gather Mode

Memory scatter-gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather µDMA operation could be used to selectively read the payload of several stored packets of a communication protocol, and store them together in sequence in a memory buffer.

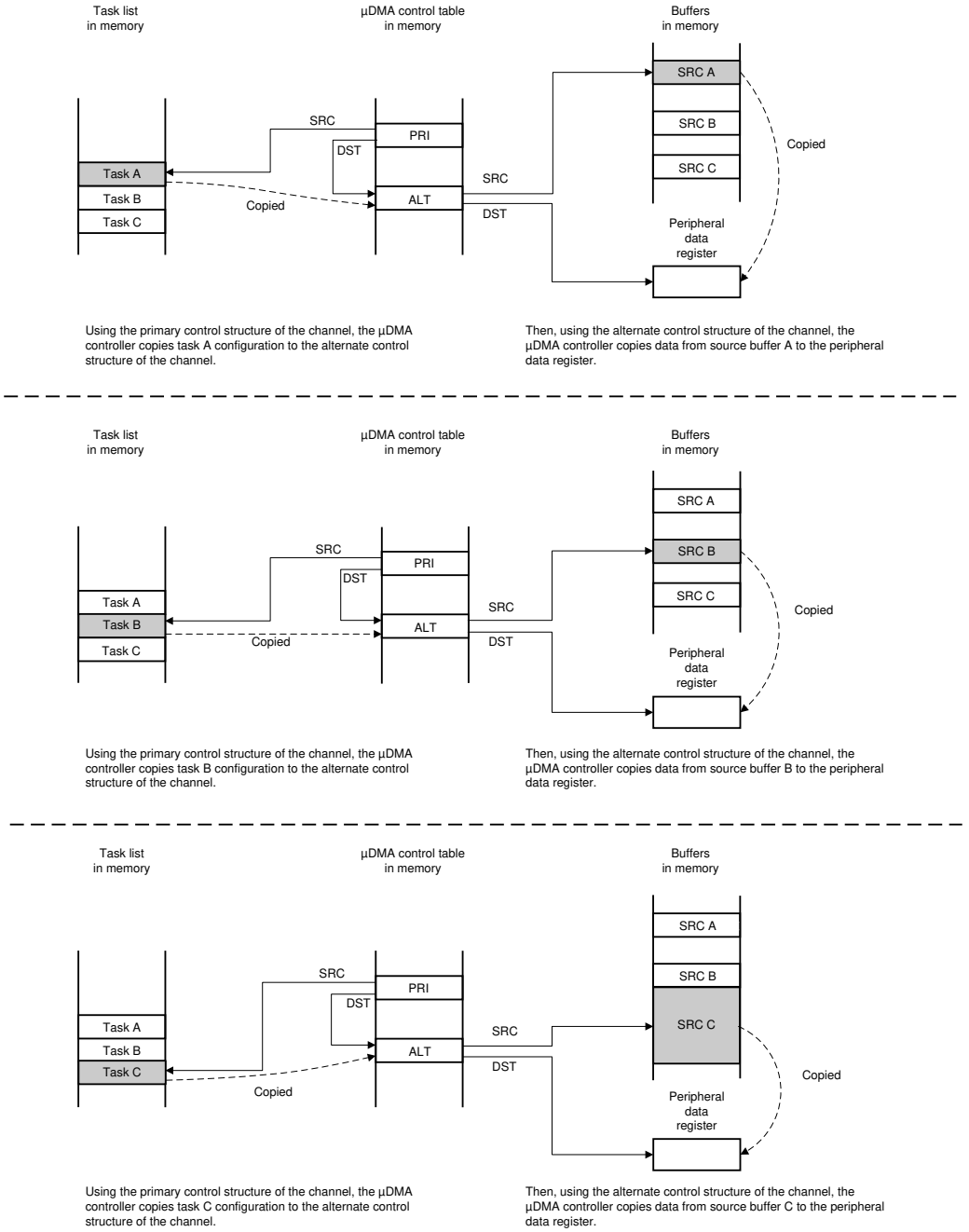
In memory scatter-gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to memory scatter-gather mode. Each entry in the table is, in turn, copied to the alternate structure where it is then executed. The µDMA controller alternates between using the primary control structure to copy the next transfer instruction from the list, and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use auto transfer mode. When the last transfer is performed using auto mode, the µDMA controller stops. A completion interrupt is generated only after the last transfer.

It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a µDMA request.

By programming the µDMA controller using this method, a set of arbitrary transfers can be performed based on a single µDMA request.

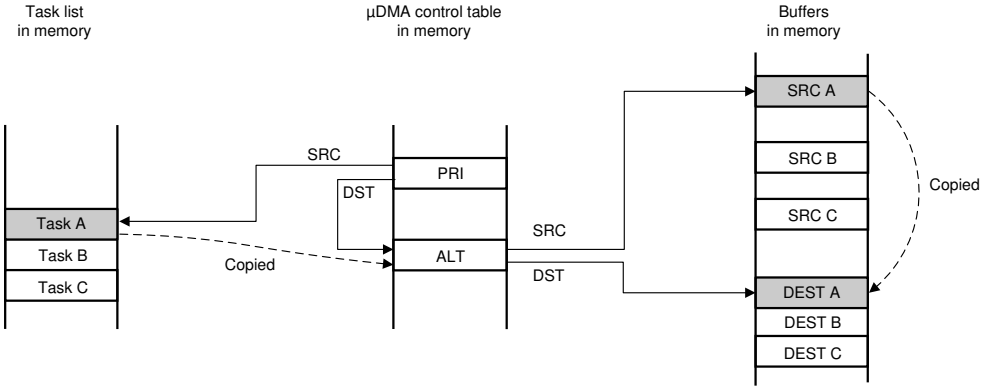
Figure 16-3 shows an example of operation in memory scatter-gather mode. This example shows a gather operation, where data in three separate buffers in memory is copied together into one buffer. Figure 16-3 shows how the application sets up a µDMA task list in memory, that is then used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel used for the operation is configured to copy from the task list to the alternate control structure.

Figure 16-4 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with Task A. The µDMA controller then performs the copy operation specified by Task A, copying the data from the source buffer A to the destination buffer. Next, the µDMA controller again uses the primary control structure to load Task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for Task C.



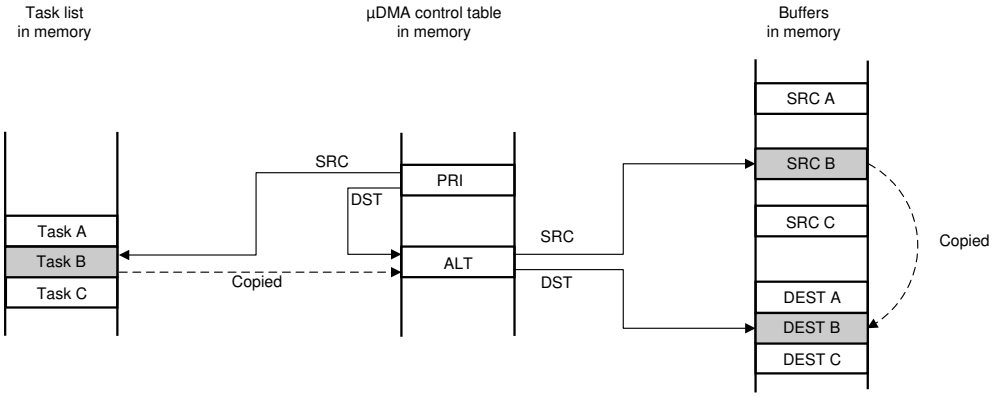
A. The application has a need to copy data items from three separate locations in memory into one combined buffer.
B. The application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
C. The application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.

Figure 16-3. Memory Scatter-Gather, Setup, and Configuration



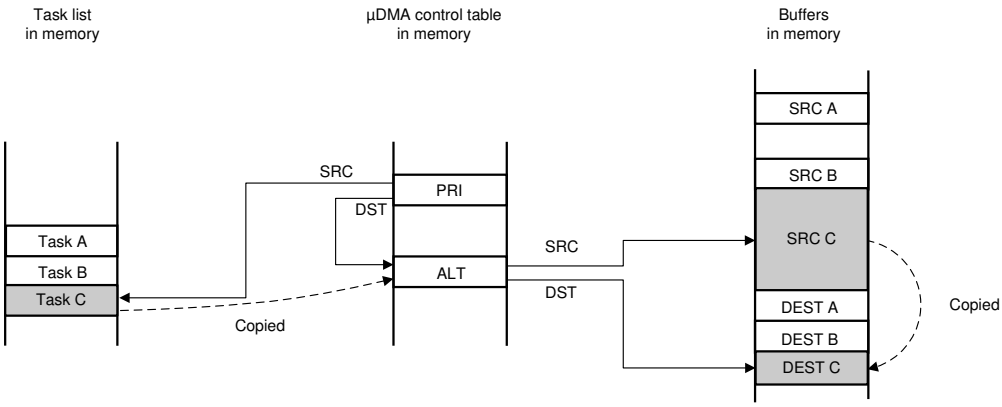
Using the primary control structure of the channel, the µDMA controller copies task A configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer A to the destination buffer.



Using the primary control structure of the channel, the µDMA controller copies task B configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer B to the destination buffer.



Using the primary control structure of the channel, the µDMA controller copies task C configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer C to destination buffer.

Figure 16-4. Memory Scatter-Gather, µDMA Copy Sequence

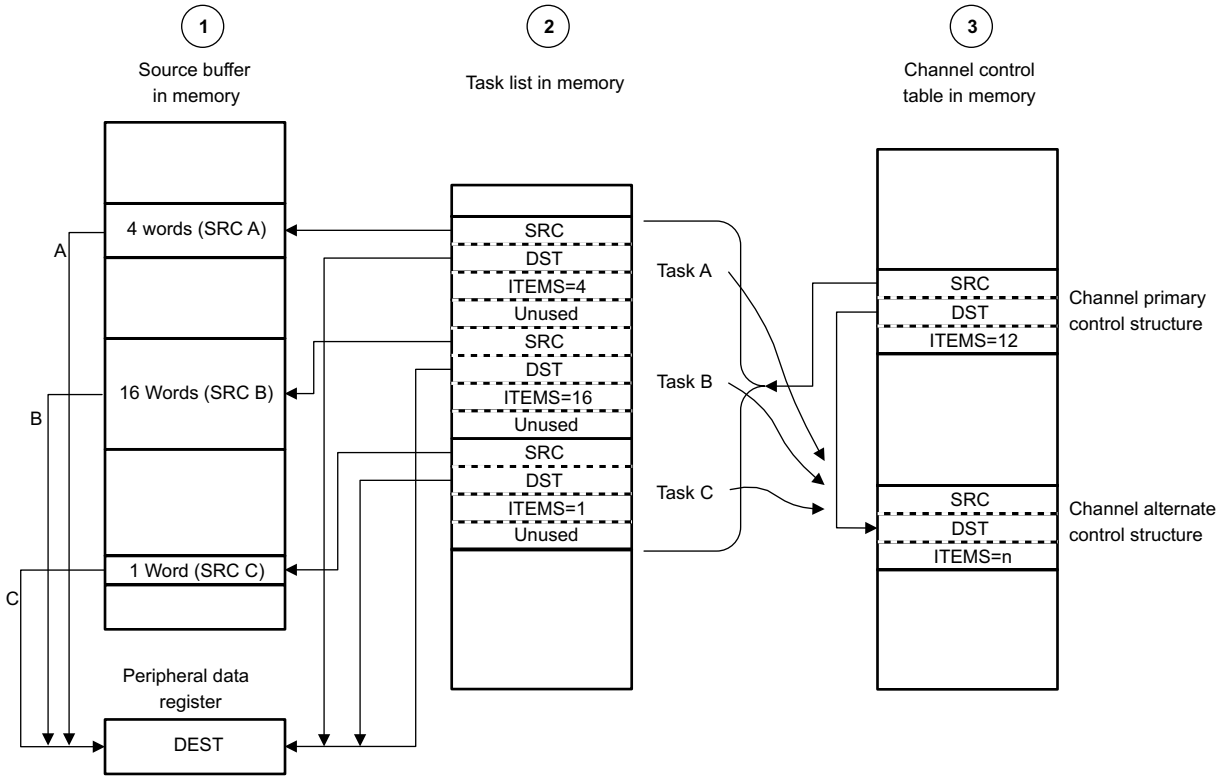
16.3.6.6 Peripheral Scatter-Gather Mode

Peripheral scatter-gather mode is similar to memory scatter-gather mode, except that the transfers are controlled by a peripheral making a µDMA request. When the µDMA controller detects a request from the peripheral, the µDMA controller uses the primary control structure to copy one entry from the list to the alternate control structure, and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a µDMA request. The µDMA controller continues to perform transfers from the list only when the peripheral makes a request, until the last transfer completes. A completion interrupt is generated only after the last transfer.

By using this method, the µDMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Figure 16-5 shows an example of operation in peripheral scatter-gather mode. This example shows a gather operation where data from three separate buffers in memory is copied to a single peripheral data register. Figure 16-5 shows how the application sets up a µDMA task list in memory, that is then used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel used for the operation is configured to copy from the task list to the alternate control structure.

Figure 16-6 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with Task A. The µDMA controller then performs the copy operation specified by Task A, copying the data from the source buffer A to the peripheral data register. Next, the µDMA controller again uses the primary control structure to load Task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for Task C.

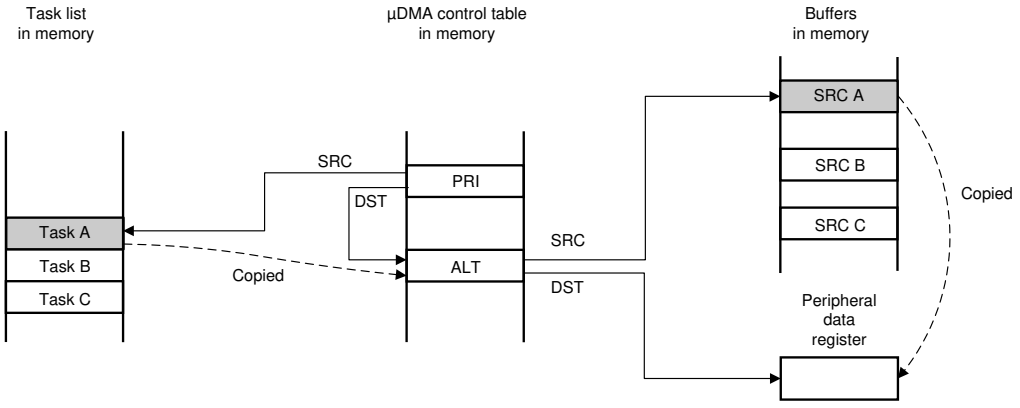


A. The application has a need to copy data items from three separate locations in memory into a peripheral data register.

B. The application sets up the µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."

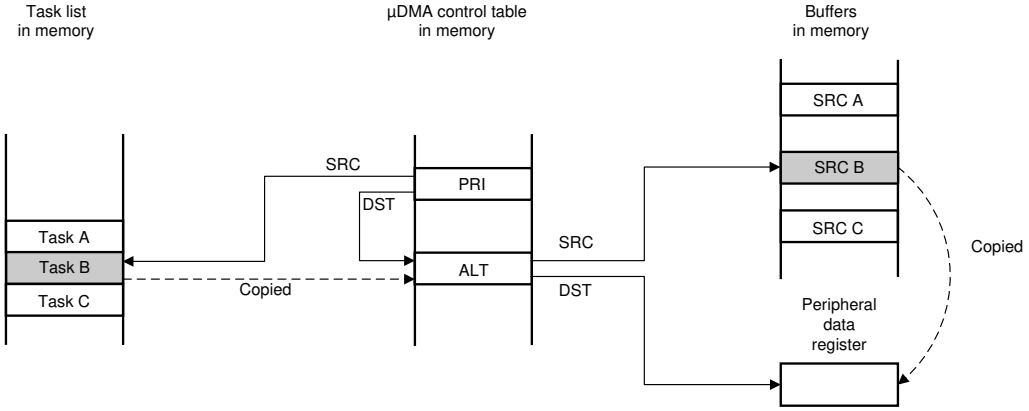
C. The application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.

Figure 16-5. Peripheral Scatter-Gather, Setup, and Configuration



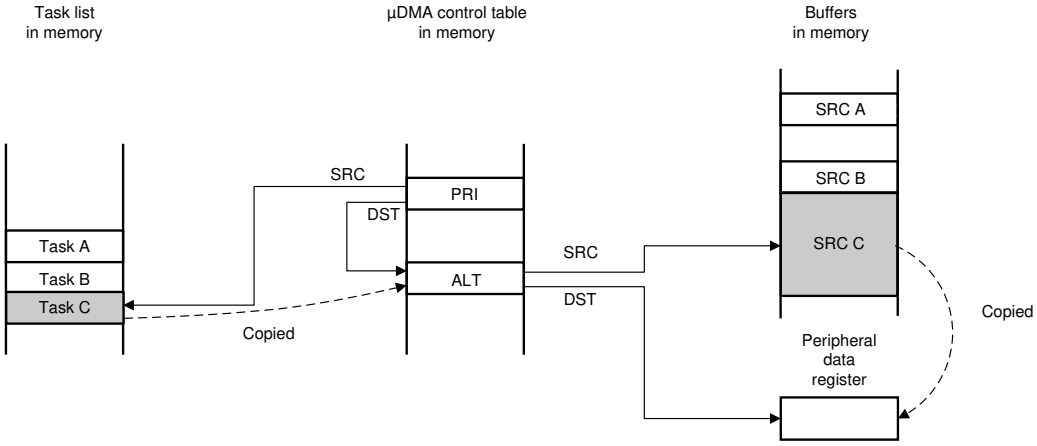
Using the primary control structure of the channel, the µDMA controller copies task A configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer A to the peripheral data register.



Using the primary control structure of the channel, the µDMA controller copies task B configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer B to the peripheral data register.



Using the primary control structure of the channel, the µDMA controller copies task C configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer C to the peripheral data register.

Figure 16-6. Peripheral Scatter-Gather, µDMA Copy Sequence

16.3.7 Transfer Size and Increments

The µDMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be automatically incremented by bytes, half-words, words, or set to no increment. The source and destination address increment values can be set independently; it is not necessary for the address increment to match the data size, as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size by using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 16-5 provides the configuration to read from a peripheral that supplies 8-bit data.

Table 16-5. µDMA Read Example: 8-Bit Peripheral

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

16.3.8 Peripheral Interface

Each peripheral that supports µDMA has a single request or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 16-2). The request signal can be disabled or enabled using the UDMA:SETREQMASK and UDMA:CLEARREQMASK registers, respectively. The µDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the µDMA channel is configured correctly and enabled, the peripheral asserts the request signal, and the µDMA controller begins the transfer.

Note

The peripheral must disable all interrupts to the event fabric when using µDMA to transfer data to and from a peripheral.

When a µDMA transfer is complete, the µDMA controller generates an interrupt; for more information, see Section 16.3.10.

For more information on how a specific peripheral interacts with the µDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

16.3.9 Software Request

Channels may be set up to perform software transfers through the UDMA:SOFTREQ register. If the channel used for software is also tied to a specific peripheral, the dma_done/interrupt signal is provided directly to the Arm® Cortex®-M33 CPU instead of sending it to the peripheral. The interrupt used is a combined interrupt, number 46 – software µDMA interrupt, for all software transfers.

If software uses a µDMA channel of the peripheral to initiate a request, then the completion interrupt occurs on the interrupt vector for the peripheral instead of occurring on the software interrupt vector.

Note

DMA software requests are specified on channels 0 and 18. For these channels, dma_done is available as events DMA_CH0_DONE and DMA_CH18_DONE in the EV field of the EVENT:UDMACH14BSEL or EVENT:CPUIRQSEL30 registers.

16.3.10 Interrupts and Errors

The µDMA controller generates a completion interrupt on the interrupt vector of the peripheral when a µDMA transfer completes. Therefore, if µDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the µDMA transfer completion interrupt. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 16-6).

When µDMA is enabled for a peripheral, the µDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (INTC). The interrupts are still reported in the interrupt registers of the peripheral. Thus, when a large amount of data is transferred using µDMA, instead of receiving multiple interrupts from the peripheral as data flows, the INTC receives only one interrupt when the transfer completes. Unmasked peripheral error interrupts continue to be sent to the INTC.

When a µDMA channel generates a completion interrupt, the CHNLS bit corresponding to the peripheral channel is set in the DMA Channel Request Done register, UDMA:REQDONE. This register can be used by the interrupt handler code of the peripheral to determine if the interrupt was caused by the µDMA channel or an error event reported by the interrupt registers of the peripheral. The completion interrupt request from the µDMA controller is automatically cleared when the interrupt handler is activated.

If the µDMA controller encounters a bus or memory protection error as it tries to perform a data transfer, the controller disables the µDMA channel that caused the error and generates an interrupt on the µDMA error interrupt vector. The processor can read the DMA Clear Bus Error register, UDMA:ERROR, to determine if an error is pending. The STATUS bit is set if an error occurred. The error can be cleared by setting the STATUS bit to 1.

Note

The error interrupt or event goes to the event fabric as DMA_ERR, and is connected as an interrupt to the Arm® Cortex®-M33 processor through the EVENT:CPUIRQSEL25 register.

Table 16-6 lists the dedicated interrupt assignments for the µDMA controller.

Table 16-6. µDMA Interrupt Assignments

Interrupt	Assignment
40	µDMA software channel transfer
41	µDMA error

16.3.2 Priority

The µDMA controller assigns priority to each channel based on the channel number and the priority-level bit for the channel. Channel 0 has the highest priority, and as the channel number increases, the priority of a channel decreases. Each channel has a priority-level bit to provide two levels of priority: default priority and high priority. If the priority-level bit is set, then that channel has a higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high-priority channels.

The priority bit for a channel can be set using the UDMA:SETCHNLPRIORITY register and cleared with the UDMA:CLEARCHNLPRIORITY register (see Section 16.5).

16.3.3 Arbitration Size

When a µDMA channel requests a transfer, the µDMA controller arbitrates among all the channels making a request, and services the µDMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before rearbitrating among the requesting channels. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the µDMA controller transfers the number of items specified by the arbitration size, the controller then checks among all the channels making a request, and services the channel with the highest priority.

If a lower-priority µDMA channel uses a large arbitration size, the latency for higher-priority channels is increased because the µDMA controller completes the lower-priority burst before checking for higher-priority requests. Therefore, lower-priority channels must not use a large arbitration size for best response on high-priority channels.

The arbitration size can also be thought of as burst size. Arbitration size is the maximum number of items that are transferred at any one time in a burst. Here, the term *arbitration* refers to the determination of the µDMA channel priority, not arbitration for the bus. When the µDMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the µDMA controller is delayed whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

16.3.4 Request Types

The µDMA controller responds to two types of requests from a peripheral: single request or burst request. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both types of requests are asserted and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. Table 16-2 lists how each peripheral supports the two request types.

Table 16-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
ADC	None (FIFO is not empty)	Sequencer IE bit (FIFO is half full)
General-purpose timer	Raw interrupt pulse	None
GPIO	Raw interrupt pulse	None
SPI MOSI	TX FIFO not full	TX FIFO level (configurable)
SPI MISO	RX FIFO not empty	RX FIFO level (configurable)
UART TX	TX FIFO not full	TX FIFO level (configurable)
UART RX	RX FIFO not empty	RX FIFO level (configurable)

16.3.4.1 Single Request

When a single request is detected (not a burst request), the µDMA controller transfers one item and then stops to wait for another request.

Note

Channels 8, 13, 14, and 15 do not respond to a single request because waitonreq is tied low.

16.3.4.2 Burst Request

When a burst request is detected, the µDMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size must be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART and SPI, which use a mix of single or burst requests, could generate a burst request based on the FIFO trigger level. In this case, the arbitration size must be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it starts and cannot be interrupted, even by a higher-priority channel. Burst transfers complete in a shorter time than the same number of nonburst transfers.

It may be desirable to use only burst transfers and not allow single transfers (for example, when the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time). The single request can be disabled in the UDMA:SETBURST register. By setting the bit for a channel in this register, the µDMA controller responds only to burst requests for that channel.

16.3.5 Channel Configuration

The µDMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each µDMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 16-3 describes the memory layout of the channel control table. Each channel may have one or two control structures in the control table—a primary control structure and an optional, alternate control structure. The table is organized with all of the primary entries in the first half of the table, and with all the alternate structures in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer completes. In this case, the alternate control structures are not used and only the first half of the table must be allocated in memory. The rest of the memory can be used for something else. If a more complex transfer mode is used, such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space must be allocated for the entire table.

Any unused memory in the control table may be used by the application, which includes the control structures for any channels that are unused by the application, as well as the unused control word for each channel.

Table 16-3. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 16-4 describes an individual control-structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four 4-byte long words: the source end pointer, the destination end pointer, the

control word, and an unused entry. The inclusive end pointers point to the ending address of the transfer. If the source or destination is nonincrementing (as for a peripheral register), then the pointer must point to the transfer address.

Table 16-4. Channel Control Structure

Offset	Description
0x000	Source end pointer
0x004	Destination end pointer
0x008	Control word
0x00C	Unused entry

The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control parameters for a channel can be set using the uDMAChannelControlSet() function from DriverLib (see SimpleLink™ CC13xx and CC26xx software development kit (SDK)). The µDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates stopped. Because the control word is modified by the µDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Before starting a transfer, a µDMA channel must be enabled by setting the appropriate bit in the UDMA:SETCHANNELEN register. A channel can be disabled by setting the channel bit in the UDMA:CLEARCHANNELEN register. At the end of a complete µDMA transfer, the controller automatically disables the channel.

16.3.6 Transfer Modes

The µDMA controller supports several transfer modes. Two of the modes support simple, one-time transfers. Several complex modes support a continuous flow of data.

16.3.6.1 Stop Mode

While stop mode is not actually a transfer mode, stop is a valid value for the *mode* field of the control word. When the mode field has the *stop* value, the µDMA controller does not perform any transfers and disables the channel if enabled. The µDMA controller updates the control word to set the mode to stop at the end of a transfer. This mode can be useful in scatter-gather operations.

16.3.6.2 Basic Mode

In basic mode, the µDMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a µDMA request signal whenever the peripheral is ready for a data transfer. Basic mode must not be used in any situation where the request is momentary, even though the entire transfer must be completed.

The µDMA controller sets the mode for that channel to stop when all of the items have been transferred using basic mode.

16.3.6.3 Auto Mode

Auto mode is similar to basic mode, except that when a transfer request is received, the transfer completes, even if the µDMA request is removed. This mode is suitable for software-triggered transfers. Generally, auto mode is not used with a peripheral.

The µDMA controller sets the mode for that channel to stop when all the items have been transferred using auto mode.

16.3.6.4 Ping-Pong Mode

Ping-pong mode is used to support a continuous data flow to or from a peripheral. Both the primary and alternate data structures must be implemented to use ping-pong mode. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure completes, the µDMA controller reads the alternate control structure for that channel to continue the transfer. Each time this occurs, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch between buffers as the data flows to or from the peripheral.

Figure 16-2 shows an example operation in ping-pong mode.

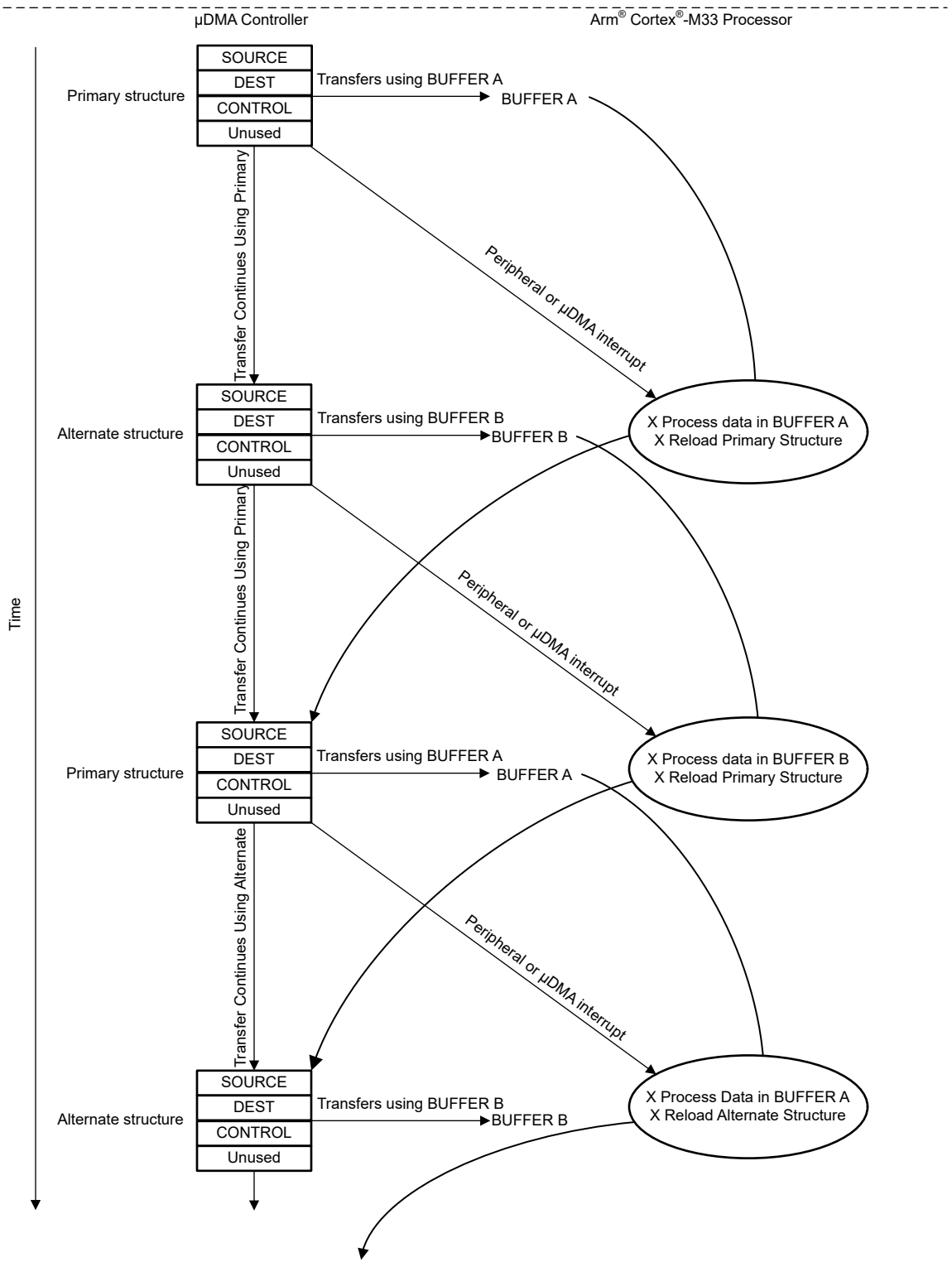


Figure 16-2. Example of Ping-Pong µDMA Transaction

16.3.6.5 Memory Scatter-Gather Mode

Memory scatter-gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather µDMA operation could be used to selectively read the payload of several stored packets of a communication protocol, and store them together in sequence in a memory buffer.

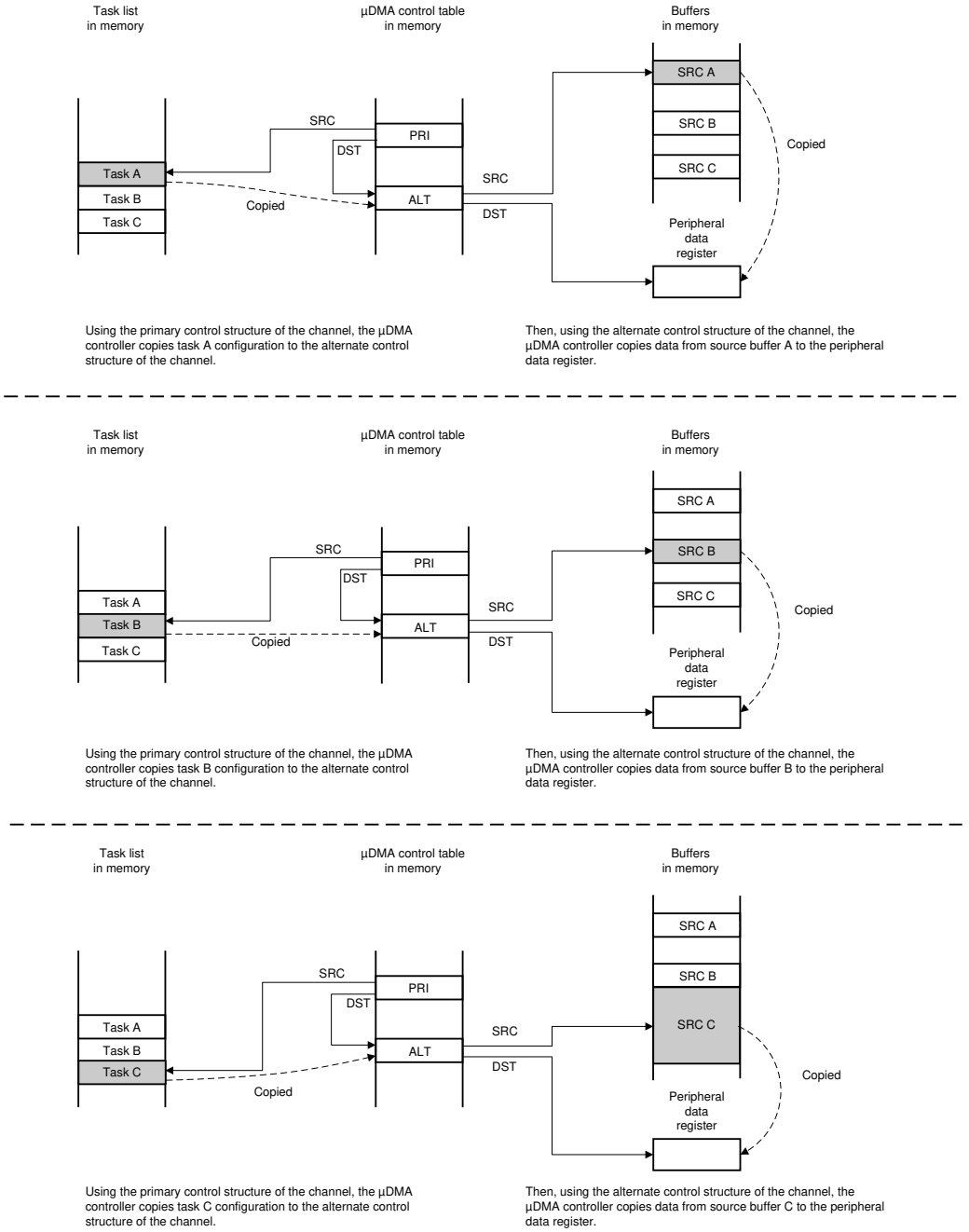
In memory scatter-gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to memory scatter-gather mode. Each entry in the table is, in turn, copied to the alternate structure where it is then executed. The µDMA controller alternates between using the primary control structure to copy the next transfer instruction from the list, and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use auto transfer mode. When the last transfer is performed using auto mode, the µDMA controller stops. A completion interrupt is generated only after the last transfer.

It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a µDMA request.

By programming the µDMA controller using this method, a set of arbitrary transfers can be performed based on a single µDMA request.

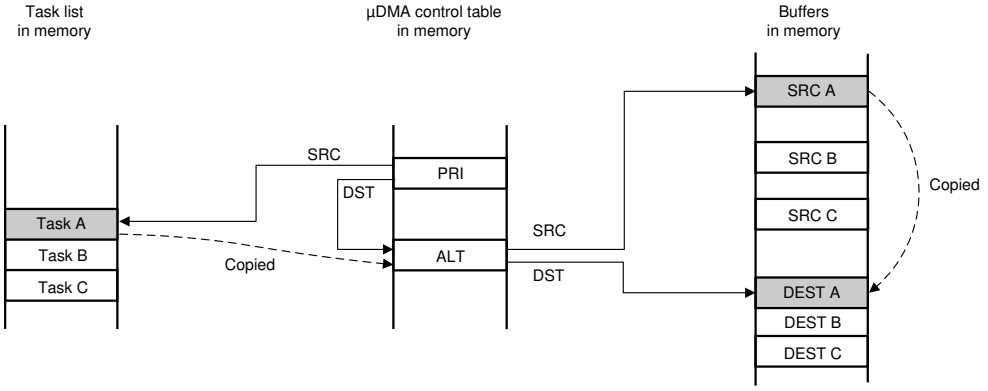
Figure 16-3 shows an example of operation in memory scatter-gather mode. This example shows a gather operation, where data in three separate buffers in memory is copied together into one buffer. Figure 16-3 shows how the application sets up a µDMA task list in memory, that is then used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel used for the operation is configured to copy from the task list to the alternate control structure.

Figure 16-4 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with Task A. The µDMA controller then performs the copy operation specified by Task A, copying the data from the source buffer A to the destination buffer. Next, the µDMA controller again uses the primary control structure to load Task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for Task C.



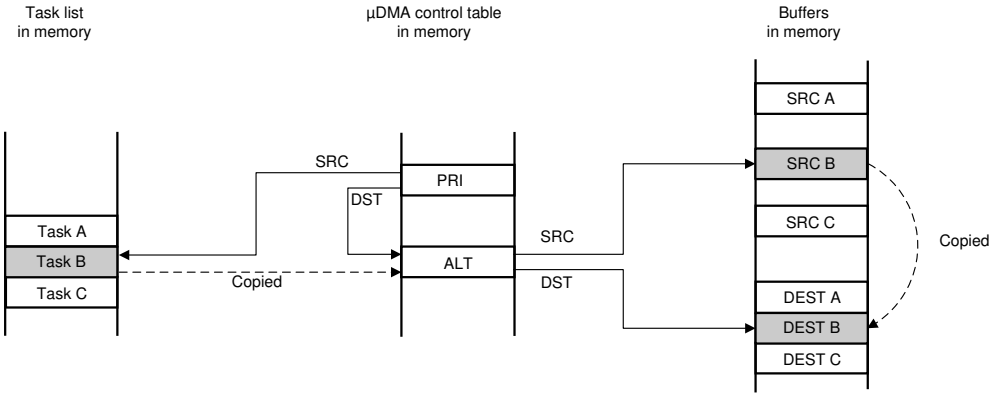
A. The application has a need to copy data items from three separate locations in memory into one combined buffer.
B. The application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
C. The application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.

Figure 16-3. Memory Scatter-Gather, Setup, and Configuration



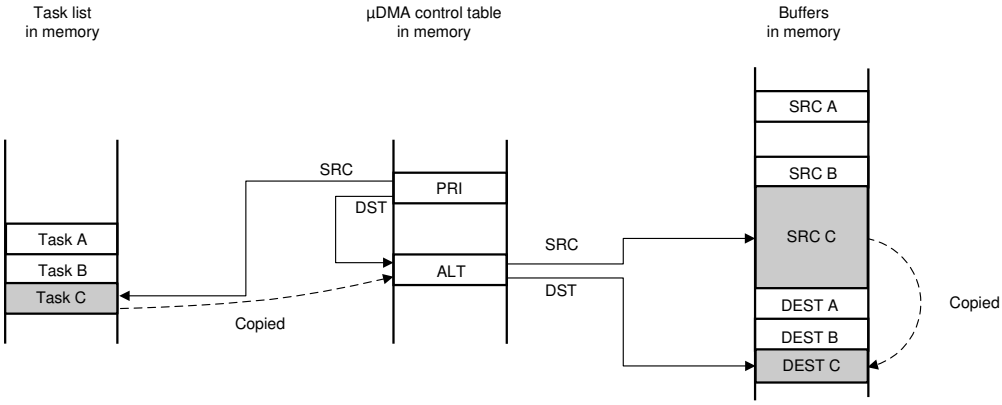
Using the primary control structure of the channel, the µDMA controller copies task A configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer A to the destination buffer.



Using the primary control structure of the channel, the µDMA controller copies task B configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer B to the destination buffer.



Using the primary control structure of the channel, the µDMA controller copies task C configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer C to destination buffer.

Figure 16-4. Memory Scatter-Gather, µDMA Copy Sequence

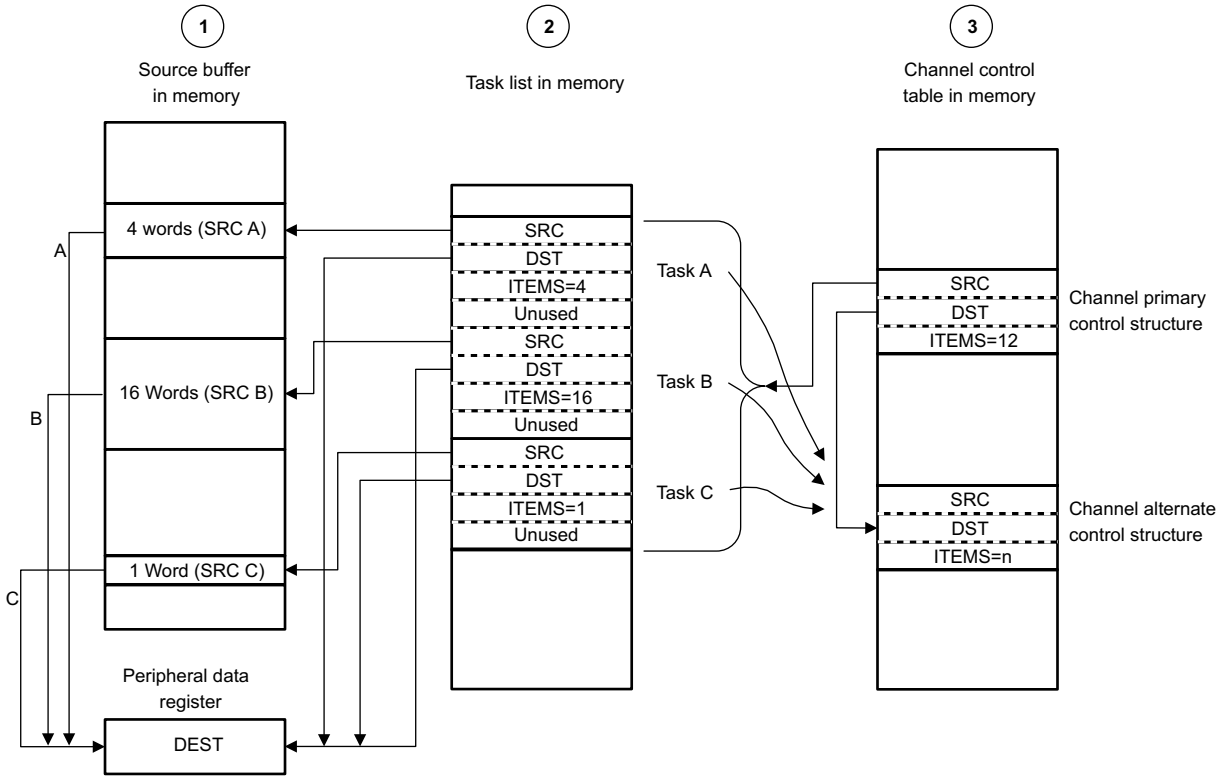
16.3.6.6 Peripheral Scatter-Gather Mode

Peripheral scatter-gather mode is similar to memory scatter-gather mode, except that the transfers are controlled by a peripheral making a µDMA request. When the µDMA controller detects a request from the peripheral, the µDMA controller uses the primary control structure to copy one entry from the list to the alternate control structure, and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a µDMA request. The µDMA controller continues to perform transfers from the list only when the peripheral makes a request, until the last transfer completes. A completion interrupt is generated only after the last transfer.

By using this method, the µDMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Figure 16-5 shows an example of operation in peripheral scatter-gather mode. This example shows a gather operation where data from three separate buffers in memory is copied to a single peripheral data register. Figure 16-5 shows how the application sets up a µDMA task list in memory, that is then used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel used for the operation is configured to copy from the task list to the alternate control structure.

Figure 16-6 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with Task A. The µDMA controller then performs the copy operation specified by Task A, copying the data from the source buffer A to the peripheral data register. Next, the µDMA controller again uses the primary control structure to load Task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for Task C.

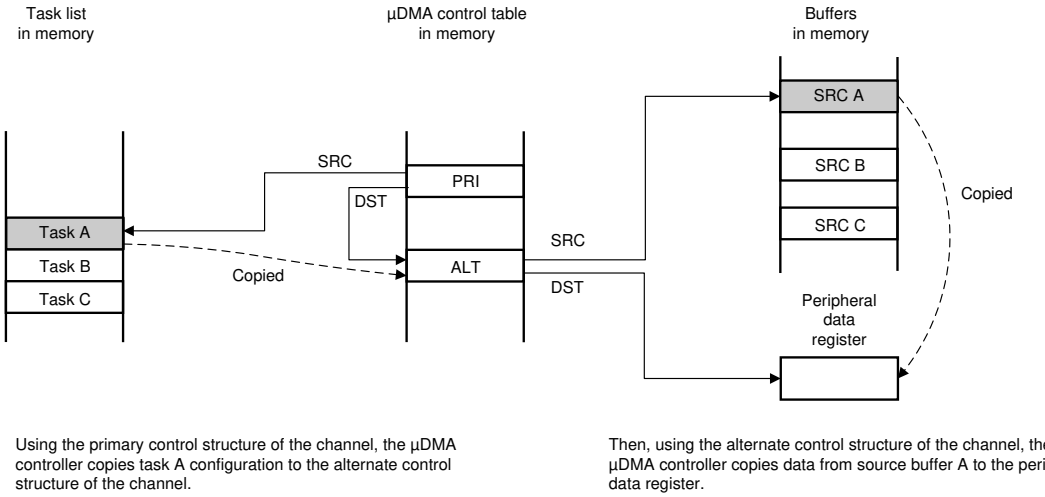


A. The application has a need to copy data items from three separate locations in memory into a peripheral data register.

B. The application sets up the µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."

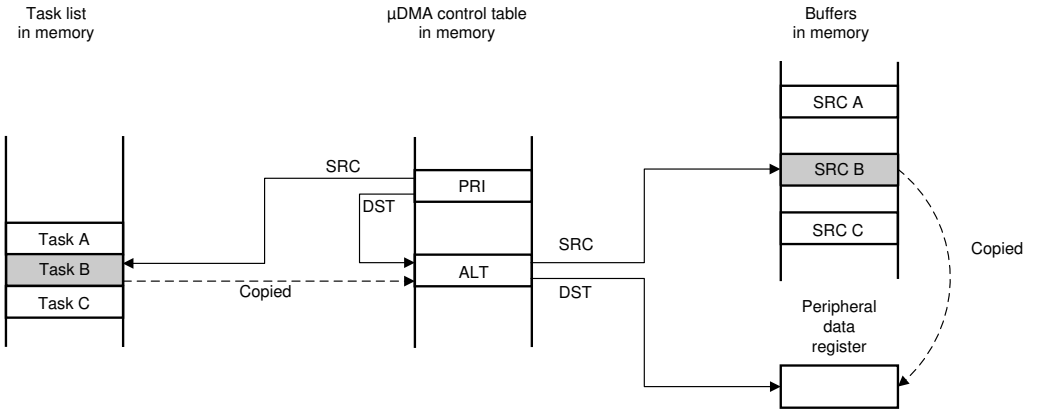
C. The application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.

Figure 16-5. Peripheral Scatter-Gather, Setup, and Configuration



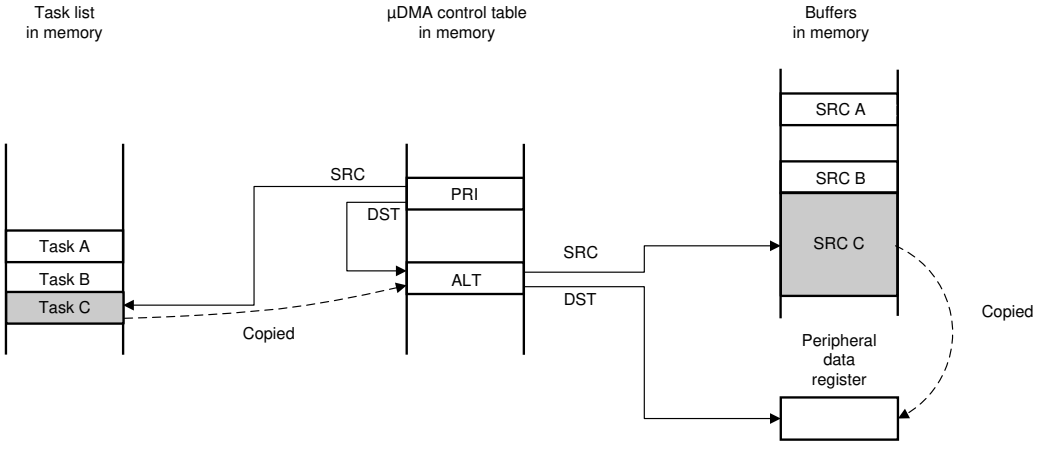
Using the primary control structure of the channel, the µDMA controller copies task A configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer A to the peripheral data register.



Using the primary control structure of the channel, the µDMA controller copies task B configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer B to the peripheral data register.



Using the primary control structure of the channel, the µDMA controller copies task C configuration to the alternate control structure of the channel.

Then, using the alternate control structure of the channel, the µDMA controller copies data from source buffer C to the peripheral data register.

Figure 16-6. Peripheral Scatter-Gather, µDMA Copy Sequence

16.3.7 Transfer Size and Increments

The µDMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be automatically incremented by bytes, half-words, words, or set to no increment. The source and destination address increment values can be set independently; it is not necessary for the address increment to match the data size, as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size by using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 16-5 provides the configuration to read from a peripheral that supplies 8-bit data.

Table 16-5. µDMA Read Example: 8-Bit Peripheral

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

16.3.8 Peripheral Interface

Each peripheral that supports µDMA has a single request or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 16-2). The request signal can be disabled or enabled using the UDMA:SETREQMASK and UDMA:CLEARREQMASK registers, respectively. The µDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the µDMA channel is configured correctly and enabled, the peripheral asserts the request signal, and the µDMA controller begins the transfer.

Note

The peripheral must disable all interrupts to the event fabric when using µDMA to transfer data to and from a peripheral.

When a µDMA transfer is complete, the µDMA controller generates an interrupt; for more information, see Section 16.3.10.

For more information on how a specific peripheral interacts with the µDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

16.3.9 Software Request

Channels may be set up to perform software transfers through the UDMA:SOFTREQ register. If the channel used for software is also tied to a specific peripheral, the dma_done/interrupt signal is provided directly to the Arm® Cortex®-M33 CPU instead of sending it to the peripheral. The interrupt used is a combined interrupt, number 46 – software µDMA interrupt, for all software transfers.

If software uses a µDMA channel of the peripheral to initiate a request, then the completion interrupt occurs on the interrupt vector for the peripheral instead of occurring on the software interrupt vector.

Note

DMA software requests are specified on channels 0 and 18. For these channels, dma_done is available as events DMA_CH0_DONE and DMA_CH18_DONE in the EV field of the EVENT:UDMACH14BSEL or EVENT:CPUIRQSEL30 registers.

16.3.10 Interrupts and Errors

The µDMA controller generates a completion interrupt on the interrupt vector of the peripheral when a µDMA transfer completes. Therefore, if µDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the µDMA transfer completion interrupt. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 16-6).

When µDMA is enabled for a peripheral, the µDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (INTC). The interrupts are still reported in the interrupt registers of the peripheral. Thus, when a large amount of data is transferred using µDMA, instead of receiving multiple interrupts from the peripheral as data flows, the INTC receives only one interrupt when the transfer completes. Unmasked peripheral error interrupts continue to be sent to the INTC.

When a µDMA channel generates a completion interrupt, the CHNLS bit corresponding to the peripheral channel is set in the DMA Channel Request Done register, UDMA:REQDONE. This register can be used by the interrupt handler code of the peripheral to determine if the interrupt was caused by the µDMA channel or an error event reported by the interrupt registers of the peripheral. The completion interrupt request from the µDMA controller is automatically cleared when the interrupt handler is activated.

If the µDMA controller encounters a bus or memory protection error as it tries to perform a data transfer, the controller disables the µDMA channel that caused the error and generates an interrupt on the µDMA error interrupt vector. The processor can read the DMA Clear Bus Error register, UDMA:ERROR, to determine if an error is pending. The STATUS bit is set if an error occurred. The error can be cleared by setting the STATUS bit to 1.

Note

The error interrupt or event goes to the event fabric as DMA_ERR, and is connected as an interrupt to the Arm® Cortex®-M33 processor through the EVENT:CPUIRQSEL25 register.

Table 16-6 lists the dedicated interrupt assignments for the µDMA controller.

Table 16-6. µDMA Interrupt Assignments

Interrupt	Assignment
40	µDMA software channel transfer
41	µDMA error

16.4 Initialization and Configuration

16.4.1 Module Initialization

The DMA controller resides in the peripheral domain, which must be powered up to enable the µDMA controller. The following steps are necessary:

1. Enable the peripheral power domain (PRCM_DOMAIN_PERIPH) by setting the PRCM:PDCTL0PERIPH.ON register bit or by using the driver library function:

```
PRCMPowerDomainOn(uint32_t ui32Domains)
```

2. Enable the µDMA controller by setting the PRCM:SECDMACLKGR.DMA_CLK_EN register bit and the PRCM:SECDMACLKGS.DMA_CLK_EN register bit or by using the driver library functions:

```
PRCMPeripheralRunEnable(uint32_t ui32Peripheral)
```

   and

```
PRCMPeripheralSleepEnable(uint32_t ui32Peripheral)
```

3. Load the setting to clock controller by setting the PRCM:CLKLOADCTL.LOAD register bit or by using the function:

```
PRCMLoadSet()
```

4. Enable the µDMA controller by setting the DMA Configuration register, UDMA:CFG, MASTERENABLE bit.
5. Program the location of the channel control table by writing the base address of the table to the DMA Channel Control Base Pointer register, UDMA:CTRL. The base address must be aligned on a 1024-byte boundary.

16.4.2 Configuring a Memory-to-Memory Transfer

The µDMA channels 0, 18, 19, and 20 are dedicated for software-initiated transfers. This specific example uses channel 0. No attributes must be set for a software-based transfer. The attributes are cleared by default, but are explicitly cleared as shown in the following sections.

16.4.2.1 Configure the Channel Attributes

Configure the channel attributes as follows, or use the following driver library function:

```
uDMAChannelAttributeDisable(uint32_t ui32Base, uint32_t ui32ChannelNum, uint32_t ui32Attr)
```

1. Program bit 0 of the DMA Set Channel Priority register, UDMA:SETCHNLPRIORITY, or the DMA Clear Channel Priority register, UDMA:CLEARCHNLPRIORITY, to set the channel to high priority or default priority.
2. Set bit 0 of the DMA Clear Channel Primary Alternate register, UDMA:CLEARCHNLPRIALT, to select the primary channel control structure for this transfer.
3. Set bit 0 of the DMA Channel Clear Useburst register, UDMA:CLEARBURST, to allow the µDMA controller to respond to single requests and burst requests.
4. Set bit 0 of the DMA Clear Channel Request Mask register, UDMA:CLEARREQMASK, to allow the µDMA controller to recognize requests for this channel.

16.4.2.2 Configure the Channel Control Structure

This example transfers 256 words from one memory buffer to another. Channel 0 is used for a software transfer, and the control structure for channel 0 must be configured to transfer 8-bit data with source and destination increments in bytes and byte-wise buffer copy. A bus arbitration size of eight can be used here.

The transfer buffer and transfer size are now configured. The transfer uses auto mode, which means that the transfer automatically runs to completion after the first request.

16.4.2.3 Start the Transfer

Finally, the channel must be enabled. A request must also be made because this is a software-initiated transfer. The request starts the transfer.

1. Enable global interrupts:

```
IntMasterEnable()
```

   and enable interrupt for DMA (DMA_DONE_COMB):

```
IntEnable(uint32_t ui32Interrupt)
```

2. Enable the channel by setting bit 0 of the DMA Set Channel Enable register, UDMA:SETCHANNELEN.
3. Issue a transfer request by setting bit 0 of the DMA Channel Software Request register, UDMA:SOFTREQ.
4. The µDMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer completes.

If needed, the status can be checked by reading the UDMA:SETCHANNELEN register bit 0. This bit is automatically cleared when the transfer completes.

16.5 UDMA Registers

Table 16-7 lists the memory-mapped registers for the UDMA registers. All register offset addresses not listed in Table 16-7 should be considered as reserved locations and the register contents should not be modified.

Table 16-7. UDMA Registers

Offset	Acronym	Register Name	Section
0h	STATUS	Status	Section 16.5.1
4h	CFG	Configuration	Section 16.5.2
8h	CTRL	Channel Control Data Base Pointer	Section 16.5.3
Ch	ALTCTRL	Channel Alternate Control Data Base Pointer	Section 16.5.4
10h	WAITONREQ	Channel Wait On Request Status	Section 16.5.5
14h	SOFTREQ	Channel Software Request	Section 16.5.6
18h	SETBURST	Channel Set UseBurst	Section 16.5.7
1Ch	CLEARBURST	Channel Clear UseBurst	Section 16.5.8
20h	SETREQMASK	Channel Set Request Mask	Section 16.5.9
24h	CLEARREQMASK	Clear Channel Request Mask	Section 16.5.10
28h	SETCHANNELEN	Set Channel Enable	Section 16.5.11
2Ch	CLEARCHANNELEN	Clear Channel Enable	Section 16.5.12
30h	SETCHNLPRIALT	Channel Set Primary-Alternate	Section 16.5.13
34h	CLEARCHNLPRIALT	Channel Clear Primary-Alternate	Section 16.5.14
38h	SETCHNLPRIORITY	Set Channel Priority	Section 16.5.15
3Ch	CLEARCHNLPRIORITY	Clear Channel Priority	Section 16.5.16
4Ch	ERROR	Error Status and Clear	Section 16.5.17
504h	REQDONE	Channel Request Done	Section 16.5.18
520h	DONEMASK	Channel Request Done Mask	Section 16.5.19

Complex bit access types are encoded to fit into small table cells. Table 16-8 shows the codes that are used for access types in this section.

Table 16-8. UDMA Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

16.5.1 STATUS Register (Offset = 0h) [Reset = 001F0000h]

STATUS is shown in Table 16-9.

Return to the Summary Table.

Status

Table 16-9. STATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	TEST	R	0h	0x0: Controller does not include the integration test logic 0x1: Controller includes the integration test logic 0x2: Undefined ... 0xF: Undefined
27-21	RESERVED	R	0h	Reserved
20-16	TOTALCHANNELS	R	1Fh	Register value returns number of available uDMA channels minus one. For example a read out value of: 0x00: Show that the controller is configured to use 1 uDMA channel 0x01: Shows that the controller is configured to use 2 uDMA channels ... 0x1F: Shows that the controller is configured to use 32 uDMA channels (32-1=31=0x1F)
15-8	RESERVED	R	0h	Reserved
7-4	STATE	R	0h	Current state of the control state machine. State can be one of the following: 0x0: Idle 0x1: Reading channel controller data 0x2: Reading source data end pointer 0x3: Reading destination data end pointer 0x4: Reading source data 0x5: Writing destination data 0x6: Waiting for uDMA request to clear 0x7: Writing channel controller data 0x8: Stalled 0x9: Done 0xA: Peripheral scatter-gather transition 0xB: Undefined ... 0xF: Undefined.
3-1	RESERVED	R	0h	Reserved
0	MASTERENABLE	R	0h	Shows the enable status of the controller as configured by CFG.MASTERENABLE: 0: Controller is disabled 1: Controller is enabled

16.5.2 CFG Register (Offset = 4h) [Reset = 00000000h]

CFG is shown in Table 16-10.

Return to the Summary Table.

Configuration

Table 16-10. CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-5	PRTOCTRL	W	0h	Sets the AHB-Lite bus protocol protection state by controlling the AHB signal HProt[3:1] as follows: Bit [7] Controls HProt[3] to indicate if a cacheable access is occurring. Bit [6] Controls HProt[2] to indicate if a bufferable access is occurring. Bit [5] Controls HProt[1] to indicate if a privileged access is occurring. When bit [n] = 1 then the corresponding HProt bit is high. When bit [n] = 0 then the corresponding HProt bit is low. This field controls HProt[3:1] signal for all transactions initiated by uDMA except two transactions below: - the read from the address indicated by source address pointer - the write to the address indicated by destination address pointer HProt[3:1] for these two exceptions can be controlled by dedicated fields in the channel configutation descriptor.
4-1	RESERVED	R	0h	Reserved
0	MASTERENABLE	W	0h	Enables the controller: 0: Disables the controller 1: Enables the controller

16.5.3 CTRL Register (Offset = 8h) [Reset = 00000000h]

CTRL is shown in Table 16-11.

Return to the Summary Table.

Channel Control Data Base Pointer

Table 16-11. CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	BASEPTR	R/W	0h	This register point to the base address for the primary data structures of each DMA channel. This is not stored in module, but in system memory, thus space must be allocated for this usage when DMA is in usage
9-0	RESERVED	R	0h	Reserved

16.5.4 ALTCTRL Register (Offset = Ch) [Reset = 00000200h]

ALTCTRL is shown in Table 16-12.

Return to the Summary Table.

Channel Alternate Control Data Base Pointer

Table 16-12. ALTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BASEPTR	R	200h	This register shows the base address for the alternate data structures and is calculated by module, thus read only

16.5.5 WAITONREQ Register (Offset = 10h) [Reset = FFFF1EFFh]

WAITONREQ is shown in Table 16-13.

Return to the Summary Table.

Channel Wait On Request Status

Table 16-13. WAITONREQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLSTATUS	R	FFFF1EFFh	Channel wait on request status: Bit [Ch] = 0: Once uDMA receives a single or burst request on channel Ch, this channel may come out of active state even if request is still present. Bit [Ch] = 1: Once uDMA receives a single or burst request on channel Ch, it keeps channel Ch in active state until the requests are deasserted. This handshake is necessary for channels where the requester is in an asynchronous domain or can run at slower clock speed than uDMA

16.5.6 SOFTREQ Register (Offset = 14h) [Reset = 00000000h]

SOFTREQ is shown in Table 16-14.

Return to the Summary Table.

Channel Software Request

Table 16-14. SOFTREQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Set the appropriate bit to generate a software uDMA request on the corresponding uDMA channel Bit [Ch] = 0: Does not create a uDMA request for channel Ch Bit [Ch] = 1: Creates a uDMA request for channel Ch Writing to a bit where a uDMA channel is not implemented does not create a uDMA request for that channel

16.5.7 SETBURST Register (Offset = 18h) [Reset = 00000000h]

SETBURST is shown in Table 16-15.

Return to the Summary Table.

Channel Set UseBurst

Table 16-15. SETBURST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Returns the useburst status, or disables individual channels from generating single uDMA requests. The value R is the arbitration rate and stored in the controller data structure. Read as: Bit [Ch] = 0: uDMA channel Ch responds to both burst and single requests on channel C. The controller performs 2^R, or single, bus transfers. Bit [Ch] = 1: uDMA channel Ch does not respond to single transfer requests. The controller only responds to burst transfer requests and performs 2^R transfers. Write as: Bit [Ch] = 0: No effect. Use the CLEARBURST.CHNLS to set bit [Ch] to 0. Bit [Ch] = 1: Disables single transfer requests on channel Ch. The controller performs 2^R transfers for burst requests. Writing to a bit where a uDMA channel is not implemented has no effect

16.5.8 CLEARBURST Register (Offset = 1Ch) [Reset = 00000000h]

CLEARBURST is shown in Table 16-16.

Return to the Summary Table.

Channel Clear UseBurst

Table 16-16. CLEARBURST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Set the appropriate bit to enable single transfer requests. Write as: Bit [Ch] = 0: No effect. Use the SETBURST.CHNLS to disable single transfer requests. Bit [Ch] = 1: Enables single transfer requests on channel Ch. Writing to a bit where a DMA channel is not implemented has no effect.

16.5.9 SETREQMASK Register (Offset = 20h) [Reset = 00000000h]

SETREQMASK is shown in Table 16-17.

Return to the Summary Table.

Channel Set Request Mask

Table 16-17. SETREQMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Returns the burst and single request mask status, or disables the corresponding channel from generating uDMA requests. Read as: Bit [Ch] = 0: External requests are enabled for channel Ch. Bit [Ch] = 1: External requests are disabled for channel Ch. Write as: Bit [Ch] = 0: No effect. Use the CLEARREQMASK.CHNLS to enable uDMA requests. Bit [Ch] = 1: Disables uDMA burst request channel [C] and uDMA single request channel [C] input from generating uDMA requests. Writing to a bit where a uDMA channel is not implemented has no effect

16.5.10 CLEARREQMASK Register (Offset = 24h) [Reset = 00000000h]

CLEARREQMASK is shown in Table 16-18.

Return to the Summary Table.

Clear Channel Request Mask

Table 16-18. CLEARREQMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Set the appropriate bit to enable DMA request for the channel. Write as: Bit [Ch] = 0: No effect. Use the SETREQMASK.CHNLS to disable channel C from generating requests. Bit [Ch] = 1: Enables channel [C] to generate DMA requests. Writing to a bit where a DMA channel is not implemented has no effect.

16.5.11 SETCHANNELEN Register (Offset = 28h) [Reset = 00000000h]

SETCHANNELEN is shown in Table 16-19.

Return to the Summary Table.

Set Channel Enable

Table 16-19. SETCHANNELEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Returns the enable status of the channels, or enables the corresponding channels. Read as: Bit [Ch] = 0: Channel Ch is disabled. Bit [Ch] = 1: Channel Ch is enabled. Write as: Bit [Ch] = 0: No effect. Use the CLEARCHANNELEN.CHNLS to disable a channel Bit [Ch] = 1: Enables channel Ch Writing to a bit where a DMA channel is not implemented has no effect

16.5.12 CLEARCHANNELEN Register (Offset = 2Ch) [Reset = 00000000h]

CLEARCHANNELEN is shown in Table 16-20.

Return to the Summary Table.

Clear Channel Enable

Table 16-20. CLEARCHANNELEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Set the appropriate bit to disable the corresponding uDMA channel. Write as: Bit [Ch] = 0: No effect. Use the SETCHANNELEN.CHNLS to enable uDMA channels. Bit [Ch] = 1: Disables channel Ch Writing to a bit where a uDMA channel is not implemented has no effect

16.5.13 SETCHNLPRIALT Register (Offset = 30h) [Reset = 00000000h]

SETCHNLPRIALT is shown in Table 16-21.

Return to the Summary Table.

Channel Set Primary-Alternate

Table 16-21. SETCHNLPRIALT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Returns the channel control data structure status, or selects the alternate data structure for the corresponding uDMA channel. Read as: Bit [Ch] = 0: uDMA channel Ch is using the primary data structure. Bit [Ch] = 1: uDMA channel Ch is using the alternate data structure. Write as: Bit [Ch] = 0: No effect. Use the CLEARCHNLPRIALT.CHNLS to disable a channel Bit [Ch] = 1: Selects the alternate data structure for channel Ch Writing to a bit where a uDMA channel is not implemented has no effect

16.5.14 CLEARCHNLPRIALT Register (Offset = 34h) [Reset = 00000000h]

CLEARCHNLPRIALT is shown in Table 16-22.

Return to the Summary Table.

Channel Clear Primary-Alternate

Table 16-22. CLEARCHNLPRIALT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Clears the appropriate bit to select the primary data structure for the corresponding uDMA channel. Write as: Bit [Ch] = 0: No effect. Use the SETCHNLPRIALT.CHNLS to select the alternate data structure. Bit [Ch] = 1: Selects the primary data structure for channel Ch. Writing to a bit where a uDMA channel is not implemented has no effect

16.5.15 SETCHNLPRIORITY Register (Offset = 38h) [Reset = 00000000h]

SETCHNLPRIORITY is shown in Table 16-23.

Return to the Summary Table.

Set Channel Priority

Table 16-23. SETCHNLPRIORITY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Returns the channel priority mask status, or sets the channel priority to high. Read as: Bit [Ch] = 0: uDMA channel Ch is using the default priority level. Bit [Ch] = 1: uDMA channel Ch is using a high priority level. Write as: Bit [Ch] = 0: No effect. Use the CLEARCHNLPRIORITY.CHNLS to set channel Ch to the default priority level. Bit [Ch] = 1: Channel Ch uses the high priority level. Writing to a bit where a uDMA channel is not implemented has no effect

16.5.16 CLEARCHNLPRIORITY Register (Offset = 3Ch) [Reset = 00000000h]

CLEARCHNLPRIORITY is shown in Table 16-24.

Return to the Summary Table.

Clear Channel Priority

Table 16-24. CLEARCHNLPRIORITY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	W	0h	Clear the appropriate bit to select the default priority level for the specified uDMA channel. Write as: Bit [Ch] = 0: No effect. Use the SETCHNLPRIORITY.CHNLS to set channel Ch to the high priority level. Bit [Ch] = 1: Channel Ch uses the default priority level. Writing to a bit where a uDMA channel is not implemented has no effect

16.5.17 ERROR Register (Offset = 4Ch) [Reset = 00000000h]

ERROR is shown in Table 16-25.

Return to the Summary Table.

Error Status and Clear

Table 16-25. ERROR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STATUS	R/W	0h	Returns the status of bus error flag in uDMA, or clears this bit Read as: 0: No bus error detected 1: Bus error detected Write as: 0: No effect, status of bus error flag is unchanged. 1: Clears the bus error flag.

16.5.18 REQDONE Register (Offset = 504h) [Reset = 00000000h]

REQDONE is shown in Table 16-26.

Return to the Summary Table.

Channel Request Done

Table 16-26. REQDONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Reflects the uDMA done status for the given channel, channel [Ch]. It's a sticky done bit. Unless cleared by writing a 1, it holds the value of 1. Read as: Bit [Ch] = 0: Request has not completed for channel Ch Bit [Ch] = 1: Request has completed for the channel Ch Writing a 1 to individual bits would clear the corresponding bit. Write as: Bit [Ch] = 0: No effect. Bit [Ch] = 1: The corresponding [Ch] bit is cleared and is set to 0

16.5.19 DONEMASK Register (Offset = 520h) [Reset = 00000000h]

DONEMASK is shown in Table 16-27.

Return to the Summary Table.

Channel Request Done Mask

Table 16-27. DONEMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CHNLS	R/W	0h	Controls the propagation of the uDMA done and active state to the assigned peripheral. Specifically used for software channels. Read as: Bit [Ch] = 0: uDMA done and active state for channel Ch is not blocked from reaching to the peripherals. Note that the uDMA done state for channel [Ch] is blocked from contributing to generation of combined uDMA done signal Bit [Ch] = 1: uDMA done and active state for channel Ch is blocked from reaching to the peripherals. Note that the uDMA done state for channel [Ch] is not blocked from contributing to generation of combined uDMA done signal Write as: Bit [Ch] = 0: Allows uDMA done and active stat to propagate to the peripherals. Note that this disables uDMA done state for channel [Ch] from contributing to generation of combined uDMA done signal Bit [Ch] = 1: Blocks uDMA done and active state to propagate to the peripherals. Note that this enables uDMA done for channel [Ch] to contribute to generation of combined uDMA done signal.

This page intentionally left blank.

Chapter 17

Timers

This chapter describes the general-purpose timers.

17.1 Introduction.....1372
17.2 Block Diagram.....1373
17.3 Functional Description.....1373
17.4 Initialization and Configuration.....1383
17.5 GPT Registers.....1387

17.1 Introduction

Programmable timers can be used to count or time external events that drive the timer input pins. The general-purpose timer module (GPTM) of the CC13x4x10 and CC26x4x10 device platform contains 4 GPTM blocks which all provide two 16-bit timers (Timer A and Timer B) that can be configured to operate independently as 16-bits timers or concatenated to operate as one 32-bit timer.

The GPTM is one timing resource available on the CC13x4x10 and CC26x4x10 device platform. Other timer resources include the system timer (SysTick) and the watchdog timer (WDT). For reference, see Section 2.3.3.3 and Chapter 19.

The GPTM contains four GPTM blocks with the following functional options:

- Operating modes:
  - 16-bit with an 8-bit prescaler or 32-bit programmable one-shot timer
  - 16-bit with an 8-bit prescaler or 32-bit programmable periodic timer
  - Two capture compare PWM pins (CCP) for each 32-bit timer
  - 24-bit input-edge count or 24-bit time-capture modes
  - 24-bit PWM mode with software-programmable output inversion of the PWM signal
- Count up or down
- Daisy chaining of timer modules allows a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- User-enabled stalling when the microcontroller asserts a CPU Halt flag during debug
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine

17.2 Block Diagram

Figure 17-1 shows the GPTM module block diagram.

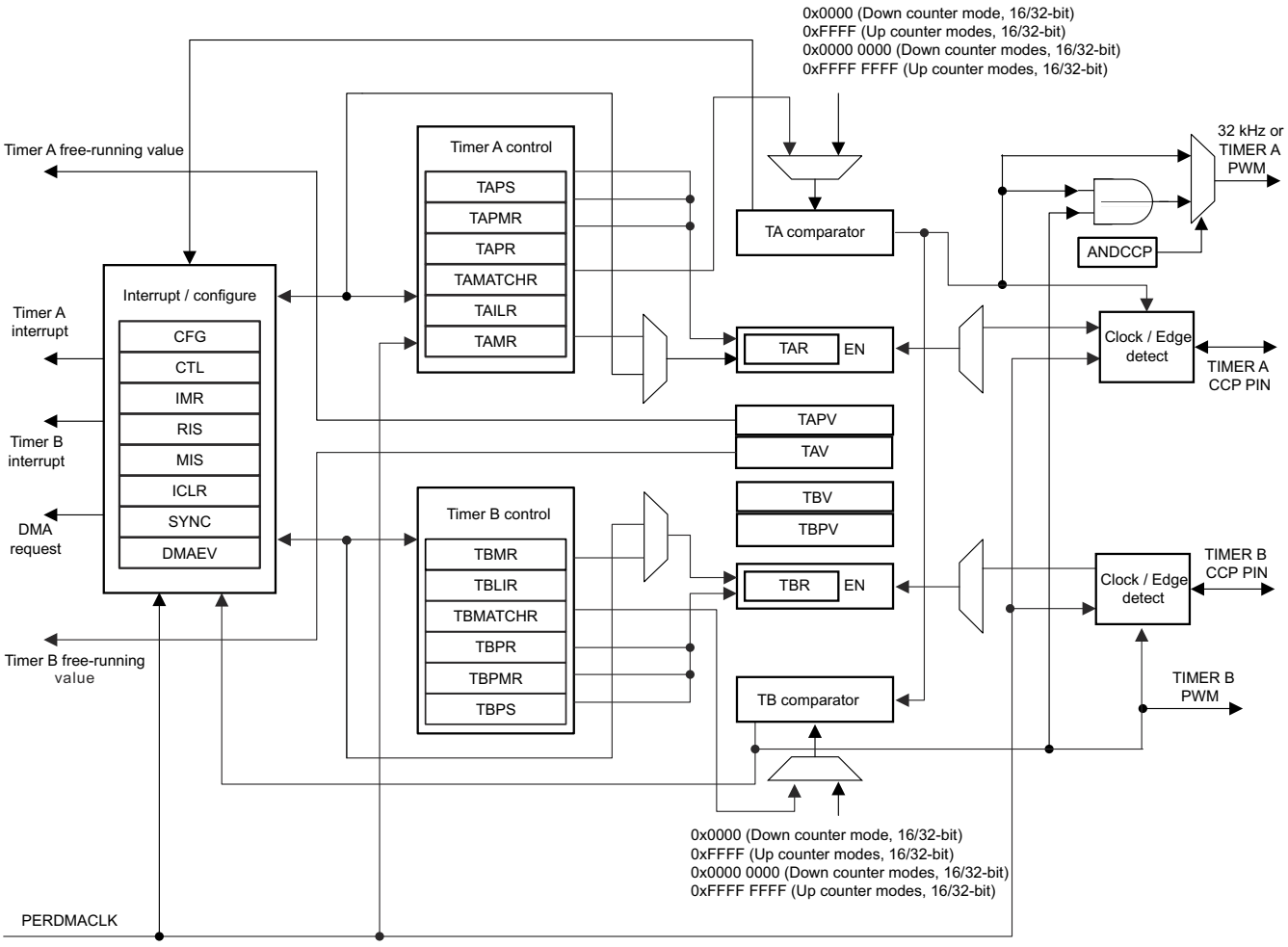


Figure 17-1. GPTM Module Block Diagram

17.3 Functional Description

The main components of each GPTM block are: two free-running up and down counters (Timer A and Timer B), two match registers, two prescaler match registers, two shadow registers, and two load and initialization registers and their associated control functions.

The function of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range. The prescaler can only be used when the timers are used individually.

PERDMACLK in Figure 17-1 is the system clock connected to the GPTM. The frequency of this clock can be changed through PRCM:GPTCLKDIV.

Table 17-1 lists the available modes for each GPTM block. When counting down in one-shot or periodic modes, the prescaler acts as a true prescaler and contains the least-significant bits (LSBs) of the count. When counting up in one-shot or periodic modes, the prescaler acts as a timer extension and holds the most-significant bits (MSBs) of the count. In input edge count, input edge time, and PWM mode, the prescaler always acts as a timer extension, regardless of the count direction.

Table 17-1. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size	Prescaler Size(1)	Prescaler Behavior (Count Direction)
One-Shot	Individual	Up or Down	16-bit	8-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	–	Not applicable
Periodic	Individual	Up or Down	16-bit	8-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	–	Not applicable
Edge Count	Individual	Up or Down	16-bit	8-bit	Timer Extension (Both)
Edge Time	Individual	Up or Down	16-bit	8-bit	Timer Extension (Both)
PWM	Individual	Down	16-bit	8-bit	Timer Extension

(1) The prescaler is available only when the timers are used individually.

Software configures the GPTM using the GPTM Configuration register (GPT:CFG), the GPTM Timer A Mode register (GPT:TAMR), and the GPTM Timer B Mode register (GPT:TBMR). When in one of the concatenated modes, Timer A and Timer B can operate in one mode only. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

17.3.1 GPTM Reset Conditions

After reset is applied to the GPTM, the module is in an inactive state, and all control registers are cleared and are in their default states. Counters (Timer A and Timer B) are initialized to all 1s, along with their corresponding load registers: the GPTM Timer A Interval Load register (GPT:TAILR) and the GPTM Timer B Interval Load register (GPT:TBILR). The prescale counters are initialized to 0x00:

- The GPTM Timer A Prescale register (GPT:TAPR) and the GPTM Timer B Prescale register (GPT:TBPR)
- The GPTM Timer A Prescale Snapshot register (GPT:TAPS) and the GPTM Timer B Prescale Snapshot register (GPT:TBPS)
- The GPTM Timer A Prescale Value register (GPT:TAPV) and the GPTM Timer B Prescale Value register (GPT:TBPV)

17.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use the Timer B control and status bits. The GPTM is placed into individual or split mode by writing a value of 0x4 to the GPTM Configuration register (GPT:CFG). In the following sections, the variable n is used in bit field and register names to imply either a Timer A function or a Timer B function. Throughout this section, the time-out event in down-count mode is 0x0; in up-count mode the time-out event is the value in the GPTM Interval Load register (GPT:TnILR) and the optional GPTM Timer n Prescale register (GPT:TnPR).

17.3.2.1 One-Shot or Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to GPT:TnMR.TnMR. The timer is configured to count up or down using the GPT:TnMR.TnCDIR bit.

When software sets the GPT:CTL.TnEN bit, the timer begins counting up from 0x0, or down from its preloaded value. Alternatively, if the GPT:TnMR.TnWOT is set when the TnEN bit is set, the timer waits for a trigger to begin counting (see Section 17.3.2.5).

When the timer is counting down and reaches the time-out event (0x0), the timer reloads its start value from the GPT:TnILR and the GPT:TnPR registers on the next cycle. When the timer is counting up and reaches the time-out event (the value in the GPT:TnILR and the optional GPT:TnPR registers), the timer reloads with 0x0. If configured to be a one-shot timer, the timer stops counting and clears the GPT:CTL.TnEN register bit. If configured as a periodic timer, the timer starts counting again on the next cycle. In periodic snap-shot mode (GPT:TnMR.TnMR = 0x2 and GPT:TnMR.TnSNAPS = 1), the actual free-running value of the timer at the time-out event is loaded into the GPT:TnR register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry by examining the snapshot values and the current value of the free-running timer, which is stored in the GPT:TnV register. Snapshot mode is not available when the timer is configured in one-shot mode.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the GPT:RIS.TnTORIS bit, and holds the bit until it is cleared by writing the GPTM Interrupt Clear register (GPT:ICR). If the time-out interrupt is enabled in the GPTM Interrupt Mask register (GPT:IMR), the GPTM also sets the GPTM Masked Interrupt Status register bit (GPT:MIS.TnTOMIS). By setting the GPT:TnMR.TnMIE register bit, an interrupt condition can also be generated when the timer value equals the value loaded into the GPTM Timer n Match register (GPT:TnMATCHR) and the GPTM Timer n Prescale Match register (GPT:TnPMR). This interrupt has the same status, masking, and clearing functions as the time-out interrupt but uses the match interrupt bits instead (for example, the raw interrupt status is monitored through the GPT:RIS.TnMRIS bit). The interrupt status bits are not updated by the hardware unless the GPT:TnMR.TnMIE register bit is set, which is different than the behavior for the time-out interrupt.

If software updates the GPT:TnILR or the GPT:TnPR registers while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting from the new value if the GPT:TnMR.TnILD bit is clear. If the TnILD bit is set, the counter loads the new value after the next time out. If software updates the GPT:TnILR register or the GPT:TnPR register while the counter is counting up, the time-out event is changed on the next cycle to the new value. If software updates GPT:TnV while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the GPT:TnMATCHR register or the GPT:TnPMR register while the counter is counting, the match registers reflect the new values on the next clock cycle if the GPT:TnMR.TnMRSU bit is clear. If the TnMRSU bit is set, the new value does not take effect until the next time out.

If the GPT:CTL.TnSTALL bit is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

Table 17-2 lists a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume a 24 MHz clock with Tc = 41.67 ns (clock period). The prescaler can only be used when a 16- or 32-bit timer is configured in 16-bit mode.

Table 17-2. 16-Bit Timer With Prescaler Configurations

Prescale (8-bit Value)	Number of Timer Clocks (Tc)(1)	Maximum Time	Unit
00000000	1	2.7	ms
00000001	2	5.5	ms
00000010	3	8.2	ms
–	–	–	–
11111101	254	693.6	ms
11111110	255	696.3	ms
11111111	256	699.1	ms

(1) Tc is the clock period.

17.3.2.2 Input Edge-Count Mode

Note

For rising-edge detection, the input signal must be high for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is ¼ of the system frequency.

In edge-count mode, the timer is configured as a 24-bit down counter, including the optional prescaler with the upper count value stored in the GPTM Timer n Prescale register (GPT:TnPR) and the lower bits in the GPT:TnR register. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in edge-count mode, the GPT:TnMR register TnCMR bit must be cleared. The type of edge that the timer counts is determined by the GPT:CTL register TnEVENT fields. During initialization in down-count mode, the GPT:TnMATCHR and the GPT:TnPMR registers are configured so that the difference between the value in the GPT:TnILR and the GPT:TnPR registers and the GPT:TnMATCHR and the GPT:TnPMR registers equals the number of edge events that must be counted. In up-count mode, the timer counts from 0x0 to the value in the GPT:TnMATCHR and the GPT:TnPMR registers. Table 17-3 lists the values that are loaded into the timer registers when the timer is enabled.

Table 17-3. Counter Values When the Timer is Enabled in Input Edge-Count Mode

Register	Count Down Mode	Count Up Mode
GPT:TnR	GPT:TnILR	0x0
GPT:TnV	GPT:TnILR	0x0
GPT:TnPV	GPT:TnPR	0x0

When software writes the GPTM Control register (GPT:CTL) TnEN bit, the timer is enabled for event capture. Each input event on the CCP pin decrements or increments the counter by 1 until the event count matches the GPT:TnMATCHR and the GPT:TnPMR registers. When the counts match, the GPTM asserts the GPTM Raw Interrupt Status register (GPT:RIS) CnMRIS bit, and holds the bit until it is cleared by writing the GPTM Interrupt Clear register (GPT:ICR). If the capture mode match interrupt is enabled in the GPTM Interrupt Mask register (GPT:IMR), the GPTM also sets the GPTM Masked Interrupt Status register (GPT:MIS) CnMMIS bit. In this mode, the GPT:TnR register holds the count of the input events while the GPT:TnV and the GPT:TnPV registers hold the free-running timer value and the free-running prescaler value.

In addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel.

After the match value is reached in down-count mode, the counter is then reloaded using the value in the GPT:TnILR and the GPT:TnPR registers, and stopped because the GPTM automatically clears the GPT:CTL.TnEN register bit. Once the event count has been reached, all further events are ignored until the TnEN bit is re-enabled by software. In up-count mode, the timer is reloaded with 0x0 and continues counting.

Figure 17-2 shows how Input edge-count mode works. In this case, the timer start value is set to GPT:TnILR = 0x000A, and the match value is set to GPT:TnMATCHR = 0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note

The last two edges are not counted, because the timer automatically clears the TnEN bit after the current count matches the value in the GPT:TnMATCHR register.

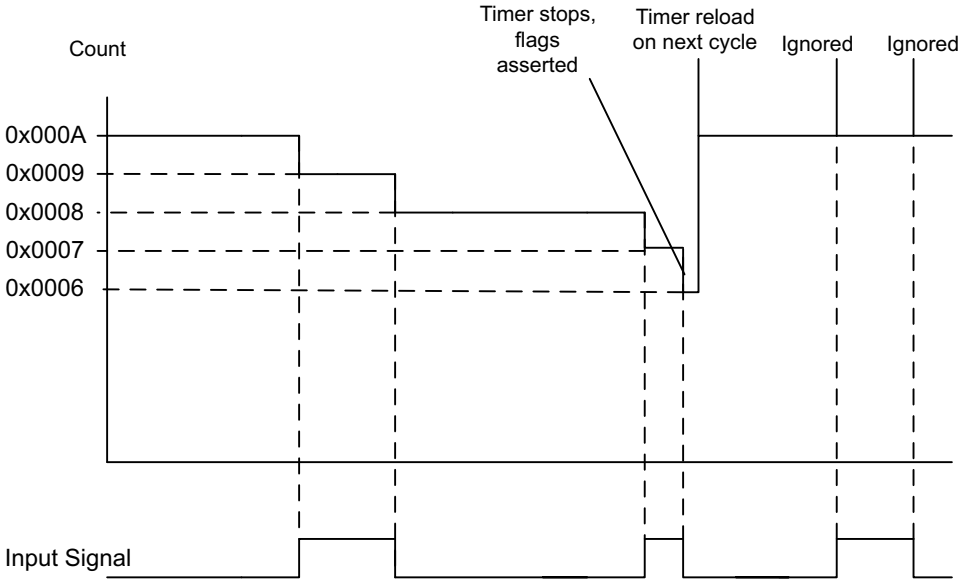


Figure 17-2. Input Edge-Count Mode Example, Counting Down

17.3.2.3 Input Edge-Time Mode

Note

For rising-edge detection, the input signal must be high for at least two system-clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be low for at least two system-clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is ¼ of the system frequency.

In edge-time mode, the timer is configured as a 24-bit down counter, including the optional prescaler with the upper timer value stored in the GPT:TnPR register and the lower bits in the GPT:TnILR register. In this mode, the timer is initialized to the value loaded in the GPT:TnILR and the GPT:TnPR registers when counting down and 0x0 when counting up. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into edge-time mode by setting the GPT:TnMR.TnCM register bit, and the type of event that the timer captures is determined by the GPT:CTL.TnEVENT register fields. Table 17-4 lists the values that are loaded into the timer registers when the timer is enabled.

Table 17-4. Counter Values When the Timer is Enabled in Input Event-Count Mode

Register	Count Down Mode	Count Up Mode
GPT:TnR	GPT:TnILR	0x0
GPT:TnV	GPT:TnILR	0x0
GPT:TnPV	GPT:TnPR	0x0

When software writes to the GPT:CTL.TnEN bit, the timer is enabled for event capture. When the selected input event is detected, the current timer counter value is captured in the GPT:TnR register and is available to be read by the microcontroller. The GPTM then asserts the GPT:RIS.CnERIS bit, and holds the bit until it is cleared by writing the GPTM Interrupt Clear register (GPT:ICR). If the capture mode event interrupt is enabled in the GPTM Interrupt Mask register (GPT:IMR), the GPTM also sets the GPT:MIS.CnEMIS bit. In this mode, the GPT:TnR register holds the time at which the selected input event occurred, while the GPT:TnV and the GPT:TnPV registers hold the free-running timer value and the free-running prescaler value. These registers can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

In addition to generating interrupts a µDMA trigger can be generated. This trigger is enabled by configuring and enabling the appropriate µDMA channel.

After an event has been captured, the timer does not stop counting. The timer continues to count until the TnEN bit is cleared. When the timer reaches the timeout value, it is reloaded with 0x0 in up-count mode, and the value from the GPT:TnILR and the GPT:TnPR registers in down-count mode.

Figure 17-3 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising-edge events.

Each time a rising-edge event is detected, the current count value is loaded into the GPTIMER:TnR register, and is held there until another rising edge is detected (at which point the new count value is loaded into the GPT:TnR register).

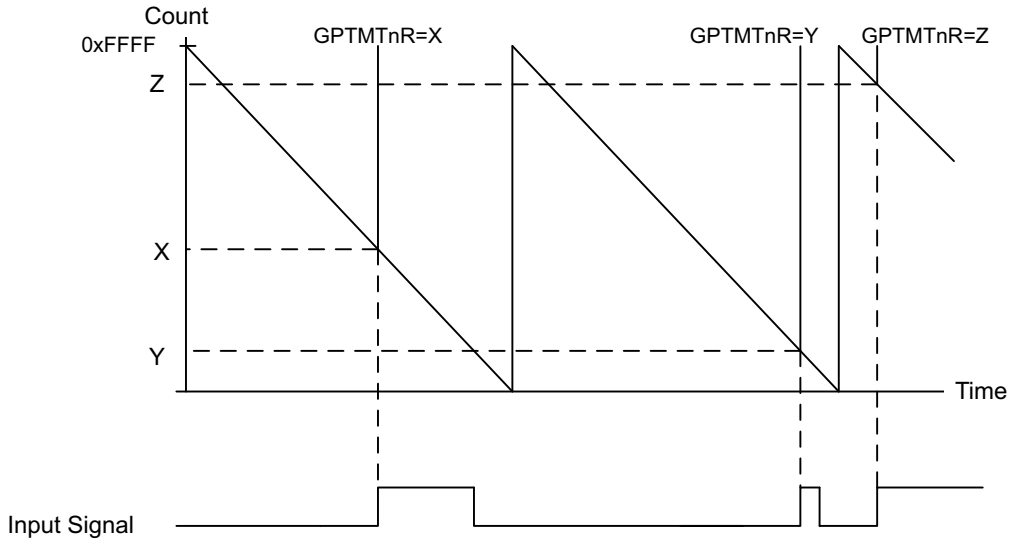


Figure 17-3. Input Edge-Time Mode Example

Note

When operating in edge-time mode, the counter uses a modulo 2^{24} count if prescaler is enabled, or 2^{16} if prescaler is not enabled. If there is a possibility the edge could take longer than the count, another timer can be used to ensure detection of the missed edge.

17.3.2.4 PWM Mode

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 16-bit down counter with a start value (and thus, period) defined by the GPT:TnILR and the GPT:TnPR registers. In this mode, the PWM frequency and period are synchronous events; therefore, they are ensured to be glitch-free. PWM mode is enabled with the GPT:TnMR register by setting the TnAMS bit to 0x1, setting the TnCM bit to 0x0, and setting the TnMR field to 0x2. Table 17-5 lists the values that are loaded into the timer registers when the timer is enabled.

Note

Wait on trigger (daisy chaining) is not supported in PWM mode. The timer starts as soon as it is enabled and does not wait for a trigger from the previous timer.

Table 17-5. Counter Values When the Timer is Enabled in PWM Mode

Register	Count Down Mode	Count Up Mode
GPT:TnR	GPT:TnILR	Not Available
GPT:TnV	GPT:TnILR	Not Available
GPT:TnPV	GPT:TnPR	Not Available

When software writes to the GPT:CTL.TnEN register bit, the counter begins counting down until it reaches the 0x0 state. Alternatively, if the GPT:TnMR.TnWOT register bit is set when the TnEN bit is set, the timer waits for a trigger to begin counting. On the next counter cycle in periodic mode, the counter reloads its start value from the GPT:TnILR and the GPT:TnPR registers, and continues counting until disabled by software clearing the GPT:CTL.TnEN register bit. The timer is capable of generating interrupts based on three types of events: rising edge, falling edge, or both. The event is configured by the GPT:CTL.TnEVENT register field, and the interrupt is enabled by setting the GPT:TnMR.TnPWMIE bit. When the event occurs, the GPT:RIS.CnERIS bit is set, and holds the bit until it is cleared by writing the GPTM Interrupt Clear register (GPT:ICLR). If the capture mode event interrupt is enabled in the GPTM Interrupt Mask register (GPT:IMR), the GPTM also sets the GPT:MIS.CnEMIS bit.

Note

The interrupt status bits are not updated unless the TnPWMIE bit is set.

In PWM mode, the GPT:TnR and the GPT:TnV registers always have the same value, as do the GPT:TnPS and the GPT:TnPV registers.

The output PWM signal asserts when the counter is at the value of the GPT:TnILR and the GPT:TnPR registers (its start state), and is deasserted when the counter value equals the value in the GPT:TnMATCHR and the GPT:TnPMR registers. Software can invert the output PWM signal by setting the GPT:CTL.TnPWML bit. Inverting the output PWM does not affect the edge detection interrupt. Therefore, if a positive-edge interrupt trigger has been set, the event-trigger interrupt is asserted when the PWM inversion generates a positive edge.

Note

If TnILR is altered to a value smaller than the current counter value LD_TO_EN must be enabled to avoid transients on the PWM output. This is true even when the “Time Out UPDATE” mode is enabled.

Figure 17-4 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50 MHz input clock and TnPWML = 0 (duty cycle would be 33% for the TnPWML = 1 configuration). For this example, the start value is GPT:TnILR = 0xC34F and the match value is GPT:TnMATCHR = 0x411A.

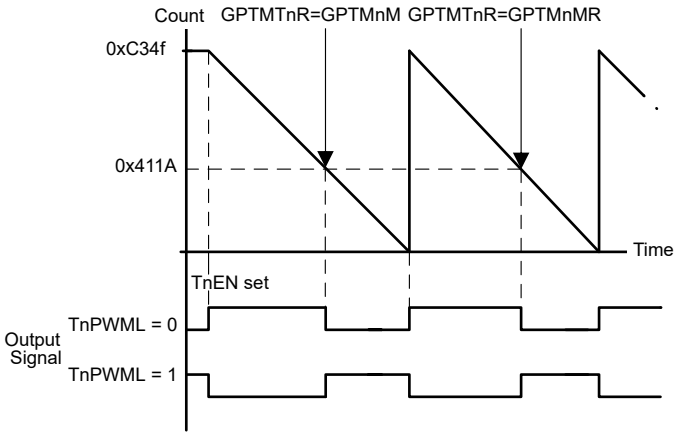


Figure 17-4. 16-Bit PWM Mode Example

When synchronizing the timers using the GPT:SYNC register, the timer must be properly configured to avoid glitches on the CCP outputs. Both the GPT:TnMR.TnPLO and the GPT:TnMR.TnMRSU bits must be set. Figure 17-5 shows how the CCP output operates when the TnPLO and TnMRSU bits are set and the GPT:TnMATCHR register value is greater than the GPT:TnILR register value.

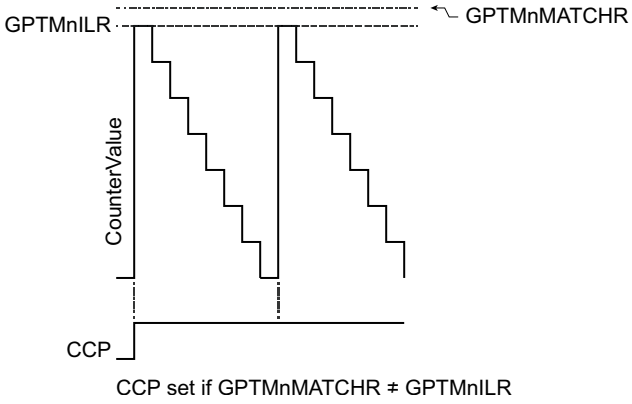


Figure 17-5. CCP Output, GPT:TnMATCHR > GPT:TnILR

Figure 17-6 shows how the CCP output operates when the GPT:TnMR.TnPLO and GPT:TnMR.TnMRSU bits are set and the GPT:TnMATCHR register value is the same as the GPT:TnILR register value. In this situation, if the TnPLO bit is 0, the CCP signal goes high when the GPT:TnILR register value is loaded, and the match would be essentially ignored.

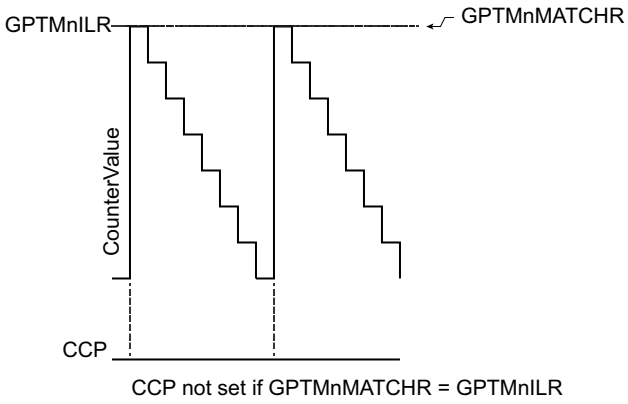


Figure 17-6. CCP Output, GPT:TnMATCHR = GPT:TnILR

Figure 17-7 shows how the CCP output operates when the GPT:TnMR.TnPLO and GPT:TnMR.TnMRSU bits are set and the GPT:TnILR register value is greater than the GPT:TnMATCHR register value.

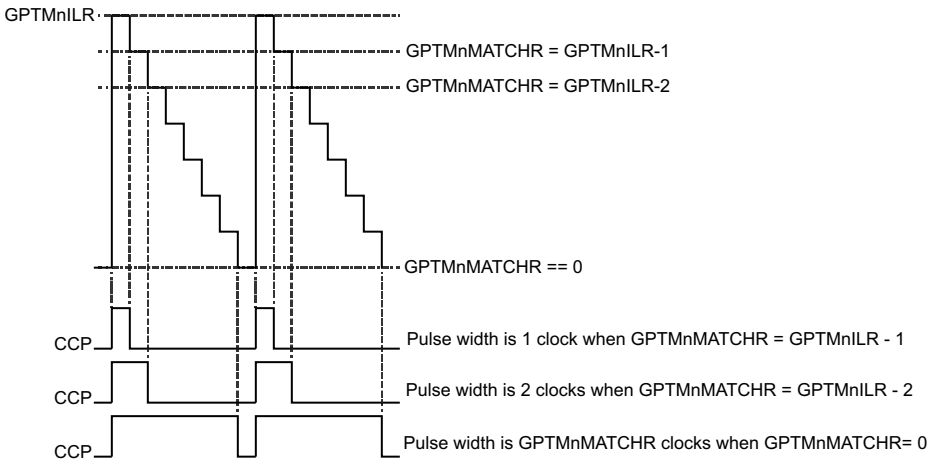


Figure 17-7. CCP Output, GPT:TnILR > GPT:TnMATCHR

17.3.2.5 Wait-for-Trigger Mode

Wait-for-trigger mode allows daisy-chaining of the timer modules such that once configured, a single timer can initiate multiple timing events using the timer triggers. Wait-for-trigger mode is enabled by setting the GPT:TnMR.TnWOT register bit. When the TnWOT bit is set, timer N+1 does not begin counting until the timer in the previous position in the daisy-chain (timer N) reaches its time-out event. The daisy-chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so forth. If Timer A is configured as a 32-bit timer (controlled by the CFG field in the GPT:CFG register), it triggers Timer A in the next module. If Timer A is configured as a 16-bit timer, it triggers Timer B in the same module and Timer B triggers Timer A in the next module. Ensure that the TAWOT bit is never set in GPTM0. Figure 17-8 shows how the GPT:CFG.CFG register bit affects the daisy-chain. This function is valid for one-shot and periodic modes.

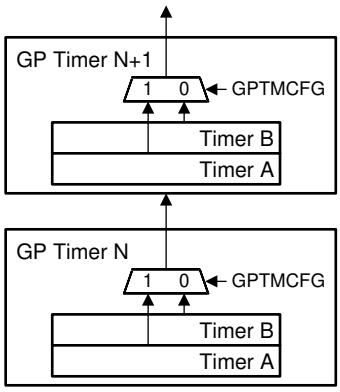


Figure 17-8. Timer Daisy-Chain

17.3.3 Synchronizing GPT Blocks

The GPTM Synchronizer Control register (GPT:SYNC) in the GPTM0 block can be used to synchronize selected timers to begin counting at the same time. To do so, the timers must be started first. Setting a bit in the GPT:SYNC register causes the associated timer to perform the actions of a time-out event. An interrupt is not generated when the timers are synchronized. If a timer is being used in concatenated mode, only the bit for Timer A must be set in the GPT:SYNC register. The register description shows which timers can be synchronized.

Table 17-6 lists the actions for the time-out event performed when the timers are synchronized in the various timer modes.

Table 17-6. Time-Out Actions for GPTM Modes

Mode	Count Direction	Time-Out Action
16-bit and 32-bit one-shot (concatenated timers)	—	Not applicable
16- bit and 32-bit periodic (concatenated timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit and 32-bit one-shot (individual and split timers)	—	Not applicable
16-bit and 32-bit periodic (individual and split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit and 32-bit edge-count (individual and split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit and 32-bit edge-time (individual and split timers)	Down	Count value = ILR
	Up	Count value = 0
16-bit PWM	Down	Count value = ILR

17.3.4 Accessing Concatenated 16- and 32-Bit GPTM Register Values

The GPTM is placed into concatenated mode by writing a 0x0 to the GPT:CFG.CFG bit field. For this configurations, certain registers are concatenated to form pseudo 32-bit registers. These registers include the following:

- GPTM Timer A Interval Load register (GPT:TAILR[15:0])
- GPTM Timer B Interval Load register (GPT:TBILR[15:0])
- GPTM Timer A register (GPT:TAR[15:0])
- GPTM Timer B register (GPT:TBR[15:0])
- GPTM Timer A Value register (GPT:TAV[15:0])
- GPTM Timer B Value register (GPT:TBV[15:0])
- GPTM Timer A Match register (GPT:TAMATCHR[15:0])
- GPTM Timer B Match register (GPT:TBMATCHR[15:0])

In the 32-bit modes, the GPTM translates a 32-bit write access to the GPT:TAILR register into a write access to both the GPT:TAILR and the GPT:TBILR registers. The resulting word ordering for such a write operation is: GPTMTBILR[15:0]:GPTMTAILR[15:0]. Likewise, a 32-bit read access to GPT:TAR register returns the value: GPTMTBR[15:0]:GPTMTAR[15:0]. A 32-bit read access to GPT:TAV returns the value: GPTMTBV[15:0]:GPTMTAV[15:0].

17.4 Initialization and Configuration

The following described the necessary steps to enable and initialize the GPTM.

TI recommends using the GPTimer driver in the SimpleLink™ CC13xx and CC26xx software development kit (SDK) when using the GPTM.

1. To use a GPT module, enable the peripheral domain and the appropriate GPT module in the PRCM by writing to the PRCM:GPTCLKGR, the PRCM:GPTCLKGS, and the PRCM:GPTCLKGDS registers.
2. Next, load the setting to the clock controller by writing to the PRCM:CLKLOADCTL register.
3. Configure the IOC module to route the output from the GPT module to the IOs.
    - The IOC module must then be configured to output the timer signal on the wanted I/O pin. For this, IOCFGn.PORT_ID must be written to the correct PORT_IDs (for more details, see Chapter 15).

The following sections show module initialization and configuration examples for each of the supported timer modes.

17.4.1 One-Shot and Periodic Timer Modes

Configure the GPTM for one-shot and periodic modes with the following sequence:

1. Ensure the timer is disabled (clear the GPT:CTL.TnEN bit) before making any changes.
2. Write the GPTM Configuration register (GPT:CFG) with a value of 0x00000000 (32-bit timer) or 0x00000004 (16-bit timer).
3. Configure the GPT:TnMR.TnMR field:
   - Write a value of 0x1 for one-shot mode
   - Write a value of 0x2 for periodic mode
4. Optionally, configure the TnSNAPS, TnWOT, TnMIE, and TnCDIR fields in the GPT:TnMR register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the GPTM Timer n Interval Load register (GPT:TnILR).
6. If interrupts are required, set the appropriate bits in the GPTM Interrupt Mask register (GPT:IMR).
7. Set the GPT:CTL.TnEN bit to enable the timer and start counting.
8. Poll the GPT:MRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the GPTM Interrupt Clear register (GPT:ICR).

In one-shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in periodic mode reloads the timer and continues counting after the time-out event.

17.4.2 Input Edge-Count Mode

Configure a timer to input edge-count mode with the following sequence:

1. Ensure the timer is disabled (clear the GPT:CTL.TnEN bit) before making any changes.
2. Write the GPTM Configuration register (GPT:CFG) with a value of 0x00000004.
3. In the GPTM Timer Mode register (GPT:TnMR), write the TnCM field to 0x0 and the TnMR field to 0x3.
4. Configure the type of events that the timer captures by writing the GPT:CTL.TnEVENT field.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale register (GPT:TnPR).
6. Load the timer start value into the GPTM Timer n Interval Load register (GPT:TnILR).
7. Load the event count into the GPTM Timer n Match register (GPT:TnMATCHR).
8. If interrupts are required, set the GPT:IMR.CnMIM bit.
9. Set the GPT:CTL.TnEN bit to enable the timer and begin waiting for edge events.
10. Poll the GPT:RIS.CnMRIS bit, or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the GPT:ICR.CnMCINT bit.

When counting down in input edge-count mode, the timer stops after the programmed number of edge events is detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat Step 4 through Step 9.

17.4.3 Input Edge-Timing Mode

Configure a timer to input edge-timing mode with the following sequence:

1. Ensure the timer is disabled (clear the GPT:CTL.TnEN bit) before making any changes.
2. Write the GPTM Configuration register (GPT:CFG) with a value of 0x00000004.
3. In the GPTM Timer Mode register (GPT:TnMR), write the TnCM field to 0x1 and write the TnMR field to 0x3.
4. Configure the type of events that the timer captures by writing the GPT:CTL.TnEVENT field.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale register (GPT:TnPR).
6. Load the timer start value into the GPTM Timer n Interval Load register (GPT:TnILR).
7. If interrupts are required, set the GPT:IMR.CnMIM bit.
8. Set the GPT:CTL.TnEN bit to enable the timer and start counting.
9. Poll the GPT:RIS.CnMRIS bit, or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to GPT:ICR.CnMCINT.

In input-edge timing mode, the timer continues to run after an edge event is detected, but the timer interval can be changed at any time by writing the GPT:TnILR register. The change takes effect at the next cycle after the write.

17.4.4 PWM Mode

Configure a timer to PWM mode with the following sequence:

1. Ensure the timer is disabled (clear the GPT:CTL.TnEN bit) before making any changes.
2. Write the GPTM Configuration register (GPT:CFG) with a value of 0x0000 0004.
3. In the GPTM Timer Mode register (GPT:TnMR), write the TnCM field to 0x1 and write the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the GPTM Control register (GPT:CTL) TnPWML field.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale register (GPT:TnPR).
6. If PWM interrupts are used, configure the interrupt condition in the GPT:CTL.TnEVENT register field, and enable the interrupts by setting the GPT:TnMR.TnPWMIE register bit.
7. Load the timer start value into the GPTM Timer n Interval Load register (GPT:TnILR).
8. Load the GPTM Timer n Match register (GPT:TnMATCHR) with the match value.
9. Set the GPTM Control register (GPT:CTL) TnEN bit to enable the timer and begin generation of the output PWM signal.

For step 5 and 8, the GPT:TnMR.TnMRSU bit defines when the GPT:TnMATCHR and GPT:TnPR registers are updated.

In PWM timing mode, the timer continues to run after the PWM signal is generated. The PWM period can be adjusted at any time by writing the GPT:TnILR register, and the change takes effect at the next cycle after the write.

17.4.5 Producing DMA Trigger Events

The GPT can produce DMA trigger events through the event handler. Single or burst requests can be passed to the µDMA controller by selecting the trigger source for µDMA channels through the event fabric. Each timer only produces one signal per A and B, but this signal can be selected as either single or burst in the event module. The DMA done interrupt is routed back to the timer module that originated the trigger.

Use the following procedure to configure µDMA triggers by GPT events:

1. Configure the GPT operation.
2. Configure the GPT:DMAEV register to enable the appropriate timer event to DMA. The application can select a match, capture, or time-out event for each timer.
3. Configure the event fabric (see Chapter 5) to select the appropriate timer. The event fabric supports five channels for the GPT DMA event, out of which four are dedicated to the GPT block. These dedicated channels are: 9, 10, 11, and 12. Single requests and burst requests are supported on channels 9 through 12 for GPT DMA events. The fifth supported channel is 14, which is configurable for GPT support and handles only burst requests. The configuration is done through the EVENT:UDMACHcrSEL register where *c* is the channel number and *r* is either the S (single) or B (burst) option. Channel 14 can be configured using the EVENT:UDMACH14BSEL register. Each timer produces only one signal per A and B, but this signal can be selected as either single or burst in the event module.
4. Enable the GPT.
5. The DMA done interrupt is routed back to the timer module that originated the trigger. The GPT:RIS DMAnRIS register bit gives the DMA transfer completed information.

17.5 GPT Registers

Table 17-7 lists the memory-mapped registers for the GPT registers. All register offset addresses not listed in Table 17-7 should be considered as reserved locations and the register contents should not be modified.

Table 17-7. GPT Registers

Offset	Acronym	Register Name	Section
0h	CFG	Configuration	Section 17.5.1
4h	TAMR	Timer A Mode	Section 17.5.2
8h	TBMR	Timer B Mode	Section 17.5.3
Ch	CTL	Control	Section 17.5.4
10h	SYNC	Synch Register	Section 17.5.5
18h	IMR	Interrupt Mask	Section 17.5.6
1Ch	RIS	Raw Interrupt Status	Section 17.5.7
20h	MIS	Masked Interrupt Status	Section 17.5.8
24h	ICLR	Interrupt Clear	Section 17.5.9
28h	TAILR	Timer A Interval Load Register	Section 17.5.10
2Ch	TBILR	Timer B Interval Load Register	Section 17.5.11
30h	TAMATCHR	Timer A Match Register	Section 17.5.12
34h	TBMATCHR	Timer B Match Register	Section 17.5.13
38h	TAPR	Timer A Pre-scale	Section 17.5.14
3Ch	TBPR	Timer B Pre-scale	Section 17.5.15
40h	TAPMR	Timer A Pre-scale Match	Section 17.5.16
44h	TBPMR	Timer B Pre-scale Match	Section 17.5.17
48h	TAR	Timer A Register	Section 17.5.18
4Ch	TBR	Timer B Register	Section 17.5.19
50h	TAV	Timer A Value	Section 17.5.20
54h	TBV	Timer B Value	Section 17.5.21
5Ch	TAPS	Timer A Pre-scale Snap-shot	Section 17.5.22
60h	TBPS	Timer B Pre-scale Snap-shot	Section 17.5.23
64h	TAPV	Timer A Pre-scale Value	Section 17.5.24
68h	TBPV	Timer B Pre-scale Value	Section 17.5.25
6Ch	DMAEV	DMA Event	Section 17.5.26
FB0h	VERSION	Peripheral Version	Section 17.5.27
FB4h	ANDCCP	Combined CCP Output	Section 17.5.28

Complex bit access types are encoded to fit into small table cells. Table 17-8 shows the codes that are used for access types in this section.

Table 17-8. GPT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

17.5.1 CFG Register (Offset = 0h) [Reset = 00000000h]

CFG is shown in Table 17-9.

Return to the Summary Table.

Configuration

Table 17-9. CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	CFG	R/W	0h	GPT Configuration 0x2- 0x3 - Reserved 0x5- 0x7 - Reserved 0h = 32BIT_TIMER : 32-bit timer configuration 4h = 16BIT_TIMER : 16-bit timer configuration. Configure for two 16-bit timers. Also see TAMR.TAMR and TBMR.TBMR.

17.5.2 TAMR Register (Offset = 4h) [Reset = 00000000h]

TAMR is shown in Table 17-10.

Return to the Summary Table.

Timer A Mode

Table 17-10. TAMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-13	TCACT	R/W	0h	Timer Compare Action Select 0h = DIS_CMP : Disable compare operations 1h = Toggle State on Time-Out 2h = Clear CCP output pin on Time-Out 3h = Set CCP output pin on Time-Out 4h = Set CCP output pin immediately and toggle on Time-Out 5h = Clear CCP output pin immediately and toggle on Time-Out 6h = Set CCP output pin immediately and clear on Time-Out 7h = Clear CCP output pin immediately and set on Time-Out
12	TACINTD	R/W	0h	One-Shot/Periodic Interrupt Disable 0h = Time-out interrupt function as normal 1h = Time-out interrupt are disabled
11	TAPLO	R/W	0h	GPTM Timer A PWM Legacy Operation 0 Legacy operation with CCP pin driven Low when the TAILR register is reloaded after the timer reaches 0. 1 CCP is driven High when the TAILR register is reloaded after the timer reaches 0. This bit is only valid in PWM mode. 0h = Legacy operation 1h = CCP output pin is set to 1 on time-out
10	TAMRSU	R/W	0h	Timer A Match Register Update mode This bit defines when the TAMATCHR and TAPR registers are updated. If the timer is disabled (CTL.TAEN = 0) when this bit is set, TAMATCHR and TAPR are updated when the timer is enabled. If the timer is stalled (CTL.TASTALL = 1) when this bit is set, TAMATCHR and TAPR are updated according to the configuration of this bit. 0h = Update TAMATCHR and TAPR, if used, on the next cycle. 1h = Update TAMATCHR and TAPR, if used, on the next time-out.
9	TAPWMIE	R/W	0h	GPTM Timer A PWM Interrupt Enable This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the CTL.TAEVENT In addition, when this bit is set and a capture event occurs, Timer A automatically generates triggers to the DMA if the trigger capability is enabled by setting the CTL.TAOTE bit and the DMAEV.CAEDMAEN bit respectively. 0 Capture event interrupt is disabled. 1 Capture event interrupt is enabled. This bit is only valid in PWM mode. 0h = Interrupt is disabled. 1h = Interrupt is enabled. This bit is only valid in PWM mode.
8	TAILD	R/W	0h	GPT Timer A PWM Interval Load Write 0h = Update the TAR register with the value in the TAILR register on the next clock cycle. If the pre-scaler is used, update the TAPS register with the value in the TAPR register on the next clock cycle. 1h = Update the TAR register with the value in the TAILR register on the next timeout. If the prescaler is used, update the TAPS register with the value in the TAPR register on the next timeout.
7	TASNAPS	R/W	0h	GPT Timer A Snap-Shot Mode 0h = Snap-shot mode is disabled. 1h = If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the GPT Timer A (TAR) register.

Table 17-10. TAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	TAWOT	R/W	0h	GPT Timer A Wait-On-Trigger 0h = Timer A begins counting as soon as it is enabled. 1h = If Timer A is enabled (CTL.TAEN = 1), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This bit must be clear for GPT Module 0, Timer A. This function is valid for one-shot, periodic, and PWM modes
5	TAMIE	R/W	0h	GPT Timer A Match Interrupt Enable 0h = The match interrupt is disabled for match events. Additionally, output triggers on match events are prevented. 1h = An interrupt is generated when the match value in TAMATCHR is reached in the one-shot and periodic modes.
4	TACDIR	R/W	0h	GPT Timer A Count Direction 0h = DOWN : The timer counts down. 1h = UP : The timer counts up. When counting up, the timer starts from a value of 0x0.
3	TAAMS	R/W	0h	GPT Timer A Alternate Mode Note: To enable PWM mode, you must also clear TACM and then configure TAMR field to 0x2. 0h = Capture/Compare mode is enabled. 1h = PWM mode is enabled
2	TACM	R/W	0h	GPT Timer A Capture Mode 0h = EDGCNT : Edge-Count mode 1h = EDGTIME : Edge-Time mode
1-0	TAMR	R/W	0h	GPT Timer A Mode 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the CFG register 1h = One-Shot Timer mode 2h = Periodic Timer mode 3h = Capture mode

17.5.3 TBMR Register (Offset = 8h) [Reset = 00000000h]

TBMR is shown in Table 17-11.

Return to the Summary Table.

Timer B Mode

Table 17-11. TBMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-13	TCACT	R/W	0h	Timer Compare Action Select 0h = DIS_CMP : Disable compare operations 1h = Toggle State on Time-Out 2h = Clear CCP output pin on Time-Out 3h = Set CCP output pin on Time-Out 4h = Set CCP output pin immediately and toggle on Time-Out 5h = Clear CCP output pin immediately and toggle on Time-Out 6h = Set CCP output pin immediately and clear on Time-Out 7h = Clear CCP output pin immediately and set on Time-Out
12	TBCINTD	R/W	0h	One-Shot/Periodic Interrupt Mode 0h = Normal Time-Out Interrupt 1h = Mask Time-Out Interrupt
11	TBPLO	R/W	0h	GPTM Timer B PWM Legacy Operation 0 Legacy operation with CCP pin driven Low when the TBILR register is reloaded after the timer reaches 0. 1 CCP is driven High when the TBILR register is reloaded after the timer reaches 0. This bit is only valid in PWM mode. 0h = Legacy operation 1h = CCP output pin is set to 1 on time-out
10	TBMRSU	R/W	0h	Timer B Match Register Update mode This bit defines when the TBMATCHR and TBPR registers are updated If the timer is disabled (CTL.TBEN is clear) when this bit is set, TBMATCHR and TBPR are updated when the timer is enabled. If the timer is stalled (CTL.TBSTALL is set) when this bit is set, TBMATCHR and TBPR are updated according to the configuration of this bit. 0h = Update TBMATCHR and TBPR, if used, on the next cycle. 1h = Update TBMATCHR and TBPR, if used, on the next time-out.
9	TBPWMIE	R/W	0h	GPTM Timer B PWM Interrupt Enable This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the CTL.TBEVENT In addition, when this bit is set and a capture event occurs, Timer A automatically generates triggers to the DMA if the trigger capability is enabled by setting the CTL.TBOTE bit and the DMAEV.CBEDMAEN bit respectively. 0 Capture event interrupt is disabled. 1 Capture event interrupt is enabled. This bit is only valid in PWM mode. 0h = Interrupt is disabled. 1h = Interrupt is enabled. This bit is only valid in PWM mode.
8	TBILD	R/W	0h	GPT Timer B PWM Interval Load Write 0h = Update the TBR register with the value in the TBILR register on the next clock cycle. If the pre-scaler is used, update the TBPS register with the value in the TBPR register on the next clock cycle. 1h = Update the TBR register with the value in the TBILR register on the next timeout. If the prescaler is used, update the TBPS register with the value in the TBPR register on the next timeout.
7	TBSNAPS	R/W	0h	GPT Timer B Snap-Shot Mode 0h = Snap-shot mode is disabled. 1h = If Timer B is configured in the periodic mode

Table 17-11. TBMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	TBWOT	R/W	0h	GPT Timer B Wait-On-Trigger 0h = Timer B begins counting as soon as it is enabled. 1h = If Timer B is enabled (CTL.TBEN is set), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes
5	TBMIE	R/W	0h	GPT Timer B Match Interrupt Enable. 0h = The match interrupt is disabled for match events. Additionally, output triggers on match events are prevented. 1h = An interrupt is generated when the match value in the TBMATCHR register is reached in the one-shot and periodic modes.
4	TBCDIR	R/W	0h	GPT Timer B Count Direction 0h = DOWN : The timer counts down. 1h = UP : The timer counts up. When counting up, the timer starts from a value of 0x0.
3	TBAMS	R/W	0h	GPT Timer B Alternate Mode Note: To enable PWM mode, you must also clear TBCM bit and configure TBMR field to 0x2. 0h = Capture/Compare mode is enabled. 1h = PWM mode is enabled
2	TBCM	R/W	0h	GPT Timer B Capture Mode 0h = EDGCNT : Edge-Count mode 1h = EDGTIME : Edge-Time mode
1-0	TBMR	R/W	0h	GPT Timer B Mode 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the CFG register 1h = One-Shot Timer mode 2h = Periodic Timer mode 3h = Capture mode

17.5.4 CTL Register (Offset = Ch) [Reset = 00000000h]

CTL is shown in Table 17-12.

Return to the Summary Table.

Control

Table 17-12. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	TBPWML	R/W	0h	GPT Timer B PWM Output Level 0: Output is unaffected. 1: Output is inverted. 0h = Not inverted 1h = Inverted
13-12	RESERVED	R	0h	Reserved
11-10	TBEVENT	R/W	0h	GPT Timer B Event Mode The values in this register are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges Note: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a postive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal. 0h = Positive edge 1h = Negative edge 3h = Both edges
9	TBSTALL	R/W	0h	GPT Timer B Stall Enable 0h = Timer B continues counting while the processor is halted by the debugger. 1h = Timer B freezes counting while the processor is halted by the debugger.
8	TBEN	R/W	0h	GPT Timer B Enable 0h = Timer B is disabled. 1h = Timer B is enabled and begins counting or the capture logic is enabled based on CFG register.
7	RESERVED	R	0h	Reserved
6	TAPWML	R/W	0h	GPT Timer A PWM Output Level 0h = Not inverted 1h = Inverted
5-4	RESERVED	R	0h	Reserved
3-2	TAEVENT	R/W	0h	GPT Timer A Event Mode The values in this register are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges Note: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a postive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal. 0h = Positive edge 1h = Negative edge 3h = Both edges

Table 17-12. CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	TASTALL	R/W	0h	GPT Timer A Stall Enable 0h = Timer A continues counting while the processor is halted by the debugger. 1h = Timer A freezes counting while the processor is halted by the debugger.
0	TAEN	R/W	0h	GPT Timer A Enable 0h = Timer A is disabled. 1h = Timer A is enabled and begins counting or the capture logic is enabled based on the CFG register.

17.5.5 SYNC Register (Offset = 10h) [Reset = 00000000h]

SYNC is shown in Table 17-13.

Return to the Summary Table.

Synch Register

Table 17-13. SYNC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-6	SYNC3	W	0h	Synchronize GPT Timer 3. 0h = No Sync. GPT3 is not affected. 1h = A timeout event for Timer A of GPT3 is triggered 2h = A timeout event for Timer B of GPT3 is triggered 3h = A timeout event for both Timer A and Timer B of GPT3 is triggered
5-4	SYNC2	W	0h	Synchronize GPT Timer 2. 0h = No Sync. GPT2 is not affected. 1h = A timeout event for Timer A of GPT2 is triggered 2h = A timeout event for Timer B of GPT2 is triggered 3h = A timeout event for both Timer A and Timer B of GPT2 is triggered
3-2	SYNC1	W	0h	Synchronize GPT Timer 1 0h = No Sync. GPT1 is not affected. 1h = A timeout event for Timer A of GPT1 is triggered 2h = A timeout event for Timer B of GPT1 is triggered 3h = A timeout event for both Timer A and Timer B of GPT1 is triggered
1-0	SYNC0	W	0h	Synchronize GPT Timer 0 0h = No Sync. GPT0 is not affected. 1h = A timeout event for Timer A of GPT0 is triggered 2h = A timeout event for Timer B of GPT0 is triggered 3h = A timeout event for both Timer A and Timer B of GPT0 is triggered

17.5.6 IMR Register (Offset = 18h) [Reset = 00000000h]

IMR is shown in Table 17-14.

Return to the Summary Table.

Interrupt Mask
This register is used to enable the interrupts.
Associated registers:
RIS, MIS, ICLR

Table 17-14. IMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	DMABIM	R/W	0h	Enabling this bit will make the RIS.DMABRIS interrupt propagate to MIS.DMABMIS 0h = Disable Interrupt 1h = Enable Interrupt
12	RESERVED	R	0h	Reserved
11	TBMIM	R/W	0h	Enabling this bit will make the RIS.TBMRIS interrupt propagate to MIS.TBMMIS 0h = Disable Interrupt 1h = Enable Interrupt
10	CBEIM	R/W	0h	Enabling this bit will make the RIS.CBERIS interrupt propagate to MIS.CBEMIS 0h = Disable Interrupt 1h = Enable Interrupt
9	CBMIM	R/W	0h	Enabling this bit will make the RIS.CBMRIS interrupt propagate to MIS.CBMMIS 0h = Disable Interrupt 1h = Enable Interrupt
8	TBTOIM	R/W	0h	Enabling this bit will make the RIS.TBTORIS interrupt propagate to MIS.TBTOMIS 0h = Disable Interrupt 1h = Enable Interrupt
7-6	RESERVED	R	0h	Reserved
5	DMAAIM	R/W	0h	Enabling this bit will make the RIS.DMAARIS interrupt propagate to MIS.DMAAMIS 0h = Disable Interrupt 1h = Enable Interrupt
4	TAMIM	R/W	0h	Enabling this bit will make the RIS.TAMRIS interrupt propagate to MIS.TAMMIS 0h = Disable Interrupt 1h = Enable Interrupt
3	RESERVED	R	0h	Reserved
2	CAEIM	R/W	0h	Enabling this bit will make the RIS.CAERIS interrupt propagate to MIS.CAEMIS 0h = Disable Interrupt 1h = Enable Interrupt
1	CAMIM	R/W	0h	Enabling this bit will make the RIS.CAMRIS interrupt propagate to MIS.CAMMIS 0h = Disable Interrupt 1h = Enable Interrupt
0	TATOIM	R/W	0h	Enabling this bit will make the RIS.TATORIS interrupt propagate to MIS.TATOMIS 0h = Disable Interrupt 1h = Enable Interrupt

17.5.7 RIS Register (Offset = 1Ch) [Reset = 00000000h]

RIS is shown in Table 17-15.

Return to the Summary Table.

Raw Interrupt Status
Associated registers:
IMR, MIS, ICLR

Table 17-15. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	DMABRIS	R	0h	GPT Timer B DMA Done Raw Interrupt Status 0: Transfer has not completed 1: Transfer has completed
12	RESERVED	R	0h	Reserved
11	TBMRIS	R	0h	GPT Timer B Match Raw Interrupt 0: The match value has not been reached 1: The match value is reached. TBMR.TBMIE is set, and the match values in TBMATCHR and optionally TBPMR have been reached when configured in one-shot or periodic mode.
10	CBERIS	R	0h	GPT Timer B Capture Mode Event Raw Interrupt 0: The event has not occured. 1: The event has occured. This interrupt asserts when the subtimer is configured in Input Edge-Time mode
9	CBMRIS	R	0h	GPT Timer B Capture Mode Match Raw Interrupt 0: The capture mode match for Timer B has not occurred. 1: A capture mode match has occurred for Timer B. This interrupt asserts when the values in the TBR and TBPR match the values in the TBMATCHR and TBPMR when configured in Input Edge-Time mode. This bit is cleared by writing a 1 to the ICLR.CBMCINT bit.
8	TBTORIS	R	0h	GPT Timer B Time-out Raw Interrupt 0: Timer B has not timed out 1: Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit. The count limit is 0 or the value loaded into TBILR, depending on the count direction.
7-6	RESERVED	R	0h	Reserved
5	DMAARIS	R	0h	GPT Timer A DMA Done Raw Interrupt Status 0: Transfer has not completed 1: Transfer has completed
4	TAMRIS	R	0h	GPT Timer A Match Raw Interrupt 0: The match value has not been reached 1: The match value is reached. TAMR.TAMIE is set, and the match values in TAMATCHR and optionally TAPMR have been reached when configured in one-shot or periodic mode.
3	RESERVED	R	0h	Reserved
2	CAERIS	R	0h	GPT Timer A Capture Mode Event Raw Interrupt 0: The event has not occured. 1: The event has occured. This interrupt asserts when the subtimer is configured in Input Edge-Time mode

Table 17-15. RIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	CAMRIS	R	0h	GPT Timer A Capture Mode Match Raw Interrupt 0: The capture mode match for Timer A has not occurred. 1: A capture mode match has occurred for Timer A. This interrupt asserts when the values in the TAR and TAPR match the values in the TAMATCHR and TAPMR when configured in Input Edge-Time mode. This bit is cleared by writing a 1 to the ICLR.CAMCINT bit.
0	TATORIS	R	0h	GPT Timer A Time-out Raw Interrupt 0: Timer A has not timed out 1: Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit. The count limit is 0 or the value loaded into TAILR, depending on the count direction.

17.5.8 MIS Register (Offset = 20h) [Reset = 00000000h]

MIS is shown in Table 17-16.

Return to the Summary Table.

Masked Interrupt Status
Values are result of bitwise AND operation between RIS and IMR
Assosciated clear register: ICLR

Table 17-16. MIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	DMABMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.DMABRIS = 1 && IMR.DMABIM = 1
12	RESERVED	R	0h	Reserved
11	TBMMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.TBMRIS = 1 && IMR.TBMIM = 1
10	CBEMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.CBERIS = 1 && IMR.CBEIM = 1
9	CBMMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.CBMRIS = 1 && IMR.CBMIM = 1
8	TBTOMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.TBTORIS = 1 && IMR.TBTOIM = 1
7-6	RESERVED	R	0h	Reserved
5	DMAAMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.DMAARIS = 1 && IMR.DMAAIM = 1
4	TAMMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.TAMRIS = 1 && IMR.TAMIM = 1
3	RESERVED	R	0h	Reserved
2	CAEMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.CAERIS = 1 && IMR.CAEIM = 1
1	CAMMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.CAMRIS = 1 && IMR.CAMIM = 1
0	TATOMIS	R	0h	0: No interrupt or interrupt not enabled 1: RIS.TATORIS = 1 && IMR.TATOIM = 1

17.5.9 ICLR Register (Offset = 24h) [Reset = 00000000h]

ICLR is shown in Table 17-17.

Return to the Summary Table.

Interrupt Clear
This register is used to clear status bits in the RIS and MIS registers

Table 17-17. ICLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	DMABINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.DMABRIS and MIS.DMABMIS
12	RESERVED	R	0h	Reserved
11	TBMCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.TBMRIS and MIS.TBMMIS
10	CBECINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.CBERIS and MIS.CBEMIS
9	CBMCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.CBMRIS and MIS.CBMMIS
8	TBTOCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.TBTORIS and MIS.TBTOMIS
7-6	RESERVED	R	0h	Reserved
5	DMAAINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.DMAARIS and MIS.DMAAMIS
4	TAMCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.TAMRIS and MIS.TAMMIS
3	RESERVED	R	0h	Reserved
2	CAECINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.CAERIS and MIS.CAEMIS
1	CAMCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.CAMRIS and MIS.CAMMIS
0	TATOCINT	R/W1C	0h	0: Do nothing. 1: Clear RIS.TATORIS and MIS.TATOMIS

17.5.10 TAILR Register (Offset = 28h) [Reset = FFFFFFFFh]

TAILR is shown in Table 17-18.

Return to the Summary Table.

Timer A Interval Load Register

Table 17-18. TAILR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAILR	R/W	FFFFFFFFh	GPT Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of TAILR.

17.5.11 TBILR Register (Offset = 2Ch) [Reset = 0000FFFFh]

TBILR is shown in Table 17-19.

Return to the Summary Table.

Timer B Interval Load Register

Table 17-19. TBILR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBILR	R/W	FFFFh	GPT Timer B Interval Load Register Writing this field loads the counter for Timer B. A read returns the current value of TBILR.

17.5.12 TAMATCHR Register (Offset = 30h) [Reset = FFFFFFFFh]

TAMATCHR is shown in Table 17-20.

Return to the Summary Table.

Timer A Match Register
Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.
In Edge-Count mode, this register along with TAILR, determines how many edge events are counted.
The total number of edge events counted is equal to the value in TAILR minus this value.
Note that in edge-count mode, when executing an up-count, the value of TAPR and TAILR must be greater than the value of TAPMR and this register.
In PWM mode, this value along with TAILR, determines the duty cycle of the output PWM signal.
When a 16/32-bit GPT is configured to one of the 32-bit modes, TAMATCHR appears as a 32-bit register. (The upper 16-bits correspond to the contents TBMATCHR).
In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of TBMATCHR.
Note : This register is updated internally (takes effect) based on TAMR.TAMRSU

Table 17-20. TAMATCHR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAMATCHR	R/W	FFFFFFFFh	GPT Timer A Match Register

17.5.13 TBMATCHR Register (Offset = 34h) [Reset = 0000FFFFh]

TBMATCHR is shown in Table 17-21.

Return to the Summary Table.

Timer B Match Register
When a GPT is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of TAMATCHR.
Reads from this register return the current match value of Timer B and writes are ignored.
In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are reserved in both cases.
Note : This register is updated internally (takes effect) based on TBMR.TBMRSU

Table 17-21. TBMATCHR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	TBMATCHR	R/W	FFFFh	GPT Timer B Match Register

17.5.14 TAPR Register (Offset = 38h) [Reset = 00000000h]

TAPR is shown in Table 17-22.

Return to the Summary Table.

Timer A Pre-scale
This register allows software to extend the range of the timers when they are used individually.
When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter.
When acting as a true prescaler, the prescaler counts down to 0 before the value in TAR and TAV registers are incremented.
In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPT.

Table 17-22. TAPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TAPSR	R/W	0h	Timer A Pre-scale. Prescaler ratio in one-shot and periodic count mode is TAPSR + 1, that is: 0: Prescaler ratio = 1 1: Prescaler ratio = 2 2: Prescaler ratio = 3 ... 255: Prescaler ratio = 256

17.5.15 TBPR Register (Offset = 3Ch) [Reset = 00000000h]

TBPR is shown in Table 17-23.

Return to the Summary Table.

Timer B Pre-scale
This register allows software to extend the range of the timers when they are used individually.
When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter.
When acting as a true prescaler, the prescaler counts down to 0 before the value in TBR and TBV registers are incremented.
In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPT.

Table 17-23. TBPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TBPSR	R/W	0h	Timer B Pre-scale. Prescale ratio in one-shot and periodic count mode is TBPSR + 1, that is: 0: Prescaler ratio = 1 1: Prescaler ratio = 2 2: Prescaler ratio = 3 ... 255: Prescaler ratio = 256

17.5.16 TAPMR Register (Offset = 40h) [Reset = 00000000h]

TAPMR is shown in Table 17-24.

Return to the Summary Table.

Timer A Pre-scale Match
This register allows software to extend the range of the TAMATCHR when used individually.

Table 17-24. TAPMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TAPSMR	R/W	0h	GPT Timer A Pre-scale Match. In 16 bit mode this field holds bits 23 to 16.

17.5.17 TBPMR Register (Offset = 44h) [Reset = 00000000h]

TBPMR is shown in Table 17-25.

Return to the Summary Table.

Timer B Pre-scale Match
This register allows software to extend the range of the TBMATCHR when used individually.

Table 17-25. TBPMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	TBPSMR	R/W	0h	GPT Timer B Pre-scale Match Register. In 16 bit mode this field holds bits 23 to 16.

17.5.18 TAR Register (Offset = 48h) [Reset = FFFFFFFFh]

TAR is shown in Table 17-26.

Return to the Summary Table.

Timer A Register
This register shows the current value of the Timer A counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that
have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.
When a GPT is configured to one of the 32-bit modes, this register appears as a 32-bit register (the upper 16-bits correspond to the contents of the Timer B (TBR) register). In
the16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits
31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the TAV register. To read the value of the prescalar in periodic snapshot
mode, read the Timer A Prescale Snapshot (TAPS) register.

Table 17-26. TAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAR	R	FFFFFFFFh	GPT Timer A Register Based on the value in the register field TAMR.TAILD, this register is updated with the value from TAILR register either on the next cycle or on the next timeout. A read returns the current value of the Timer A Count Register, in all cases except for Input Edge count and Timer modes. In the Input Edge Count Mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

17.5.19 TBR Register (Offset = 4Ch) [Reset = 0000FFFFh]

TBR is shown in Table 17-27.

Return to the Summary Table.

Timer B Register
This register shows the current value of the Timer B counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that
have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.
When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the TAR register. Reads from this register return the current
value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the
upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the TBV register. To read the value of the
prescalar in periodic snapshot mode, read the Timer B Prescale Snapshot (TBPS) register.

Table 17-27. TBR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBR	R	FFFFh	GPT Timer B Register Based on the value in the register field TBMR.TBILD, this register is updated with the value from TBILR register either on the next cycle or on the next timeout. A read returns the current value of the Timer B Count Register, in all cases except for Input Edge count and Timer modes. In the Input Edge Count Mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

17.5.20 TAV Register (Offset = 50h) [Reset = FFFFFFFFh]

TAV is shown in Table 17-28.

Return to the Summary Table.

Timer A Value
When read, this register shows the current, free-running value of Timer A in all modes. Softwarecan use this value to determine the time elapsed between an interrupt and the ISR entry when using
the snapshot feature with the periodic operating mode. When written, the value written into this register is loaded into the TAR register on the next clock cycle.
When a 16/32-bit GPTM is configured to one of the 32-bit modes, this register appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Value (TBV) register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic
down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

Table 17-28. TAV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TAV	R/W	FFFFFFFFh	GPT Timer A Register A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the TAR register on the next clock cycle. Note: In 16-bit mode, only the lower 16-bits of this register can be written with a new value. Writes to the prescaler bits have no effect

17.5.21 TBV Register (Offset = 54h) [Reset = 0000FFFFh]

TBV is shown in Table 17-29.

Return to the Summary Table.

Timer B Value
When read, this register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the TBR register on the next clock cycle.
When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the TAV register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes.
In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

Table 17-29. TBV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TBV	R/W	FFFFh	GPT Timer B Register A read returns the current, free-running value of Timer B in all modes. When written, the value written into this register is loaded into the TBR register on the next clock cycle. Note: In 16-bit mode, only the lower 16-bits of this register can be written with a new value. Writes to the prescaler bits have no effect

17.5.22 TAPS Register (Offset = 5Ch) [Reset = 00000000h]

TAPS is shown in Table 17-30.

Return to the Summary Table.

Timer A Pre-scale Snap-shot
Based on the value in the register field TAMR.TAILD, this register is updated with the value from TAPR register either on the next cycle or on the next timeout.
This register shows the current value of the Timer A pre-scaler in the 16-bit mode.

Table 17-30. TAPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PSS	R	0h	GPT Timer A Pre-scaler

17.5.23 TBPS Register (Offset = 60h) [Reset = 00000000h]

TBPS is shown in Table 17-31.

Return to the Summary Table.

Timer B Pre-scale Snap-shot
Based on the value in the register field TBMR.TBILD, this register is updated with the value from TBPR register either on the next cycle or on the next timeout.
This register shows the current value of the Timer B pre-scaler in the 16-bit mode.

Table 17-31. TBPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PSS	R	0h	GPT Timer B Pre-scaler

17.5.24 TAPV Register (Offset = 64h) [Reset = 00000000h]

TAPV is shown in Table 17-32.

Return to the Summary Table.

Timer A Pre-scale Value
This register shows the current value of the Timer A free running pre-scaler in the 16-bit mode.

Table 17-32. TAPV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PSV	R	0h	GPT Timer A Pre-scaler Value

17.5.25 TBPV Register (Offset = 68h) [Reset = 00000000h]

TBPV is shown in Table 17-33.

Return to the Summary Table.

Timer B Pre-scale Value
This register shows the current value of the Timer B free running pre-scaler in the 16-bit mode.

Table 17-33. TBPV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PSV	R	0h	GPT Timer B Pre-scaler Value

17.5.26 DMAEV Register (Offset = 6Ch) [Reset = 00000000h]

DMAEV is shown in Table 17-34.

Return to the Summary Table.

DMA Event
This register allows software to enable/disable GPT DMA trigger events.

Table 17-34. DMAEV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Software should not rely on the value of a reserved field. Writing any other value may result in undefined behavior.
11	TBMDMAEN	R/W	0h	GPT Timer B Match DMA Trigger Enable
10	CBEDMAEN	R/W	0h	GPT Timer B Capture Event DMA Trigger Enable
9	CBMDMAEN	R/W	0h	GPT Timer B Capture Match DMA Trigger Enable
8	TBTODMAEN	R/W	0h	GPT Timer B Time-Out DMA Trigger Enable
7-5	RESERVED	R/W	0h	Software should not rely on the value of a reserved field. Writing any other value may result in undefined behavior.
4	TAMDMAEN	R/W	0h	GPT Timer A Match DMA Trigger Enable
3	RESERVED	R	0h	Reserved
2	CAEDMAEN	R/W	0h	GPT Timer A Capture Event DMA Trigger Enable
1	CAMDMAEN	R/W	0h	GPT Timer A Capture Match DMA Trigger Enable
0	TATODMAEN	R/W	0h	GPT Timer A Time-Out DMA Trigger Enable

17.5.27 VERSION Register (Offset = FB0h) [Reset = 00000400h]

VERSION is shown in Table 17-35.

Return to the Summary Table.

Peripheral Version
This register provides information regarding the GPT version

Table 17-35. VERSION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VERSION	R	400h	Timer Revision.

17.5.28 ANDCCP Register (Offset = FB4h) [Reset = 00000000h]

ANDCCP is shown in Table 17-36.

Return to the Summary Table.

Combined CCP Output
This register is used to logically AND CCP output pairs for each timer

Table 17-36. ANDCCP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	LD_TO_EN	R/W	0h	PWM assertion would happen at timeout 0: PWM assertion happens when counter matches load value 1: PWM assertion happens at timeout of the counter
0	CCP_AND_EN	R/W	0h	Enables AND operation of the CCP outputs for timers A and B. 0 : PWM outputs of Timer A and Timer B are the internal generated PWM signals of the respective timers. 1 : PWM output of Timer A is ANDed version of Timer A and Timer B PWM signals and Timer B PWM ouput is Timer B PWM signal only.

Chapter 18

Real-Time Clock (RTC)

This chapter describes the functionality and design of the always-on, real-time clock (AON_RTC) for the CC13x4x10 and CC26x4x10 device platform.

18.1 Introduction 1422
18.2 Functional Specifications 1422
18.3 RTC Register Information 1424
18.4 RTC Registers 1426

18.1 Introduction

This section describes the functionality and design of the always-on, real-time clock (AON_RTC) for the CC13x4x10 and CC26x4x10 device platform. The AON_RTC implements a second and subsecond counter with support for software-compensation of ppm-offsets, with three match register and one compare register.

A special mechanism is in place to support power down of the MCU domain while the AON_RTC continues to operate. The AON_RTC is powered in all power modes except for the deepest power-down mode, known as Shutdown.

18.2 Functional Specifications

This section gives a functional description of the AON_RTC.

18.2.1 Functional Overview

The functionality of the AON_RTC is described as follows:

- Increments on positive edges of the 32 kHz clock
- A 70-bit incrementing counter with support for programmable increment to support ppm-adjustment
- Three general-purpose channels (0, 1, and 2) with comparators supporting the generation of events
- Software and hardware reset of events
- All events can be delayed by a programmable amount to generated corresponding delayed events.
- A programmable set of the delayed events can be combined to generate a delayed combined event.

18.2.2 Free-Running Counter

The AON_RTC implements a 70-bit, free-running counter incremented by a programmable value for each 32 kHz clock. The programmable value allows compensation of ppm-offsets in the 32 kHz clock, making it possible for the counter to operate with a very high precision.

The counter starts from 0 when enabled following power up of the AON_RTC, but can also be reset to 0 or any other new value by the software. The counter measures seconds (32 bit) and subseconds (32 bit).

By default, the AON_RTC increments its counter with 1/32768 seconds each 32 kHz clock tick. A subsecond increment value of 0x20 000 corresponds to 1/32768 seconds. Increasing or decreasing the subsecond increments value increases or decreases the speed of the AON_RTC by the same amount.

Change the increment by updating the AUX_SYSIF:RTCSUBSECINC0 and the AUX_SYSIF:RTCSUBSECINC1 registers, and then load the new setting to the AON_RTC by a write to the AUX_SYSIF:RTCSUBSECINCCTL.UPD_REQ register. The new subsecond increment value must not be changed by AUX until it has received an acknowledgment from the AON_RTC. The acknowledgment can be read from the AUX_SYSIF:RTCSUBSECINCCTL.UPD_ACK register. After the acknowledgment has been received, the AUX_SYSIF:RTCSUBSECINCCTL.UPD_REQ register can be written back to 0 and a new subsecond increment can be uploaded, if needed.

To perform an atomic read of the free-running counter, a read must first be done of the seconds part AON_RTC:SEC. This will latch the subseconds part AON_RTC:SUBSEC until read. In addition AON_RTC:TIME register returns the lower halfword of the AON_RTC:SEC register and the upper halfword of the AON_RTC:SUBSEC register. This is the same format as the match and capture registers.

18.2.3 Channels

The AON_RTC contains three independent channels (0, 1, and 2) that have slightly different behaviors.

All channels can operate in a compare mode; each channel generates a compare event when a programmable time limit has been reached or exceeded. This is the only mode of operation for channel 0.

Channel 1 can be operated in a capture mode, where an external event causes the current value of the free-running timer to be latched, to remember the time of the event. A capture event is subsequently generated. As the channel 1 timer is operating in either compare mode or capture mode, the same physical event is generated in each mode.

Channel 2 can operate in a continuous compare mode, automatically incrementing its compare value following a compare event. This enables the generation of completely equidistant events.

Figure 18-1 shows the three AON_RTC channels.

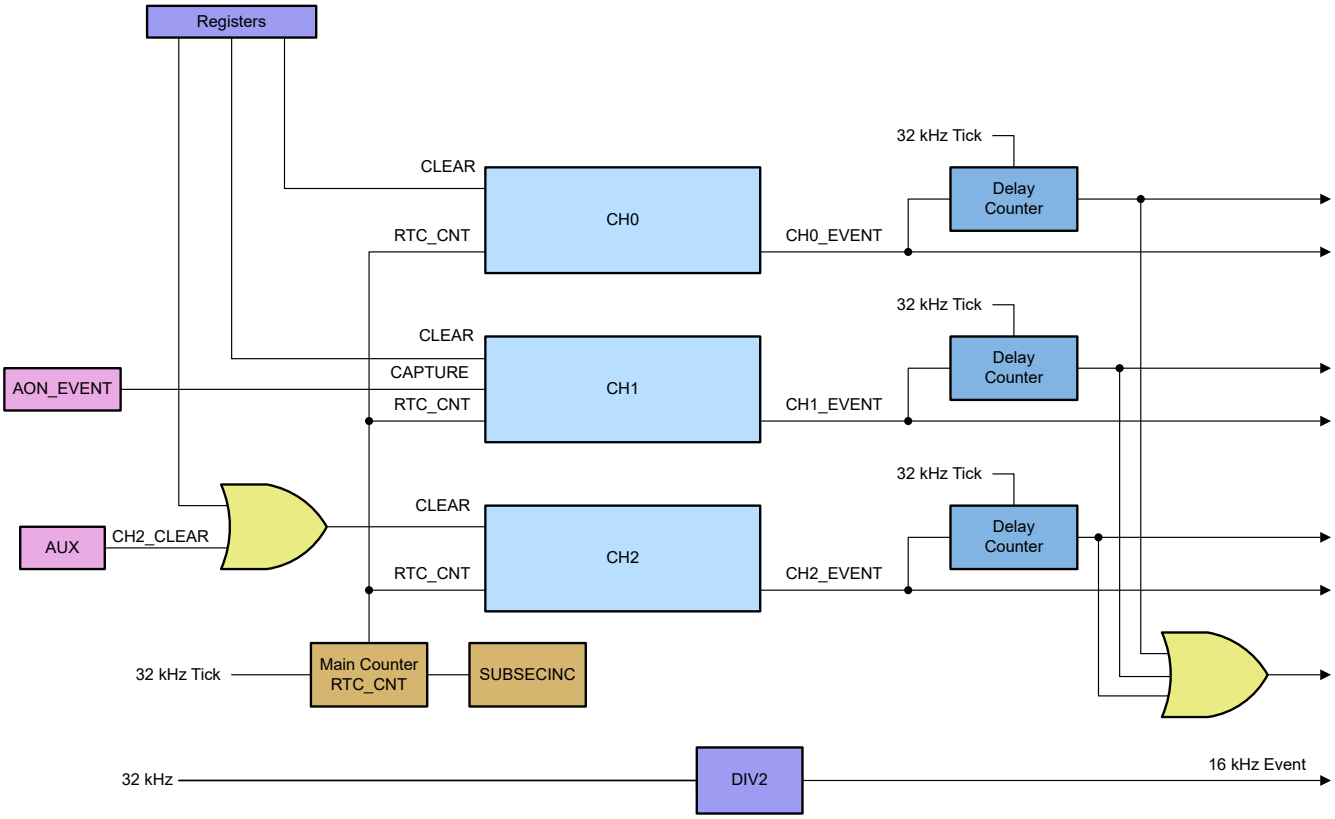


Figure 18-1. AON_RTC Channels

18.2.3.1 Capture and Compare

A RTC capture event can be set up on channel 1 by configuring a capture source in AON_EVENT:RTCSEL and setting channel 1 to capture mode in AON_RTC:CHCTL. The captured RTC value is available in the register AON_RTC:CH1CAPT after a capture event

Compare events are configured by writing to the corresponding channel compare register AON_RTC:CHnCMP.

Note

If a compare value is set so that the compare value minus current value is larger than the seconds wrap-around time minus one second ($2^{32} \times SCLK\_LF_{period} - 1$), an immediate compare event is set to avoid losing the event.

18.2.4 Events

A programmable combination of the three delayed events can be combined into a seventh delayed event. All events are fed to the AON event fabric, where they are available, for example, as wake-up events for the MCU or AUX domains.

The three channels can individually generate an event. Each of these three events can generate a corresponding event by delaying the channel event by a programmable amount of clocks, using a delay counter. The delay counters use the uncompensated 32-kHz clock; thus, the delay time varies with this clock. This process can generate precise events in the future, even when, for example, the MCU must be woken up following an event—a process that takes an nondeterministic, yet bounded, amount of time.

Note

Disabling a channel does not clear any pending events from that channel. The only way to clear an event is by asserting the external clear signal, or by writing 1 to the corresponding CHx bit in the AON_RTC:EVFLAGS register.

18.3 RTC Register Information

The RTC registers are placed in the AON domain and are clocked using 2 MHz MF clock. All configuration and status registers are preserved in all power modes except for Shutdown. The MCU domain contains an interface to the AON_RTC registers to ensure fast access with minimum latency on the system bus. Due to synchronization between the MCU interface and the AON domain, there is a delay in the system that software must take into account.

18.3.1 Register Access

A write access is delayed with one or two 2 MHz MF clock periods. The system bus is not affected by this delay, so the MCU completes the bus transactions before the actual AON_RTC register is written in the AON_RTC. This process enables the application to write several registers consecutively, without any extra delay due to synchronization.

Due to synchronization, a read access always reads a value that is two to three system clocks (48 MHz) delayed. In this case, the system bus is not halted.

The AON_RTC:EVFLAGS register has a fast-clear feature. When written to 1, the MCU intermediately clears the EVFLAGS bit field. This process enables the MCU to clear the source quickly if the status is used as an interrupt or event. Due to synchronization, the actual flag in the RTC is not cleared until 1 or 2 clock cycles later. For this reason, a new event is masked for up to two 2 MHz MF periods.

18.3.2 Entering Sleep and Wakeup From Sleep

Before entering sleep, it must be ensured that all write requests to the AON registers are completed. This is done by the hardware in the MCU domain.

Upon wakeup from sleep, the application must wait for one 2 MHz MF period. This wait ensures that the MCU domain register interface is correctly synchronized. If registers are read before synchronization is completed, the value might not be updated. For example, reading the AON_RTC:SEC register might show the value from before entering sleep, and not the current value.

18.3.3 AON_RTC:SYNC Register

The AON_RTC:SYNC register synchronizes between the MCU domain and AON domain.

A read request from the AON_RTC:SYNC register does not return if there are outstanding write requests to the AON registers; in other words, the bus is halted until all outstanding requests are completed.

A write request triggers a dummy write to the AON domain. This write can ensure synchronization to the MF clock. This dummy write takes one to two 2 MHz MF clock cycles.

1. Write to the AON_RTC:SYNC register or any other register in the AON domain. The write triggers an outstanding write request to be registered on the AON domain.
2. Read from the AON_RTC:SYNC register. This read does not return until all outstanding requests are completed.

The AON_RTC:SYNC register operation is typically used when a specific order must be ensured. For example, when disabling a channel, the AON_RTC:SYNC register can be polled to ensure that the channel has been disabled and no further events can occur:

1. Set AON_RTC:CHCTL.CH2_EN = 0.
2. Read the AON_RTC:SYNC register.
3. The channel is now disabled. No further events can occur.

Another typical use is to ensure the correct values are updated in the MCU domain on wakeup. This MCU domain is only updated on a positive edge of the 2 MHz MF clock.

1. Write to the AON_RTC:SYNC register.
2. Read the AON_RTC:SYNC register.
3. Other AON_RTC registers can now be read safely as their information is correctly updated.

18.4 RTC Registers

18.4.1 AON_RTC Registers

Table 18-1 lists the memory-mapped registers for the AON_RTC registers. All register offset addresses not listed in Table 18-1 should be considered as reserved locations and the register contents should not be modified.

Table 18-1. AON_RTC Registers

Offset	Acronym	Register Name	Section
0h	CTL	Control	Section 18.4.1.1
4h	EVFLAGS	Event Flags, RTC Status	Section 18.4.1.2
8h	SEC	Second Counter Value, Integer Part	Section 18.4.1.3
Ch	SUBSEC	Second Counter Value, Fractional Part	Section 18.4.1.4
10h	SUBSECINC	Subseconds Increment	Section 18.4.1.5
14h	CHCTL	Channel Configuration	Section 18.4.1.6
18h	CH0CMP	Channel 0 Compare Value	Section 18.4.1.7
1Ch	CH1CMP	Channel 1 Compare Value	Section 18.4.1.8
20h	CH2CMP	Channel 2 Compare Value	Section 18.4.1.9
24h	CH2CMPINC	Channel 2 Compare Value Auto-increment	Section 18.4.1.10
28h	CH1CAPT	Channel 1 Capture Value	Section 18.4.1.11
2Ch	SYNC	AON Synchronization	Section 18.4.1.12
30h	TIME	Current Counter Value	Section 18.4.1.13
34h	SYNCLF	Synchronization to SCLK_LF	Section 18.4.1.14

Complex bit access types are encoded to fit into small table cells. Table 18-2 shows the codes that are used for access types in this section.

Table 18-2. AON_RTC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

18.4.1.1 CTL Register (Offset = 0h) [Reset = 00000000h]

CTL is shown in Table 18-3.

Return to the Summary Table.

Control
This register contains various bitfields for configuration of RTC
RTL Name = CONFIG

Table 18-3. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	COMB_EV_MASK	R/W	0h	Eventmask selecting which delayed events that form the combined event. 0h = No event is selected for combined event. 1h = Use Channel 0 delayed event in combined event 2h = Use Channel 1 delayed event in combined event 4h = Use Channel 2 delayed event in combined event
15-12	RESERVED	R	0h	Reserved
11-8	EV_DELAY	R/W	0h	Number of SCLK_LF clock cycles waited before generating delayed events. (Common setting for all RTC cannels) the delayed event is delayed 0h = No delay on delayed event 1h = Delay by 1 clock cycles 2h = Delay by 2 clock cycles 3h = Delay by 4 clock cycles 4h = Delay by 8 clock cycles 5h = Delay by 16 clock cycles 6h = Delay by 32 clock cycles 7h = Delay by 48 clock cycles 8h = Delay by 64 clock cycles 9h = Delay by 80 clock cycles Ah = Delay by 96 clock cycles Bh = Delay by 112 clock cycles Ch = Delay by 128 clock cycles Dh = Delay by 144 clock cycles
7	RESET	W1C	0h	RTC Counter reset. Writing 1 to this bit will reset the RTC counter. This bit is cleared when reset takes effect
6-3	RESERVED	R	0h	Reserved
2	RTC_4KHZ_EN	R/W	0h	RTC_4KHZ is a 4 KHz reference output, tapped from SUBSEC.VALUE bit 19 which is used by AUX timer. 0: RTC_4KHZ signal is forced to 0 1: RTC_4KHZ is enabled (provied that RTC is enabled EN)
1	RTC_UPD_EN	R/W	0h	RTC_UPD is a 16 KHz signal used to sync up the radio timer. The 16 Khz is SCLK_LF divided by 2 0: RTC_UPD signal is forced to 0 1: RTC_UPD signal is toggling @16 kHz
0	EN	R/W	0h	Enable RTC counter 0: Halted (frozen) 1: Running

18.4.1.2 EVFLAGS Register (Offset = 4h) [Reset = 00000000h]

EVFLAGS is shown in Table 18-4.

Return to the Summary Table.

Event Flags, RTC Status
This register contains event flags from the 3 RTC channels. Each flag will be cleared when writing a '1' to the corresponding bitfield.

Table 18-4. EVFLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	CH2	R/W1C	0h	Channel 2 event flag, set when CHCTL.CH2_EN = 1 and the RTC value matches or passes the CH2CMP value. An event will be scheduled to occur as soon as possible when writing to CH2CMP provided that the channel is enabled and the new value matches any time between next RTC value and 1 second in the past Writing 1 clears this flag. AUX_SCE can read the flag through AUX_EVCTL:EVSTAT2.AON_RTC_CH2 and clear it using AUX_SYSIF:RTCEVCLR.RTC_CH2_EV_CLR.
15-9	RESERVED	R	0h	Reserved
8	CH1	R/W1C	0h	Channel 1 event flag, set when CHCTL.CH1_EN = 1 and one of the following: - CHCTL.CH1_CAPT_EN = 0 and the RTC value matches or passes the CH1CMP value. - CHCTL.CH1_CAPT_EN = 1 and capture occurs. An event will be scheduled to occur as soon as possible when writing to CH1CMP provided that the channel is enabled, in compare mode and the new value matches any time between next RTC value and 1 second in the past. Writing 1 clears this flag.
7-1	RESERVED	R	0h	Reserved
0	CH0	R/W1C	0h	Channel 0 event flag, set when CHCTL.CH0_EN = 1 and the RTC value matches or passes the CH0CMP value. An event will be scheduled to occur as soon as possible when writing to CH0CMP provided that the channels is enabled and the new value matches any time between next RTC value and 1 second in the past. Writing 1 clears this flag.

18.4.1.3 SEC Register (Offset = 8h) [Reset = 00000000h]

SEC is shown in Table 18-5.

Return to the Summary Table.

Second Counter Value, Integer Part

Table 18-5. SEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	Unsigned integer representing Real Time Clock in seconds. When reading this register the content of SUBSEC.VALUE is simultaneously latched. A consistent reading of the combined Real Time Clock can be obtained by first reading this register, then reading SUBSEC register.

18.4.1.4 SUBSEC Register (Offset = Ch) [Reset = 00000000h]

SUBSEC is shown in Table 18-6.

Return to the Summary Table.

Second Counter Value, Fractional Part

Table 18-6. SUBSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	Unsigned integer representing Real Time Clock in fractions of a second (VALUE/2^{32} seconds) at the time when SEC register was read. Examples : - 0x0000_0000 = 0.0 sec - 0x4000_0000 = 0.25 sec - 0x8000_0000 = 0.5 sec - 0xC000_0000 = 0.75 sec

18.4.1.5 SUBSECINC Register (Offset = 10h) [Reset = 00800000h]

SUBSECINC is shown in Table 18-7.

Return to the Summary Table.

Subseconds Increment
Value added to SUBSEC.VALUE on every SCLK_LFclock cycle.

Table 18-7. SUBSECINC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VALUEINC	R	00800000h	This value compensates for a SCLK_LF clock which has an offset from 32768 Hz. The compensation value can be found as 2^{38} / freq, where freq is SCLK_LF clock frequency in Hertz This value is added to SUBSEC.VALUE on every cycle, and carry of this is added to SEC.VALUE. To perform the addition, bits [23:6] are aligned with SUBSEC.VALUE bits [17:0]. The remaining bits [5:0] are accumulated in a hidden 6-bit register that generates a carry into the above mentioned addition on overflow. The default value corresponds to incrementing by precisely 1/32768 of a second. NOTE: This register is read only. Modification of the register value must be done using registers AUX_SYSIF:RTCSUBSECINC0 , AUX_SYSIF:RTCSUBSECINC1 and AUX_SYSIF:RTCSUBSECINCCTL

18.4.1.6 CHCTL Register (Offset = 14h) [Reset = 00000000h]

CHCTL is shown in Table 18-8.

Return to the Summary Table.

Channel Configuration

Table 18-8. CHCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18	CH2_CONT_EN	R/W	0h	Set to enable continuous operation of Channel 2
17	RESERVED	R	0h	Reserved
16	CH2_EN	R/W	0h	RTC Channel 2 Enable 0: Disable RTC Channel 2 1: Enable RTC Channel 2
15-10	RESERVED	R	0h	Reserved
9	CH1_CAPT_EN	R/W	0h	Set Channel 1 mode 0: Compare mode (default) 1: Capture mode
8	CH1_EN	R/W	0h	RTC Channel 1 Enable 0: Disable RTC Channel 1 1: Enable RTC Channel 1
7-1	RESERVED	R	0h	Reserved
0	CH0_EN	R/W	0h	RTC Channel 0 Enable 0: Disable RTC Channel 0 1: Enable RTC Channel 0

18.4.1.7 CH0CMP Register (Offset = 18h) [Reset = 00000000h]

CH0CMP is shown in Table 18-9.

Return to the Summary Table.

Channel 0 Compare Value

Table 18-9. CH0CMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	RTC Channel 0 compare value. Bit 31 to 16 represents seconds and bits 15 to 0 represents subseconds of the compare value. The compare value is compared against SEC.VALUE (15:0) and SUBSEC.VALUE (31:16) values of the Real Time Clock register. A Cannel 0 event is generated when {SEC.VALUE(15:0),SUBSEC.VALUE (31:16)} is reaching or exciting the compare value. Writing to this register can trigger an immediate(1) event in case the new compare value matches a Real Time Clock value from 1 second in the past up till current Real Time Clock value. Example: To generate a compare 5.5 seconds RTC start,- set this value = 0x0005_8000 (1) It can take up to one SCLK_LF clock cycles before event occurs due to synchronization.

18.4.1.8 CH1CMP Register (Offset = 1Ch) [Reset = 00000000h]

CH1CMP is shown in Table 18-10.

Return to the Summary Table.

Channel 1 Compare Value

Table 18-10. CH1CMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	RTC Channel 1 compare value. Bit 31 to 16 represents seconds and bits 15 to 0 represents subseconds of the compare value. The compare value is compared against SEC.VALUE (15:0) and SUBSEC.VALUE (31:16) values of the Real Time Clock register. A Cannel 0 event is generated when {SEC.VALUE(15:0),SUBSEC.VALUE (31:16)} is reaching or exciting the compare value. Writing to this register can trigger an immediate(1) event in case the new compare value matches a Real Time Clock value from 1 second in the past up till current Real Time Clock value. Example: To generate a compare 5.5 seconds RTC start,- set this value = 0x0005_8000 (1) It can take up to one SCLK_LF clock cycles before event occurs due to synchronization.

18.4.1.9 CH2CMP Register (Offset = 20h) [Reset = 00000000h]

CH2CMP is shown in Table 18-11.

Return to the Summary Table.

Channel 2 Compare Value

Table 18-11. CH2CMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	RTC Channel 2 compare value. Bit 31 to 16 represents seconds and bits 15 to 0 represents subseconds of the compare value. The compare value is compared against SEC.VALUE (15:0) and SUBSEC.VALUE (31:16) values of the Real Time Clock register. A Cannel 0 event is generated when {SEC.VALUE(15:0),SUBSEC.VALUE (31:16)} is reaching or exciting the compare value. Writing to this register can trigger an immediate(1) event in case the new compare value matches a Real Time Clock value from 1 second in the past up till current Real Time Clock value. Example: To generate a compare 5.5 seconds RTC start,- set this value = 0x0005_8000 (1) It can take up to one SCLK_LF clock cycles before event occurs due to synchronization.

18.4.1.10 CH2CMPINC Register (Offset = 24h) [Reset = 00000000h]

CH2CMPINC is shown in Table 18-12.

Return to the Summary Table.

Channel 2 Compare Value Auto-increment
This register is primarily used to generate periodical wake-up for the AUX_SCE module, through the [AUX_EVCTL.EVSTAT0.AON_RTC] event.

Table 18-12. CH2CMPINC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VALUE	R/W	0h	If CHCTL.CH2_CONT_EN is set, this value is added to CH2CMP.VALUE on every channel 2 compare event.

18.4.1.11 CH1CAPT Register (Offset = 28h) [Reset = 00000000h]

CH1CAPT is shown in Table 18-13.

Return to the Summary Table.

Channel 1 Capture Value
If CHCTL.CH1_EN = 1and CHCTL.CH1_CAPT_EN = 1, capture occurs on each rising edge of the event selected in AON_EVENT:RTCSEL.

Table 18-13. CH1CAPT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SEC	R	0h	Value of SEC.VALUE bits 15:0 at capture time.
15-0	SUBSEC	R	0h	Value of SUBSEC.VALUE bits 31:16 at capture time.

18.4.1.12 SYNC Register (Offset = 2Ch) [Reset = 00000000h]

SYNC is shown in Table 18-14.

Return to the Summary Table.

AON Synchronization
This register is used for synchronizing between MCU and entire AON domain.

Table 18-14. SYNC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	WBUSY	R/W	0h	This register will always return 0,- however it will not return the value until there are no outstanding write requests between MCU and AON Note: Writing to this register prior to reading will force a wait until next SCLK_MF edge. This is recommended for syncing read registers from AON when waking up from sleep Failure to do so may result in reading AON values from prior to going to sleep

18.4.1.13 TIME Register (Offset = 30h) [Reset = 00000000h]

TIME is shown in Table 18-15.

Return to the Summary Table.

Current Counter Value

Table 18-15. TIME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	SEC_L	R	0h	Returns the lower halfword of SEC register.
15-0	SUBSEC_H	R	0h	Returns the upper halfword of SUBSEC register.

18.4.1.14 SYNCLF Register (Offset = 34h) [Reset = 00000000h]

SYNCLF is shown in Table 18-16.

Return to the Summary Table.

Synchronization to SCLK_LF
This register is used for synchronizing MCU to positive or negative edge of SCLK_LF.

Table 18-16. SYNCLF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	PHASE	R	0h	This bit will always return the SCLK_LF phase. The return will delayed until a positive or negative edge of SCLK_LF is seen. 0: Falling edge of SCLK_LF 1: Rising edge of SCLK_LF

Chapter 19

Watchdog Timer (WDT)

The watchdog timer (WDT) is used to regain control when the system has failed due to a software error or to the failure of an external device to respond in the expected way. The WDT can generate a non-maskable interrupt (NMI), a regular interrupt, or a reset when a time-out value is reached. In addition, the WDT can be configured to generate an interrupt to the microcontroller (MCU) on its first time-out and to generate a reset signal on its second time-out.

19.1 Introduction 1442
19.2 Functional Description 1442
19.3 Initialization and Configuration 1443
19.4 WDT Registers 1444

19.1 Introduction

WDT has the following features:

- 32-bit down counter with a programmable load register
- Programmable interrupt generation logic with interrupt masking and optional NMI function
- Lock register protection from runaway software
- Reset generation logic with an enable or disable
- User-enabled stalling when the microcontroller asserts the CPU Halt flag during debug

The WDT can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the WDT has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

There are two possible interrupts that can be driven out of the WDT. The interrupt choice is controlled using the WDT:CTL.INTTYPE register.

19.2 Functional Description

The WDT module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the WDT interrupt. Figure 19-1 shows the WDT block diagram.

The watchdog interrupt can be programmed to be a non-maskable interrupt (NMI) using the WDT:CTL.INTTYPE register. After the first time-out event, the 32-bit counter is reloaded with the value of the WDT Load register (WDT:LOAD), and the timer resumes counting down from that value. To prevent the WDT configuration from being inadvertently altered by software, the write access to the watchdog registers can be locked by writing the WDT:LOCK register to any value. To unlock the WDT, write the WDT:LOCK register to the value 0x1ACCE551.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the WDT:CTL.RESEN register to 1, the WDT asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the WDT:LOAD register, and counting resumes from that value.

If the WDT:LOAD register is written with a new value while the WDT counter is counting, then the counter is loaded with the new value and continues counting.

Writing to the WDT:LOAD register does not clear an active interrupt. An interrupt must be cleared by writing to the Watchdog Interrupt Clear register (WDT:ICR). The watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is enabled again, the 32-bit counter is preloaded with the load register value (not its last state).

Note

The watchdog causes a warm reset in the system. This warm reset can be blocked by ICEPick, which is useful for debugging. When ICEPick is asserted, the warm reset is blocked from the rest of the system; however, watchdog itself is reset.

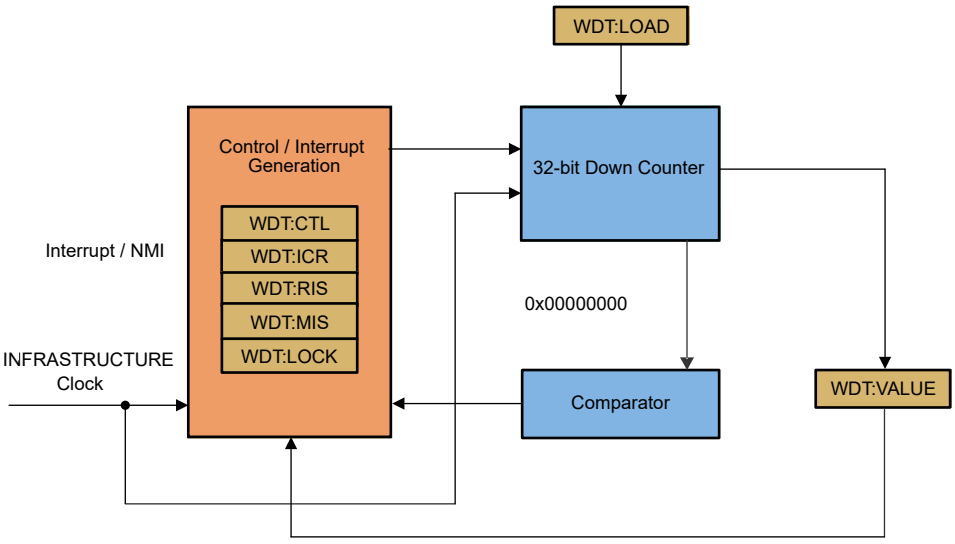


Figure 19-1. WDT Block Diagram

19.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled. The WDT is running off the INFRASTRUCTURE clock sourced by the MCU PRCM module. The WDT is then configured using the following sequence:

1. Load the WDT:LOAD register with the desired timer load value.
2. If the watchdog is configured to trigger system resets, set the WDT:CTL.RESEN bit.
3. Set the WDT:CTL.INTEN register bit to enable the WDT.
4. Lock the WDT module using the WDT:LOCK register.

19.4 WDT Registers

Table 19-1 lists the memory-mapped registers for the WDT registers. All register offset addresses not listed in Table 19-1 should be considered as reserved locations and the register contents should not be modified.

Table 19-1. WDT Registers

Offset	Acronym	Register Name	Section
0h	LOAD	Configuration	Section 19.4.1
4h	VALUE	Current Count Value	Section 19.4.2
8h	CTL	Control	Section 19.4.3
Ch	ICR	Interrupt Clear	Section 19.4.4
10h	RIS	Raw Interrupt Status	Section 19.4.5
14h	MIS	Masked Interrupt Status	Section 19.4.6
418h	TEST	Test Mode	Section 19.4.7
41Ch	INT_CAUS	Interrupt Cause Test Mode	Section 19.4.8
C00h	LOCK	Lock	Section 19.4.9

Complex bit access types are encoded to fit into small table cells. Table 19-2 shows the codes that are used for access types in this section.

Table 19-2. WDT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

19.4.1 LOAD Register (Offset = 0h) [Reset = FFFFFFFFh]

LOAD is shown in Table 19-3.

Return to the Summary Table.

Configuration

Table 19-3. LOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTLOAD	R/W	FFFFFFFFh	This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter is restarted to count down from the new value. If this register is loaded with 0x0000.0000, an interrupt is immediately generated.

19.4.2 VALUE Register (Offset = 4h) [Reset = FFFFFFFFh]

VALUE is shown in Table 19-4.

Return to the Summary Table.

Current Count Value

Table 19-4. VALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTVALUE	R	FFFFFFFFh	This register contains the current count value of the timer.

19.4.3 CTL Register (Offset = 8h) [Reset = 00000000h]

CTL is shown in Table 19-5.

Return to the Summary Table.

Control

Table 19-5. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	INTTYPE	R/W	0h	WDT Interrupt Type 0: WDT interrupt is a standard interrupt. 1: WDT interrupt is a non-maskable interrupt.
1	RESEN	R/W	0h	WDT Reset Enable. Defines the function of the WDT reset source (see PRCM:WARMRESET.WDT_STAT if enabled) 0: Disabled. 1: Enable the Watchdog reset output.
0	INTEN	R/W	0h	WDT Interrupt Enable 0: Interrupt event disabled. 1: Interrupt event enabled. Once set, this bit can only be cleared by a hardware reset.

19.4.4 ICR Register (Offset = Ch) [Reset = 00000000h]

ICR is shown in Table 19-6.

Return to the Summary Table.

Interrupt Clear

Table 19-6. ICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTICR	W	0h	This register is the interrupt clear register. A write of any value to this register clears the WDT interrupt and reloads the 32-bit counter from the LOAD register.

19.4.5 RIS Register (Offset = 10h) [Reset = 00000000h]

RIS is shown in Table 19-7.

Return to the Summary Table.

Raw Interrupt Status

Table 19-7. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	WDTRIS	R	0h	This register is the raw interrupt status register. WDT interrupt events can be monitored via this register if the controller interrupt is masked. Value Description 0: The WDT has not timed out 1: A WDT time-out event has occurred

19.4.6 MIS Register (Offset = 14h) [Reset = 00000000h]

MIS is shown in Table 19-8.

Return to the Summary Table.

Masked Interrupt Status

Table 19-8. MIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	WDTMIS	R	0h	This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the WDT interrupt enable bit CTL.INTEN. Value Description 0: The WDT has not timed out or is masked. 1: An unmasked WDT time-out event has occurred.

19.4.7 TEST Register (Offset = 418h) [Reset = 00000000h]

TEST is shown in Table 19-9.

Return to the Summary Table.

Test Mode

Table 19-9. TEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	STALL	R/W	0h	WDT Stall Enable 0: The WDT timer continues counting if the CPU is stopped with a debugger. 1: If the CPU is stopped with a debugger, the WDT stops counting. Once the CPU is restarted, the WDT resumes counting.
7-1	RESERVED	R	0h	Reserved
0	TEST_EN	R/W	0h	The test enable bit 0: Enable external reset 1: Disables the generation of an external reset. Instead bit 1 of the INT_CAUS register is set and an interrupt is generated

19.4.8 INT_CAUS Register (Offset = 41Ch) [Reset = 00000000h]

INT_CAUS is shown in Table 19-10.

Return to the Summary Table.

Interrupt Cause Test Mode

Table 19-10. INT_CAUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	CAUSE_RESET	R	0h	Indicates that the cause of an interrupt was a reset generated but blocked due to TEST.TEST_EN (only possible when TEST.TEST_EN is set).
0	CAUSE_INTR	R	0h	Replica of RIS.WDTRIS

19.4.9 LOCK Register (Offset = C00h) [Reset = 00000000h]

LOCK is shown in Table 19-11.

Return to the Summary Table.

Lock

Table 19-11. LOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDTLOCK	R/W	0h	WDT Lock: A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates (NOTE: TEST.TEST_EN bit is not lockable). A read of this register returns the following values: 0x0000.0000: Unlocked 0x0000.0001: Locked

This page intentionally left blank.

Chapter 20

AUX Domain Sensor Controller and Peripherals

This chapter describes the functionality of the AUX subsystem on the CC13x4x10 and CC26x4x10 device platform.

20.1 Introduction 1456
20.2 Power and Clock Management 1458
20.3 Sensor Controller 1461
20.4 Digital Peripheral Modules 1479
20.5 Analog Peripheral Modules 1505
20.6 Event Routing and Usage 1521
20.7 Sensor Controller Alias Register Space 1527
20.8 AUX Domain Sensor Controller and Peripherals Registers 1532

20.1 Introduction

The auxiliary (AUX) domain is an ultra-low power and independent coprocessor subsystem located inside the AON domain. The AUX domain holds a power-efficient, programmable 16-bit Sensor Controller Engine (SCE) that has access to the following:

- AUX SRAM: instruction and data memory for the Sensor Controller
  - 4 KB = 2048 Sensor Controller 16-bit instructions/data words
- Analog peripherals:
  - ADC: 12-bit analog-to-digital converter, sample rate maximum of 200 kHz
  - COMPA: continuous-time comparator
  - COMPB: low-power clocked comparator
  - ISRC: 0- to 20-µA current source
  - Reference DAC: digital-to-analog converter
- Digital peripherals:
  - TDC: high-precision time-to-digital converter
  - AUX Timer0 and Timer1: simple 16-bit synchronous timers
  - AUX Timer2: asynchronous multipurpose timer with four capture/compare channels
  - Hardware semaphores: used to share peripherals between the Sensor Controller and the System CPU
  - AUX I/O controllers
  - AUX SPI master: for power-efficient sensor polling
  - Math accelerator: including 40-bit accumulator (signed/unsigned addition, multiplication)
  - Pulse counter
- I/O pins:
  - Up to 32 digital-only, general-purpose I/O pins
  - Up to 8 analog-capable, general-purpose I/O pins
- System oscillator control
- Multiple clocking and power options

The Sensor Controller does not have access to the MCU domain peripherals, RAM, Flash, or registers. This separation allows the MCU domain to enter and exit standby mode independently of the Sensor Controller. However, the System CPU and µDMA have access to all AUX domain peripherals at all times and may use peripherals that are not used by the Sensor Controller.

Access speed from the MCU domain is independent of the selected AUX clock, and Sensor Controller operation is independent of the MCU domain. The Sensor Controller has limited access to certain AON domain peripherals, such as the Real-Time Clock and the Battery Monitor and Temperature Sensor. The Sensor Controller can also request recharge of VDDR and monitor such events.

The System CPU can directly use the peripherals in the AUX domain. By leveraging the Sensor Controller however, simple background tasks can be run independently of the System CPU to achieve minimal system power consumption. Sensor Controller tasks can be set up to run periodically or run based on asynchronous events, such as GPIO events. Some examples of tasks fit for the Sensor Controller are as follows:

- Analog sensor polling, using the ADC or comparator
- Digital sensor polling, using SPI or bit-banged I^2C or other standards
- Capacitive sensing, using the integrated current source, comparator, and time-to-digital converter
- I/O waveform generation
  - General I/O control using the Sensor Controller
  - PWM using AUX Timer2
- Signal period and pulse width measurements, using TDC or AUX Timer2
- Task-dependent wakeup of the System CPU: Data is collected, optionally filtered, and ready for System CPU processing

20.1.1 AUX Block Diagram

Figure 20-1 shows connections for the modules in the AUX domain.

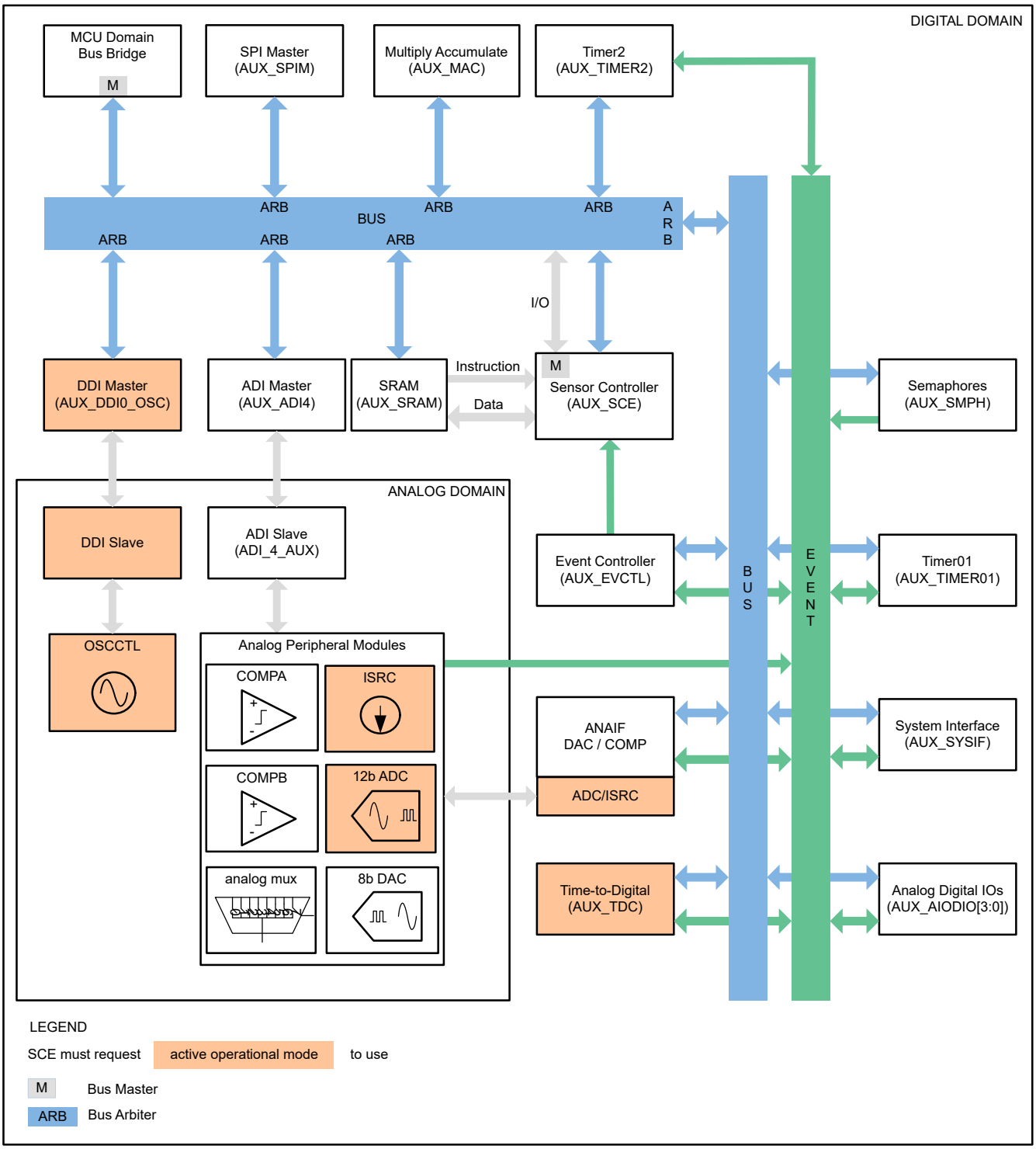


Figure 20-1. AUX Domain Block Diagram

Figure 20-1 shows that there are two bus masters in the AUX domain, the Sensor Controller and MCU domain Bus Bridge. The System CPU and the µDMA access the AUX domain over the bus bridge. Bus arbitration between the Sensor Controller and the MCU domain Bus Bridge happens upon access of modules that are

connected to the same bus arbiter. The bus arbiter always prioritizes the Sensor Controller. Hence, the System CPU can access the oscillator interface while the Sensor Controller accesses the SPI master without interfering with the other.

20.2 Power and Clock Management

The AUX power and clock management is mostly independent of the MCU domain. An active MCU domain state has the potential to:

- Increase the AUX bus clock speed to minimize access time.
- Cause small clock jitter on the clock selected by AON_PMCTL:AUXSCECLK that clocks the Sensor Controller and selected peripherals.

These topics are described in Section 20.2.1 through Section 20.2.4. The MCU domain is active when it receives SCLK_HF, which occurs in active and Idle (two of the power modes in TI's Power Manager).

20.2.1 Operational Modes

The AUX domain requests to operate in active, low-power, or power-down modes (see Figure 20-2).

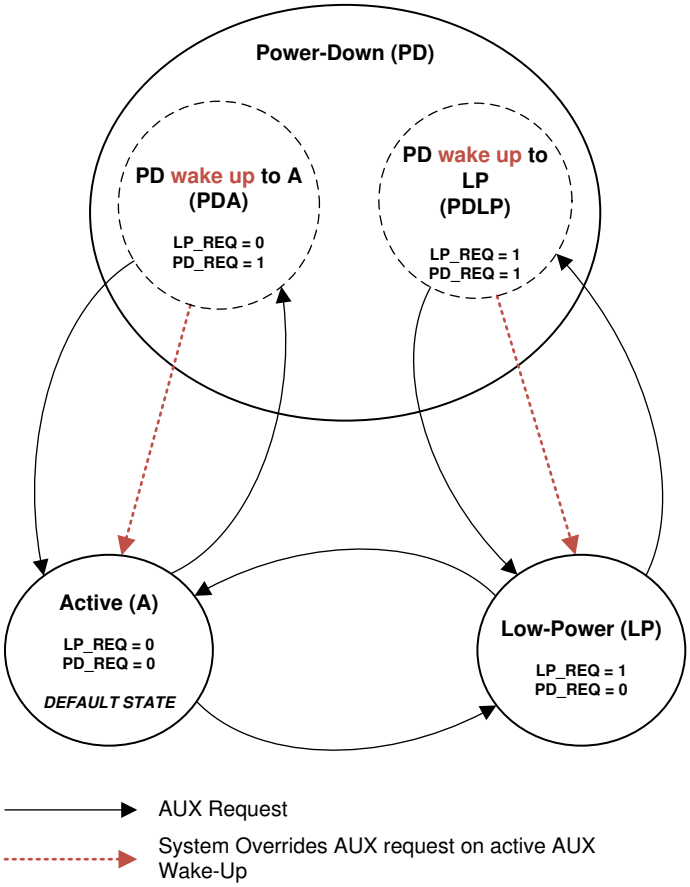


Figure 20-2. AUX State Diagram

Each operational mode is characterized by:

- System power supply request
- Sensor Controller operational clock rate (SCE clock rate)
- Sensor Controller module availability
- System response to an active AUX wake-up event

  The system does not change the AUX operational mode request. As shown in Figure 20-2, the system overrides the request as long as the AUX wake-up event is active. The AUX wakeup applies only to the Sensor Controller (for a description, see Section 20.1).

Table 20-1 summarizes the properties for each operational mode.

Table 20-1. AUX Operational Modes

Operational Mode		Active		Low-Power	Power-Down	
AUX SCE Clock Source		AON_PMCTL:AUXSCECLK.SRC		SCLK_MF	AON_PMCTL:AUXSCECLK.PDSRC	
SCE Clock Rate		24 MHz	2 MHz	2 MHz	32.768 kHz	NO_CLOCK
AUX Bus Clock Rate	MCU Inactive	24 MHz	24 MHz	2 MHz	32.768 kHz	NO_CLOCK
	MCU Active	24 MHz	24 MHz	24 MHz	24 MHz	24 MHz
Wake-Up System Response		None		None	Active or Low-Power	
Power Supply Request		DCDC or GLDO		µLDO	µLDO	
SCE Module Availability		All		Do not use ADC, ISRC, TDC, or OSCCTL.		

Follow the steps described in AUX_SYSIF:OPMODEREQ to request an operational mode (see Section 20.8.9). For AUX wakeup, see Section 20.1. An Active operational mode request prevents the system from entering standby.

Note

An active operational mode request prevents the system from entering standby or shutdown because the power controller requests a high-frequency clock source.

20.2.1.1 Dual-Rate AUX Clock

The AUX module operates internally at either the Sensor Controller Engine (SCE) clock rate or the bus clock rate. The Sensor Controller accesses the AUX modules at SCE clock rate, while the System CPU and µDMA access always happens at bus clock rate. As listed in Table 20-1, the AUX bus clock rate can be higher than the SCE clock rate under two conditions:

- SCE clock rate is 2 MHz in Active operational mode
- MCU domain is active

In both cases, the bus clock rate equals 24 MHz. Table 20-2 lists the operational clock rates for the various AUX modules.

To ensure correct timing, the modules and clocked resources used by the Sensor Controller must operate at SCE clock rate if possible, as the bus clock rate can change when the MCU power mode changes. This is especially important for the SPIM and Timer01 peripherals.

There are two side effects when the bus clock rate is higher than the SCE clock rate:

- Sensor Controller access to the DDI slave, ADI slave, and Timer2 completes faster when the bus clock rate becomes higher than the SCE clock rate.
- Certain Sensor Controller instructions like *iobset* and *iobclr* become non-atomic. The Sensor Controller and the System CPU or µDMA must implement a concept of resource or module ownership to avoid inconsistency.

Table 20-2. AUX Operational Clock Rates

Module	Operational Clock Rate(1)		
	AUX Bus Rate	AUX SCE Rate	AUX Timer2 Rate
ALIAS SPACE		x	
AUX_SPIM	(X)	X	
AUX_MAC	(X)	X	
AUX_TIMER2			X
AUX_TDC	X		
AUX_EVCTL	X	(XX)	
AUX_SYSIF	X		
AUX_TIMER01	(X)	X	
AUX_SMPH	X		
AUX_ANAIF:ADC	X		
AUX_ANAIF:DAC	(X)	X	
AUX_DDI0_OSC	X		
AUX_ADI4	X		
AUX_AIODIO0		X	
AUX_AIODIO1		X	
AUX_AIODIO2		X	
AUX_AIODIO3		X	
AUX_RAM	X		
AUX_SCE		X	

(1) X = Default rate
(X) = Rate override capability
(XX) = Rate override capability for event synchronization

20.2.2 Use Scenarios

20.2.2.1 MCU

When the MCU domain is active, all AUX modules are both accessible with minimum latency and functional to an MCU master, independent of the AUX clock and power management. Hence, the System CPU application has no need to configure the AUX operational mode to use a peripheral. The System CPU application cannot use an AUX peripheral in Standby mode (power mode defined in TI's Power Manager), but the AUX peripheral can create a wakeup event to the System CPU from this power mode. The Sensor Controller Engine can however run when the rest of the system is in Standby, and can thus be used to minimize current consumption.

TI's Power Manager requests the Power-Down operational mode (see Figure 20-2), by default.

20.2.2.2 Sensor Controller

The Sensor Controller Studio framework ensures that the Sensor Controller actively switches between different operational modes to reach the lowest possible task current consumption.

The Sensor Controller shall only execute code in Low-Power and Power-Down operational modes if AON_PMCTL:RECHARGECFG.MODE = COMPARATOR. Failure to follow this requirement can result in brownout.

20.2.3 SCE Clock Emulation

The SCE clock source gets emulated when it equals SCLK_MF or SCLK_LF and the MCU domain is active. This is done to ensure that access time to peripherals and resources in AUX from System CPU does not scale with the AUX clock rate. Clock emulation results in SCE clock period jitter:

- SCLK_MF clock emulation: The instant SCE clock period jitter equals ±2 SCLK_HF periods.
- SCLK_LF clock emulation: The instant SCE clock period jitter is 2 SCLK_HF periods. A single SCE clock period increases or decreases by 6 to 8 SCLK_HF periods when emulation starts or ends.

Clock emulation has the following implications:

- The small time uncertainty that clock emulation introduces in the clock period affects only modules that operate at the SCE clock rate.
- Frequency configuration of the SPI SCLK may require guard against shorter base clock period to meet I/O timing requirements. The impacted use cases have SCLK frequency of 1 MHz or less, which is rare.
- The SCE clock period jitter is not accumulative.

20.2.4 AUX RAM Retention

The AUX RAM can be powered off with or without retention. The AUX clock is inactive during power sequencing of AUX RAM; hence, stalling all access to AUX until the RAM power sequence is complete.

AON_PMCTL:RAMCFG.AUX_SRAM_RET_EN controls RAM retention, while AON_PMCTL:RAMCFG.AUX_SRAM_PWR_OFF controls RAM power.

20.3 Sensor Controller

20.3.1 Sensor Controller Studio

TI provides a development tool for the Sensor Controller called Sensor Controller Studio. This section gives an overview of the Sensor Controller programming model, Sensor Controller task development, and task testing, debugging and run-time logging. Full documentation on how to use the Sensor Controller is documented in the Sensor Controller Studio itself, as explained in Section 20.3.1.4.

The programming model and firmware framework are optimized for low overhead when communicating with the System CPU application, low AUX RAM memory footprint, and efficient use of the Sensor Controller's instruction set. To minimize power consumption, the Sensor Controller requests/enters power down when it idles to minimize power, see Section 20.2.1 for details.

20.3.1.1 Programming Model

Sensor Controller Studio is project based. Each Sensor Controller project may contain up to eight Sensor Controller tasks. The output of a project is a set of C source files called the Sensor Controller Interface (SCIF) driver, which can easily be integrated into the System CPU application.

The Sensor Controller is user programmable. The Sensor Controller task code programming language uses a syntax that is similar to C, but has limited features compared to C since it specifically targets the Sensor Controller Engine’s instruction set and firmware framework. For example, there is only support for 16-bit variables, and there are predefined roles for each block of task code (initialization, scheduled execution, event handling and termination).

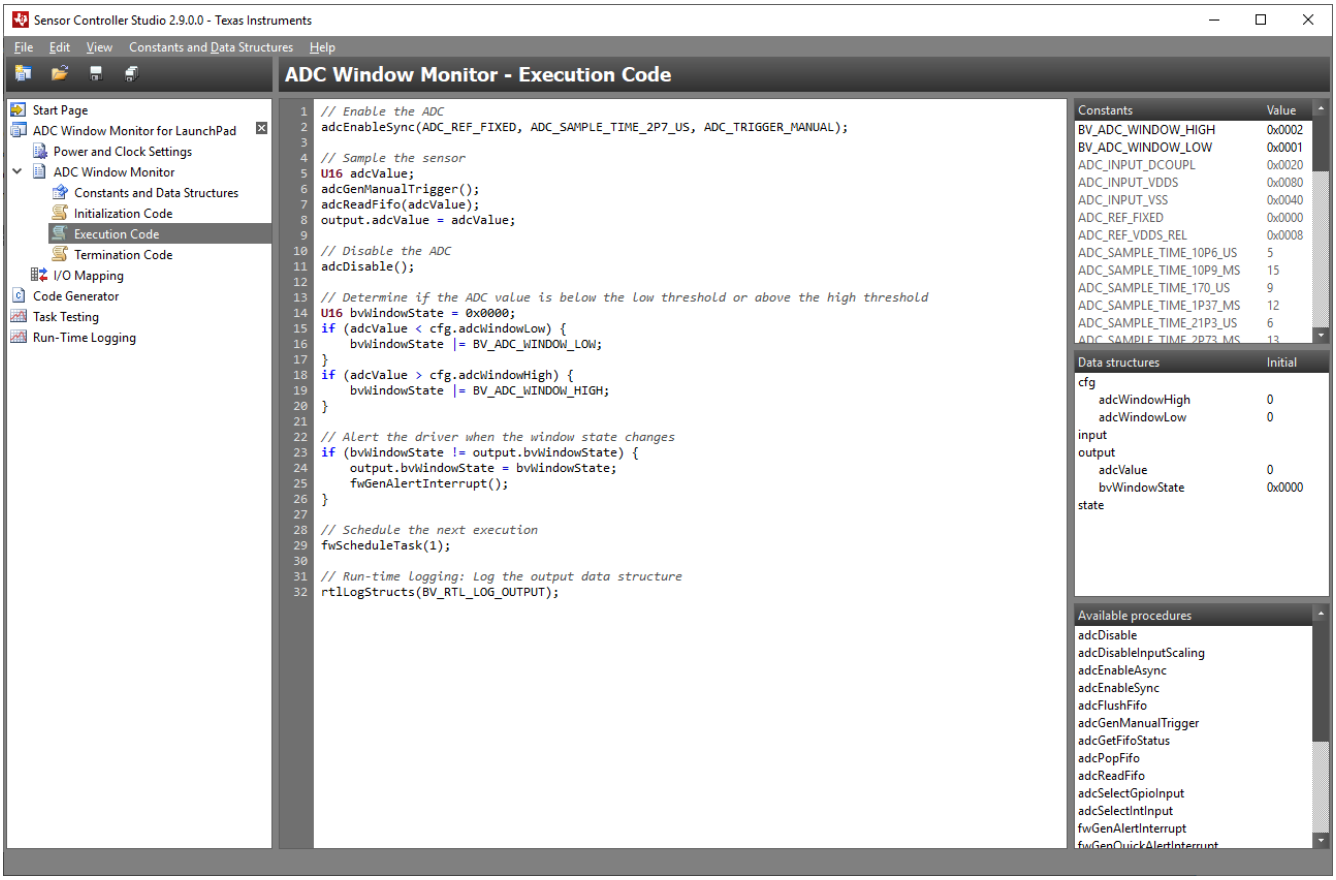


Figure 20-3. ADC Window Monitor - Execution Code

Hardware peripherals and software algorithms are available as resources. For each task, a set of resources must be selected and configured to enable different types of functionality, such as:

- Analog and digital peripheral modules (for example ADC)
- Analog and digital general-purpose I/O pins
- Simple and complex software algorithms
- Bit-banged serial interfaces
- Task scheduling and event handling
- Communication with the System CPU

Each task resource enables a set of procedures (equivalent to functions in C), with associated constants and global variables. The Sensor Controller calls these procedures from the task code, and can thereby combine use of multiple resources, and can also reuse resources (for example the ADC, Figure 20-4) across multiple tasks.

The Sensor Controller tasks and the System CPU application exchange data through data structure variables (located in the AUX RAM). Sensor Controller Studio generates type definitions and other needed definitions for easy access to these variables.

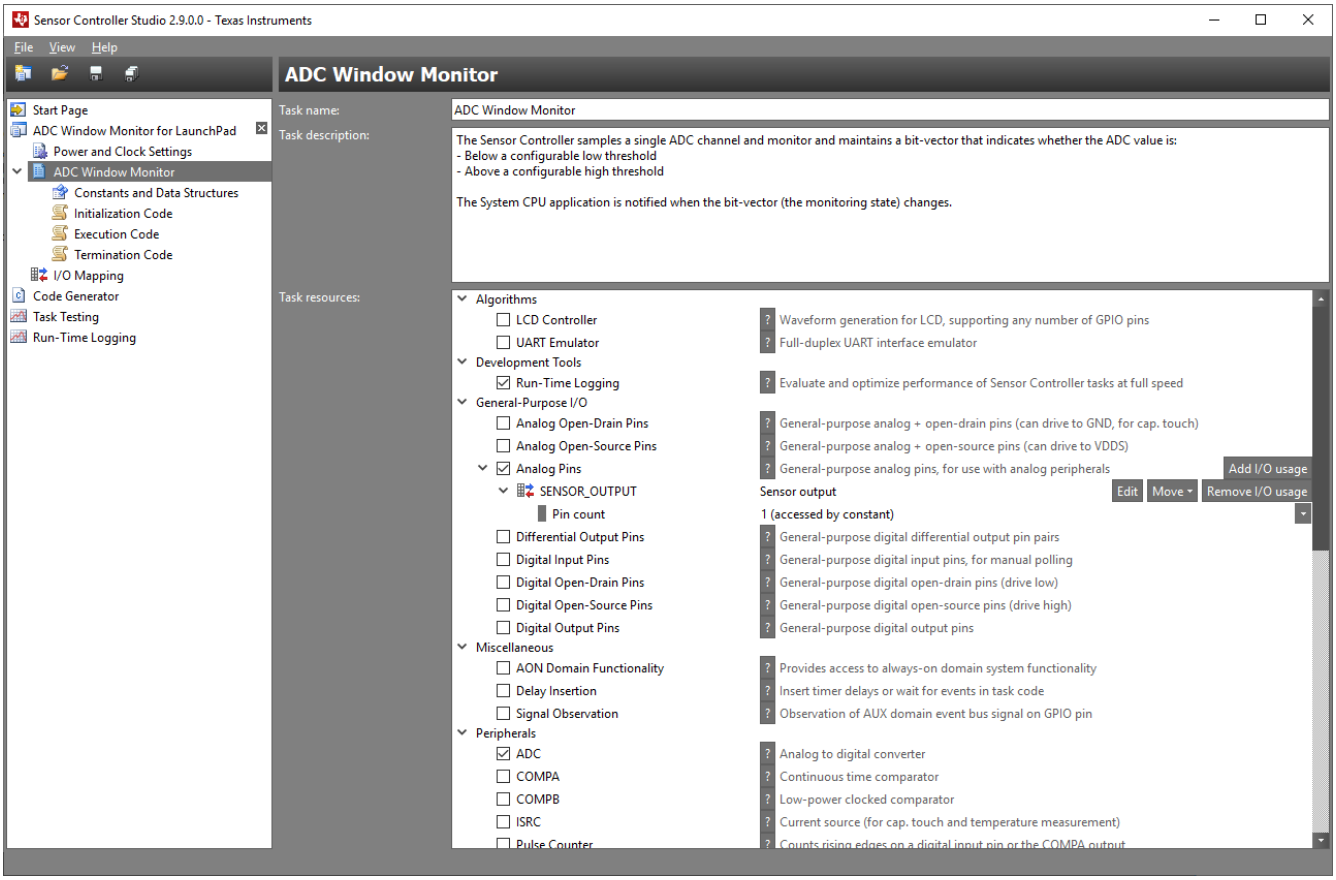


Figure 20-4. ADC Window Monitor

20.3.1.2 Task Development

A Sensor Controller project consists mainly of:

- Name and description, and other project properties
- Target chip selection and other static configuration of the Sensor Controller
- For each task:
  - Name and description
  - Resource selection and configuration
  - Resource-defined and user-defined constants
  - Resource-defined and user-defined data structures variables (located in the AUX RAM)
  - Sensor Controller task code:
    - Initialization code: Runs when the System CPU application starts the task
    - Execution code: Can be triggered regularly by AON_RTC, or by software
    - Event handler code: Can be triggered by various peripheral events
    - Termination code: Runs when the System CPU application stops the task
- I/O pin mapping

The generated SCIF driver consists of three parts:

- A generic application programming interface
- A driver setup, with all project specific definitions and data, including the AUX RAM image, code for I/O pin configuration, resource specific functions for use by the System CPU application, and so on
- An operating system abstraction layer that allows the driver to be used seamlessly with both TI-RTOS, nortos, and the TI Driver Porting Layer (DPL)

20.3.1.3 Task Testing, Task Debugging and Run-Time Logging

Sensor Controller Studio provides tools for testing, debugging, evaluating and tuning Sensor Controller tasks. Common for these tools is that they run the task code on the actual Sensor Controller on the target chip, and provide generic support for displaying the data structure variables in real-time. The data structure values can also be saved to file for external analysis.

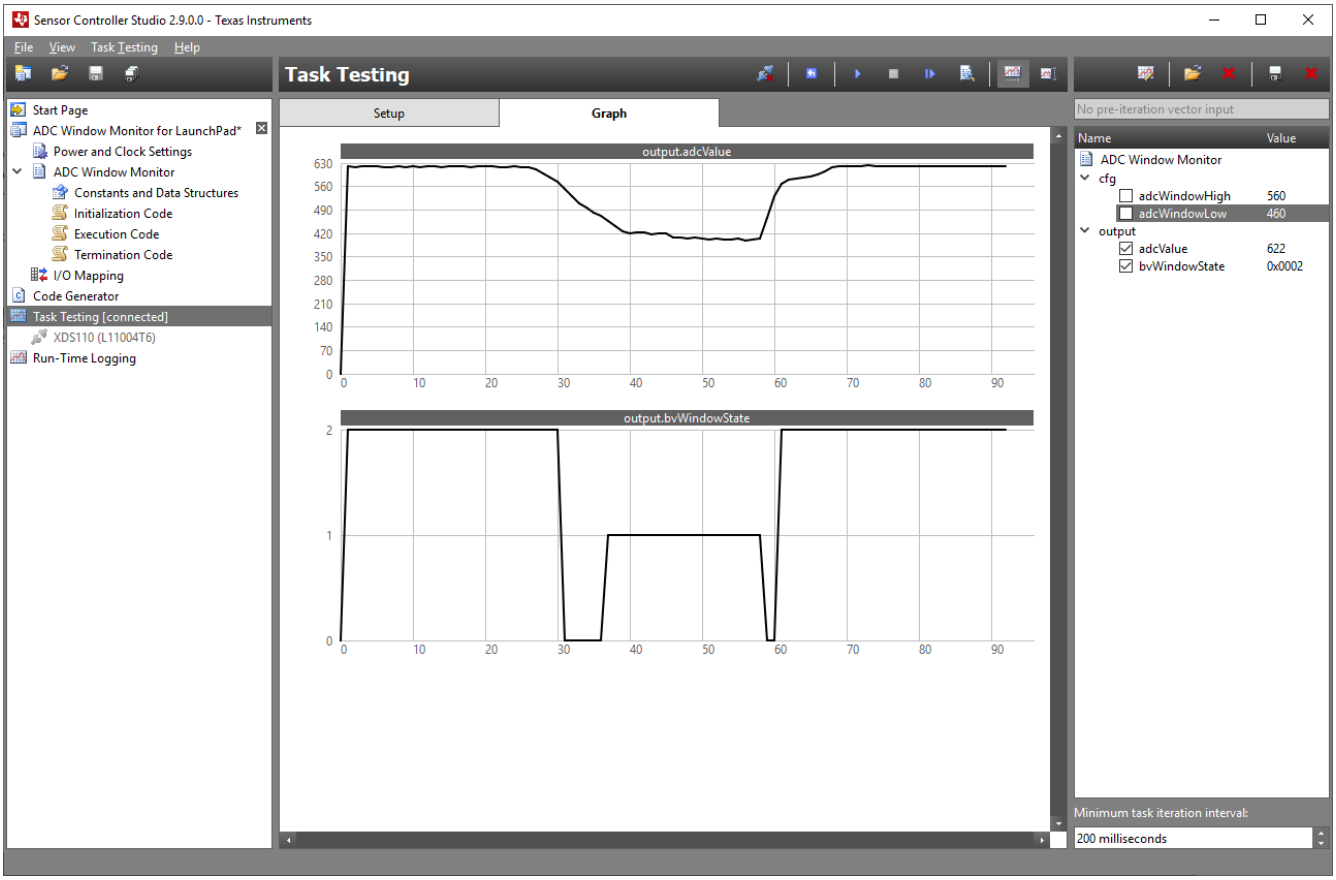


Figure 20-5. Task Testing

Table 20-3 summarizes and compares the most important properties of the available tools.

Table 20-3. Task Testing and Task Debugging

	Task Testing and Task Debugging	Run-Time Logging
Purpose	Evaluation, systematic testing and debugging. Selected task iterations can be debugged at instruction level.	Performance evaluation and optimization.
Execution speed	All code within a task code block runs at full speed. Timing between task code blocks cannot be controlled precisely, and depends on communication with PC.	Timing matches actual application.
Preparations	Specify a sequence of actions for each task iteration.	Enable the Run-Time Logging resource and use the associated procedures in task code to control logging.
Communication interface(s)	JTAG	UART + JTAG, UART only, or TCP/IP
Task count supported	One task at a time.	One or more tasks within one project.
Data structure logging	All data structures are logged. No overflow.	Selected data structures are logged. Overflow can occur, data discarded silently.

Table 20-3. Task Testing and Task Debugging (continued)

	Task Testing and Task Debugging	Run-Time Logging
Data structure editing	All data structure values can be modified, but only while the target is stopped. New data structure member values can be loaded from file and be applied between task iterations.	Data structures that are not logged can be modified at any time.

20.3.1.4 Documentation

Sensor Controller Studio comes with a help viewer (Figure 20-6) that provides all relevant documentation for Sensor Controller development. This documentation includes:

- An introduction to the Sensor Controller concept
- Detailed Sensor Controller Studio tool documentation
- Resource and procedure documentation
- Task code programming language reference and other reference documentation
- Detailed revision history

The generated SCIF driver is documented using Doxygen syntax. This documentation includes project and task specific information.

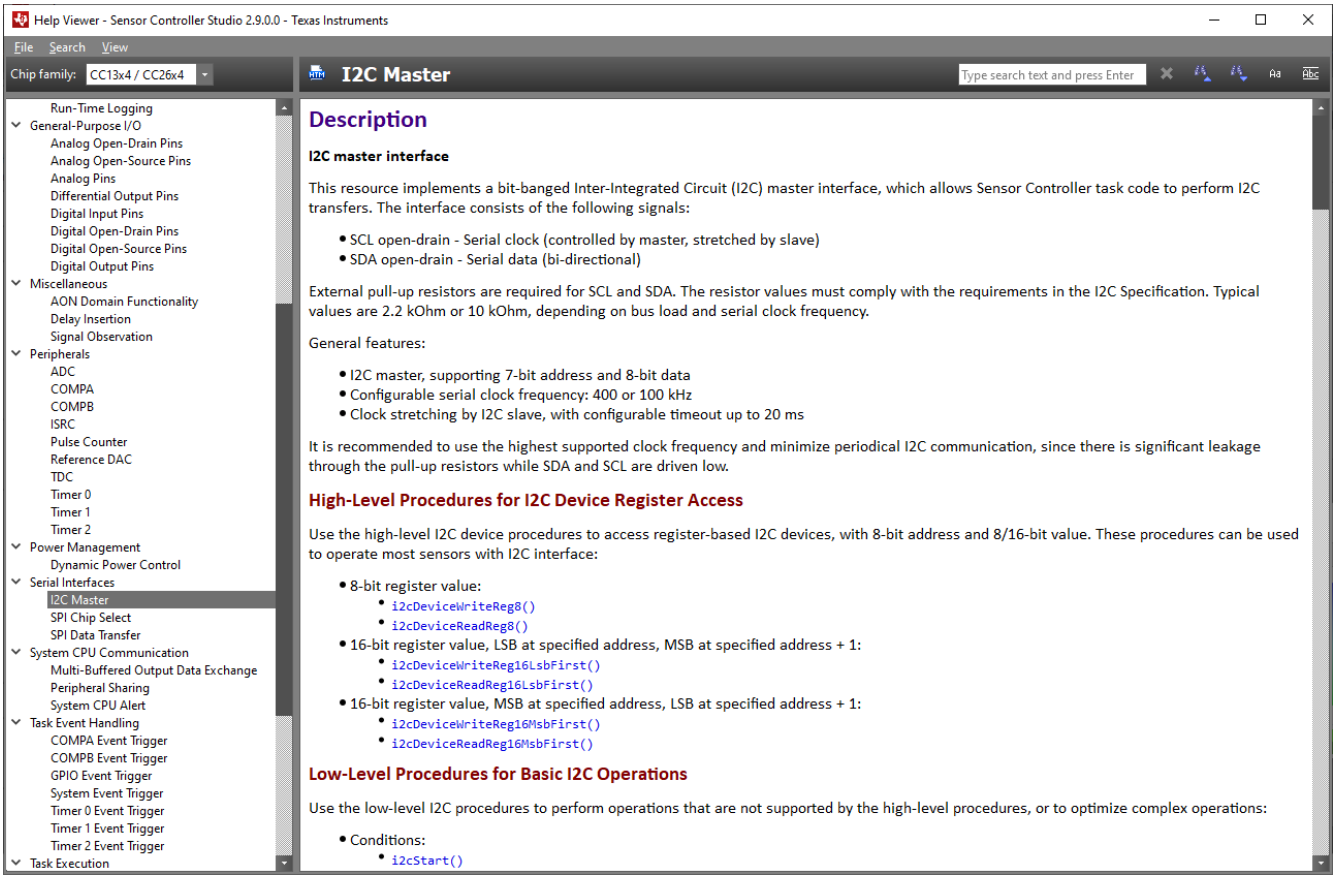


Figure 20-6. Help Viewer

20.3.2 Sensor Controller Engine (SCE)

This section provides an overview of the Sensor Controller Engine CPU core. For most users, this information can provide an understanding of the Sensor Controller Engine's capabilities and limitations, and also be used as a reference when debugging Sensor Controller tasks.

There is normally no need to write any assembly code for the Sensor Controller. It is possible to create your own task resources and procedures for the Sensor Controller Studio if needed, for example, when:

- There is a need to access hardware registers or register fields that are not supported by the provided procedures.
- There is a need for extremely optimized code, using code constructions that cannot be generated by the Sensor Controller Studio compiler.

20.3.2.1 Registers

There are eight 16-bit general-purpose registers: R0 through R7. A few instructions require dedicated use of R0 and R1.

There are four status flags: zero (Z), negative (N), carry (C), and overflow (V).

There is a 16-bit program counter, PC. There is also a dedicated stack for storing the program counter, which allows for three levels of subroutine calls. The Sensor Controller firmware framework uses one of these levels, which means that procedures can use two levels.

Pipeline Hazards

Due to internal pipelining and the desire to minimize register bypass logic, the instructions with dedicated use of R0 require some attention. R0 must not be loaded from memory or I/O by an instruction immediately preceding any instruction using R0 as a dedicated register. The affected instructions are:

```
ld Rd, [Rs+R0]
```

```
st Rd, [Rs+R0]
```

```
jmp R0
```

```
jsr R0
```

Violating this rule will cause the previous value of R0 to be used instead of the newly loaded value.

20.3.2.2 Memory Architecture

Memory Access to Instructions and Data

Data and instruction memory addressing is 16-bit oriented (an address increment of 1 corresponds to 2 bytes). Data words and instructions are mapped to the same memory address space. On the CC13x4x10 and CC26x4x10 devices, the AUX RAM is 4 KB with an address range from 0x0000 through 0x07FF.

Data memory is accessed through load and store instructions, supporting direct, register indirect, register indirect with post-increment, and register indexed modes. The only supported access size is 16 bit.

There is no access to memory other than the AUX RAM.

Information on System CPU access to the AUX RAM:

- The Sensor Controller has priority.
- Each 32-bit access gets divided into two 16-bit accesses at the AUX RAM interface. Hence, a 32-bit access is slower than a 16-bit or 8-bit access.
- Access to AUX RAM is significantly slower than access to System RAM, and hence it makes sense to minimize data transfer between the Sensor Controller and the System CPU application.

I/O Access to Module Registers

The Sensor Controller can access the AUX domain module registers through the I/O address space. I/O addressing is 8-bit oriented (an address increment of 1 corresponds to 1 byte), except for the register aliasing space, that is addresses 0 to 255, where each entry maps to a 16-bit register.

I/O registers are accessed through input and output instructions, supporting:
- Direct 1-bit accesses for 8 LSBs of a 16-bit register
- Direct or register indirect 16-bit accesses

Unaligned 16-bit accesses are not supported.

There is no access to registers outside of the AUX domain. Also, there is no access to the AUX RAM and AUX_SCE module through the I/O address space.

As described in Section 2.4.7.3, the boot configuration of PRCM:BUSSECCFG.BUS_CFG decides interconnect and firewall configuration of the device. When either contingency 1 (0xF3) or Non-secure (0xF9) modes are specified, there are no additional restrictions to which I/O registers SCE can access. When any other mode/value is specified, certain I/O register accesses will be blocked by the bus fabric with the additional consequence that SCE gets suspended:

- AUX_SYSIF:RTCSUBSECINC0
- AUX_SYSIF:RTCSUBSECINC1
- AUX_SYSIF:RTCSUBSECINCTL

Access to AUX_DDI0_OSC is also restricted in this case, with permissions for write and read of DDI half-words configured in AUX_SCE:NONSECDDIACC0-3.

20.3.2.3 Program Flow

General program flow is controlled by explicit use of jump and branch, conditional branch, subroutine call and return instructions. Execution can also be halted temporarily by using the `wev0` or `wev1` instructions to wait for the specified event before resuming execution. Preemption in any case is not supported.

The Sensor Controller enters standby mode by executing the `sleep` instruction. From standby mode, it can then wake up and start execution from one of eight event vectors, which are configured through AUX_SYSIF.

Zero-Overhead Loop

There is support for one level of zero-overhead loops using the `loop` instructions. This stores the addresses of the first and last instruction of the loop, and starts the loop counter.

20.3.2.4 Instruction Set

20.3.2.4.1 Instruction Timing

The Sensor Controller runs at 24 MHz in active mode and at 2 MHz in low-power mode. All Sensor Controller instructions use 2 clock cycles, with the following exceptions:
- I/O accesses (`in` , `out` , `iobset` , `iobclr` , and `iobtst`) to ADI and DDI registers, which go through multi-cycle interfaces. These registers are related to analog peripherals and oscillators.
- Power management instructions (`wev0` , `wev1` , and `sleep`), which are used to wait for event signals.

The Sensor Controller has priority over the System CPU and µDMA when accessing memory and module registers. ADI and DDI accesses can be delayed due to ongoing System CPU accesses.

20.3.2.4.2 Instruction Prefix

Some instructions with an 8-bit or 10-bit immediate value may be extended to a 16-bit immediate value by executing a prefix instruction, `pfix` , immediately before it. The prefix is valid for the next instruction only.

The last column in the following tables indicates which immediate value may be extended, if any.

The Sensor Controller Studio assembler adds prefix instructions automatically as needed, and detects immediate values that are invalid or out of range. The assembler also repairs branch instructions where the target address is out of range.

20.3.2.4.3 Instructions

Table 20-4. Memory Word Access

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
ld Rd, [#addr]	Load direct	Rd = mem[addr]	—	—	—	—	0ddd.10aa.aaaa.aaaa	addr
ld Rd, [Rs]	Load indirect	Rd = mem[Rs]	—	—	—	—	1ddd.1111.0000.1sss	
ld Rd, [Rs++]	Load indirect, post-increment	Rd = mem[Rs], Rs++	—	—	—	—	1ddd.1111.0001.0sss	
ld Rd, [Rs+R0]	Load indexed	Rd = mem[Rs+R0]	—	—	—	—	1ddd.1111.0001.1sss	
st Rd, [#addr]	Store direct	mem[addr] = Rd	—	—	—	—	0ddd.11aa.aaaa.aaaa	addr
st Rd, [Rs]	Store indirect	mem[Rs] = Rd	—	—	—	—	1ddd.1111.0010.1sss	
st Rd, [Rs++]	Store indirect, post-increment	mem[Rs] = Rd, Rs++	—	—	—	—	1ddd.1111.0011.0sss	
st Rd, [Rs+R0]	Store indexed	mem[Rs+R0] = Rd	—	—	—	—	1ddd.1111.0011.1sss	

Table 20-5. I/O Word Access

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
in Rd, [#addr]	Input direct	Rd = reg[addr]	—	—	—	—	1ddd.1001.aaaa.aaaa	addr
in Rd, [Rs]	Input indirect	Rd = reg[Rs]	—	—	—	—	1ddd.1111.0000.0sss	
out Rd, [#addr]	Output direct	reg[addr] = Rd	—	—	—	—	1ddd.1011.aaaa.aaaa	addr
out Rd, [Rs]	Output indirect	reg[Rs] = Rd	—	—	—	—	1ddd.1111.0010.0sss	

Table 20-6. I/O Bit Access

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
iobclr #imm, [#addr]	I/O bit clear direct	reg[addr] &= ~2^imm	—	—	—	—	010i.01ii.aaaa.aaaa	addr
iobset #imm, [#addr]	I/O bit set direct	reg[addr] |= 2^imm	—	—	—	—	011i.01ii.aaaa.aaaa	addr
iobtst #imm, [#addr]	I/O bit test direct	reg[addr] & 2^imm	—	—	√	—	001i.01ii.aaaa.aaaa	addr

Table 20-7. Register Access

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
ld Rd, #simm	Load immediate	Rd = simm	—	—	—	—	0ddd.00ii.iiii.iiii	simm
ld Rd, Rs	Load register	Rd = Rs	—	—	—	—	1ddd.1101.0100.0sss	

Table 20-8. Logical Operations

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
and Rd, #imm	AND immediate	Rd &= imm	x	x	0	0	1ddd.0000.iiii.iiii	imm
or Rd, #imm	OR immediate	Rd |= imm	x	x	0	0	1ddd.0010.iiii.iiii	imm
xor Rd, #imm	XOR immediate	Rd ^= imm	x	x	0	0	1ddd.0100.iiii.iiii	imm
tst Rd, #imm	Test immediate	Rd & imm	x	x	0	0	1ddd.1100.iiii.iiii	imm
and Rd, Rs	AND register	Rd &= Rs	x	x	0	0	1ddd.1101.0000.0sss	
or Rd, Rs	OR register	Rd |= Rs	x	x	0	0	1ddd.1101.0000.1sss	

Table 20-8. Logical Operations (continued)

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
xor Rd, Rs	XOR register	Rd ^= Rs	x	x	0	0	1ddd.1101.0001.0sss	
tst Rd, Rs	Test register	Rd & Rs	x	x	0	0	1ddd.1101.0011.0sss	
inv Rd	Invert register	Rd = ~Rd	x	x	0	0	1ddd.1101.1001.0010	

Table 20-9. Arithmetic Operations

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
add Rd, #simm	Add immediate	Rd += simm	x	x	x	x	1ddd.1000.iiii.iiii	simm
cmp Rd, #simm	Compare immediate	Rd - simm	x	x	x	x	1ddd.1010.iiii.iiii	simm
sub Rd, Rs	Subtract register	Rd -= Rs	x	x	x	x	1ddd.1101.0001.1sss	
add Rd, Rs	Add register	Rd += Rs	x	x	x	x	1ddd.1101.0010.0sss	
cmp Rd, Rs	Compare register	Rd - Rs	x	x	x	x	1ddd.1101.0010.1sss	
subr Rd, Rs	Subtract reverse register	Rd = Rs - Rd	x	x	x	x	1ddd.1101.0011.1sss	
abs Rd	Absolute register	Rd = (Rd >= 0) ? Rd : -Rd	x	x	x	x	1ddd.1101.1001.0000	
neg Rd	Negate register	Rd = -Rd	x	x	x	x	1ddd.1101.1001.0001	

Table 20-10. Shift Operations

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
lsl Rd, Rs	Logical shift left register	Rd <<= Rs	x	x	x	0	1ddd.1101.1000.0sss	
lsr Rd, Rs	Logical shift right register	Rd >>= Rs	x	x	x	0	1ddd.1101.1000.1sss	
asr Rd, Rs	Arithmetic shift right register	Rd >>= Rs, preserve sign	x	x	x	0	1ddd.1101.1001.1sss	
lsl Rd, #imm	Logical shift left immediate(1)	Rd <<= imm	x	x	x	0	1ddd.1101.1010.0iii	
lsr Rd, #imm	Logical shift right immediate(1)	Rd >>= imm	x	x	x	0	1ddd.1101.1010.1iii	
asr Rd, #imm	Arithmetic shift right immediate(1)	Rd >>= imm, preserve sign	x	x	x	0	1ddd.1101.1011.1iii	

(1) Shift immediate can be 1 to 8 bits, where imm = 0 corresponds to 8 bits.

Table 20-11. Program Flow Control

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
jmp addr	Jump direct	pc = addr	—	—	—	—	0000.01aa.aaaa.aaaa	addr
jsr addr	Jump subroutine direct	push(stack, pc + 1), pc = addr	—	—	—	—	0001.01aa.aaaa.aaaa	addr
jmp R0	Jump indirect	pc = R0	—	—	—	—	1000.1101.1011.0111	
jsr R0	Jump subroutine indirect	push(stack, pc + 1), pc = R0	—	—	—	—	1001.1101.1011.0111	
rts	Return subroutine	pc = pop(stack)	—	—	—	—	1010.1101.1011.0111	
bra rel	Branch relative	pc = pc + 1 + rel	—	—	—	—	1000.1110.rrrr.rrrr	
b<cc> rel	Branch relative if condition is met	if (cc) pc = pc + 1 + rel	—	—	—	—	1ccc.c110.rrrr.rrrr	
bev0 #ev, rel	Branch if event 0	if (!events[ev]) pc = pc + 1 + rel	—	—	—	—	1eee.0001.rrrr.rrrr	
bev01#ev, rel	Branch if event 1	if (events[ev]) pc = pc + 1 + rel	—	—	—	—	1eee.0011.rrrr.rrrr	

For all other instructions and also when not taking a conditional branch, the program counter is incremented by 1: pc = pc + 1. See Table 20-12 for the conditional operations.

Table 20-12. Program Flow Conditions

Syntax <cc>	Description	Condition	Encoding [3:0]
gtu	Greater than, unsigned	!C & !Z	0010
geu / iob0	Greater than or equal to, unsigned / Tested I/O bit = 0	!C	0100
eq / z	Equal / Zero	Z	0110
novf	Not overflow	!V	1000
pos	Positive	!N	1010
ges	Greater or equal to, signed	(N & V) | (!N & !V)	1100
gts	Greater than, signed	((N & V) | (!N & !V)) & !Z	1110
leu	Less than or equal to, unsigned	C | Z	0011
ltu / iob1	Less than, unsigned / Tested I/O bit = 1	C	0101
neq / nz	Not equal / Non-zero	!Z	0111
ovf	Overflow	V	1001
neg	Negative	N	1011
lts	Less than, signed	(N & !V) | (!N & V)	1101
les	Less than or equal to, signed	(N & !V) | (!N & V) | Z	1111

Table 20-13. Loop Flow Control

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
loop R1, rel	Loop register, n = 1..255[1]	lc = LSB(R1), ls = pc + 1, le = pc + rel	x	x	x	x	1000.0101.rrrr.rrrr	
loop #n, rel	Loop immediate, n = 2^x, (x = 1..7)[1]	lc = n, ls = pc + 1, le = pc + rel	x	x	x	x	1nnn.0101.rrrr.rrrr	

(1) The assembler offsets the end-of-loop label so it can be placed after the last instruction of the loop.

Table 20-14. Power Management

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
wev0 #ev	Wait for event 0	Stop until events[ev] == 0	x	x	x	x	1eee.1101.1011.0000	
wev1 #ev	Wait for event 1	Stop until events[ev] == 1	x	x	x	x	1eee.1101.1011.0001	
sleep	Sleep	Stop until wakeup, then pc = 2 * vector	x	x	x	x	1011.1101.1011.0111	

Table 20-15. Miscellaneous

Syntax	Description	Operation	Status Flags				Encoding [15:0]	Prefix
			Z	N	C	V		
nop	No operation	R7 = R7 (no effect)	x	x	x	x	1111.1101.0100.0111	
pfix #imm	Prefix[1]	Use prefix on next instruction	x	x	x	x	1000.0110.iiii.iiii	
dw #imm	Data word	Inserts 16-bit immediate data value	x	x	x	x	iiii.iiii.iiii.iiii	

(1) The prefix (`pfix`) instruction is inserted automatically by the assembler where needed. Do not insert this instruction manually.

20.3.2.5 SCE Event Interface

Table 20-16 summarizes the Sensor Controller event interface.

Table 20-16. SCE Event Interface

Event Name	Number	Description
AUX_PROG_DLY_IDLE	0	Programmable delay event. See Section 20.3.2.7 or AUX_EVCTL:PROGDLY (in Section 20.8.3).
AUX_COMPA	1	Comparator A event
AUX_COMPB	2	Comparator B event
AUX_TDC_DONE	3	TDC conversion done or time-out event
AUX_TIMER0_EV_OR_IDLE	4	Timer0 reached its target or is idle
AUX_TIMER1_EV_OR_IDLE	5	Timer1 reached its target or is idle
AUX_ADC_FIFO_NOT_EMPTY	6	ADC FIFO contains samples
SCEWEV_PROG	7	Programmable event. See AUX_EVCTL:SCEWEVCFG0 and AUX_EVCTL:SCEWEVCFG1 in Section 20.8.3.

The following instructions use the events listed in Table 20-16 for controller core power management or program flow control:

- *wev0* and *wev1* instructions:

  The Sensor Controller halts instruction execution, freezes the internal state, and becomes clock-gated until the selected event becomes 0 or 1.
- *bev0* and *bev1* instructions:

  The Sensor Controller makes a conditional branch depending on the event level.

The SCEWEV_PROG event can be programmed as a function of up to two events with configurable polarities. This programming is useful for example, when the Sensor Controller must wait for an I/O event with time-out, which can be achieved while consuming minimal power.

20.3.2.6 Math Accelerator (MAC)

The Sensor Controller can leverage the MAC module for mathematical operations, such as:

- 16-bit signed or unsigned multiplication with optional accumulation of the result
- 16-bit or 32-bit signed or unsigned addition of programmable term and accumulator

The results from these operations are always stored in the 40-bit signed accumulator. The Sensor Controller can perform the following operations directly on the accumulator:

- Initialize the accumulator
- Reset the accumulator
- Read arbitrary 16-bit slice of the accumulator
- Read the number of leading sign or zero bits in the accumulator
- Perform logical shifts and arithmetic right-shift of the accumulator

Figure 20-7 shows a block diagram of the MAC.

For consistent timing, it is important that the Sensor Controller and the MAC operational rates are equal. Hence, the Sensor Controller application must set the MAC_OP_RATE bit of the AUX_SYSIF:PEROPRATE register to SCE_RATE (see Section 20.8.9).

Though not the primary use case, the System CPU can also use the MAC. To achieve consistent timing in this case, the application must set the MAC_OP_RATE bit of the AUX_SYSIF:PEROPRATE register to BUS_RATE (see Section 20.8.9).

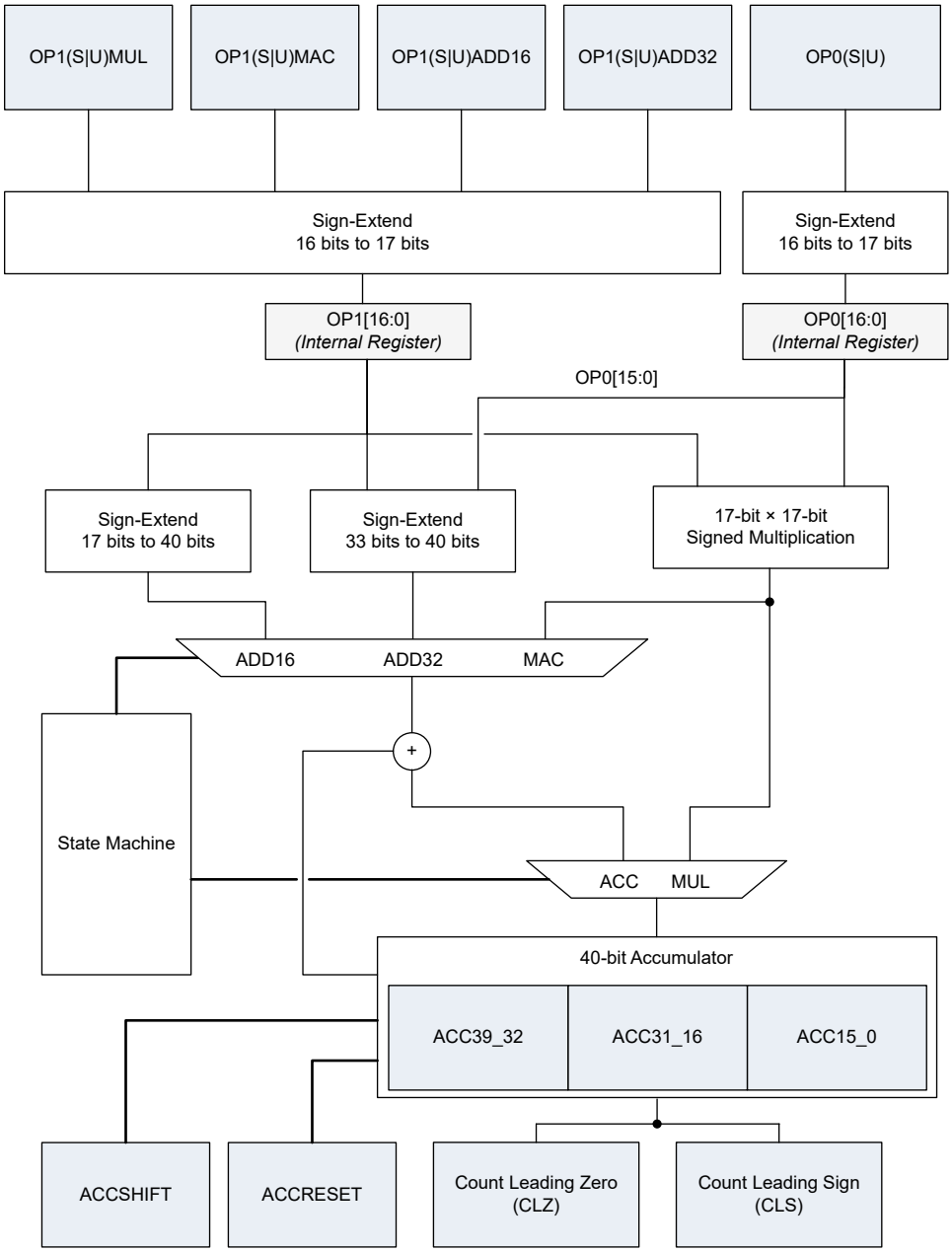


Figure 20-7. MAC Block Diagram

The result of addition and multiplication operations is still valid if operations are unequal, but the time to complete the operations can vary.

The addition and multiplication operations require multiple peripheral clock cycles to complete. Consider the following assembly example that was generated from the Sensor Controller Studio.

```
# Sample assembly (Example code produced by the Sensor Controller Studio)
# The contents of r4 are assumed to be signed in this example
0: out     R4,[#IOP_MAC_OP1SADD16]   ; add r4 to the accumulator
1: in      R1,[#IOP_MAC_15_0]        ; Store result bits 15:0 in r1
2: out     R4,[#IOP_MAC_OP1SMUL]     ; Multiply r4 with operand 0(assumed pre-programmed)
3: in      R2,[#IOP_MAC_31_16]       ; Store result bits 31:16 in r2
4: in      R3,[#IOP_MAC_15_0]        ; Store result bits 15:0  in r3
# The Sensor Controller uses the alias I/O memory space
IOP_MAC_OP1SMUL                 equ 11
IOP_MAC_OP1SADD16               equ 15
IOP_MAC_31_16                   equ 24
IOP_MAC_15_0                    equ 23
```

As shown in Figure 20-8, the addition operation requires one peripheral clock cycle to complete. Because the Sensor Controller attempts to read the result of the operation immediately after triggering it, the Sensor Controller stalls for one clock cycle. This condition gates the clock to the Sensor Controller until the result is ready.

Figure 20-8 also shows that the multiplication operation requires five peripheral clock cycles to complete. As the Sensor Controller tries to read the result of the operation immediately after triggering it, the Sensor Controller stalls for five clock cycles. The clock to the Sensor Controller becomes clock gated until the result is ready. The multiply-accumulate operation requires the same number of peripheral clock cycles to complete.

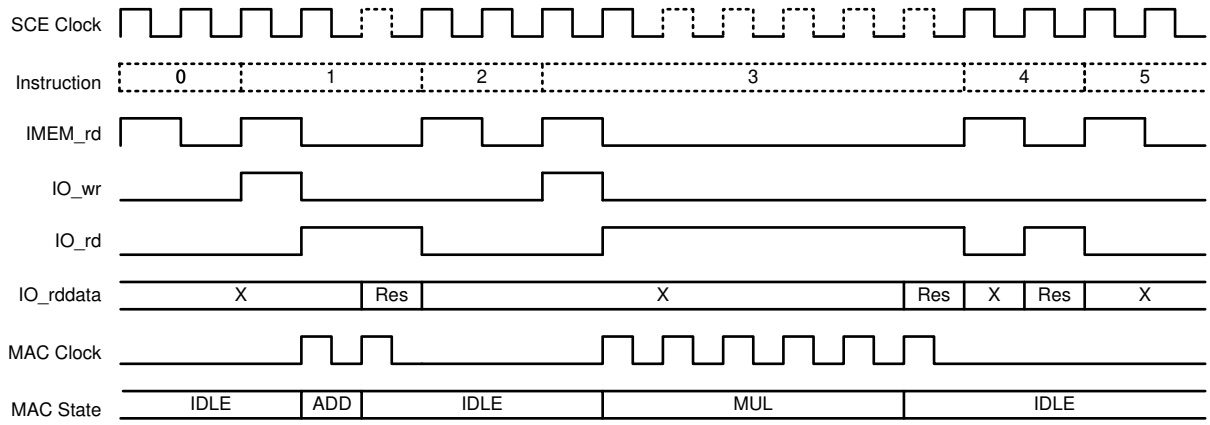


Figure 20-8. MAC Timing Diagram

20.3.2.7 Programmable Microsecond Delay

Table 20-16 describes the AUX_PROG_DLY_IDLE event. The level of this event depends on the value of the 16-bit AUX_EVCTL:PROGDLY counter. When the counter value is zero, the level is high; otherwise, the level is low.

When the counter is loaded with a nonzero value, it decrements by 1 LSB for each edge of a 500 kHz clock with 50% duty cycle until the counter value is zero. The 500 kHz clock signal is synchronized at the SCE rate. Hence, one LSB corresponds to a 1 µs delay on average because the edge-detect logic is susceptible to synchronization jitter.

The next example shows how to delay further Sensor Controller execution by a minimum of 15 µs.

```
# Sample assembly (Example code produced by the Sensor Controller Studio)
0: ld      R4, #(15+1)              ; Load value for AUX_EVCTL:PROGDLY counter
1: out     R4, [#IOP_EVCTL_PROGDLY] ; Load the AUX_EVCTL:PROGDLY counter
2: wev1    #0                       ; Wait until AUX_EVCTL:PROGDLY counter is 0.
3: ...
# The Sensor Controller uses the alias I/O memory space
IOP_EVCTL_PROGDLY               equ 73
```

The AUX_EVCTL:PROGDLY write instant and the 500 kHz clock edges have random phase. Therefore, it is required to add 1 µs to the 15 µs to ensure a delay of at least 15 µs.

The AUX_EVCTL:PROGDLY counter is functional only in Active and Low-Power operational modes. The AUX_PROG_DLY_IDLE event must not be used in power-down operational mode.

20.3.2.8 Wake-Up Event Handling

Figure 20-9 shows that the Sensor Controller implements an 8-bit event interface and an 8-bit vectorized wake-up event interface.

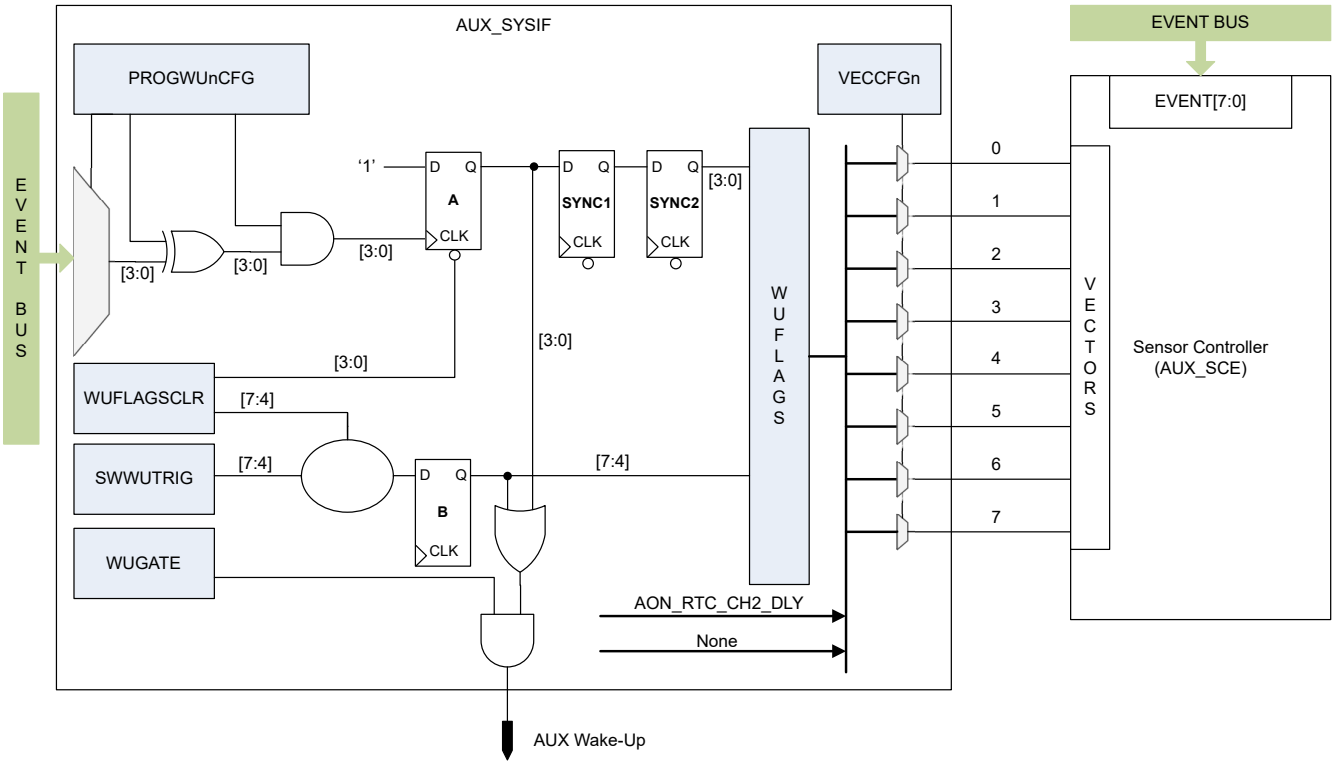


Figure 20-9. SCE Wake-Up and Event Interface

The AUX_SYSIF:VECCFGn registers map each of the vector inputs to a selectable wake-up event (see Section 20.8.9). Table 20-17 lists the possible wake-up events.

Table 20-17. Wake-Up Event Interface

Event Mapping	Value	Description
NONE	0	Disable vector input
PROG_WU0	1	AUX_SYSIF:WUFLAGS.PROG_WU0
PROG_WU1	2	AUX_SYSIF:WUFLAGS.PROG_WU1
PROG_WU2	3	AUX_SYSIF:WUFLAGS.PROG_WU2
PROG_WU3	4	AUX_SYSIF:WUFLAGS.PROG_WU3
SW_WU0	5	AUX_SYSIF:WUFLAGS.SW_WU0
SW_WU1	6	AUX_SYSIF:WUFLAGS.SW_WU1
SW_WU2	7	AUX_SYSIF:WUFLAGS.SW_WU2
SW_WU3	8	AUX_SYSIF:WUFLAGS.SW_WU3
AON_RTC_CH2_DLY	9	AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

The Sensor Controller executes the *sleep* instruction when the wake-up event interface is properly configured. The *sleep* instruction halts instruction execution, freezes the internal state, and clock-gates the Sensor Controller until one or more wake-up inputs are set. Instruction execution will then resume from the vector address corresponding to the wake-up input of highest priority. Highest priority is assigned to vector 0 and priority decreases linearly down to vector input 7, as listed in Table 20-18.

Table 20-18. Wake-Up Vector Priority

Wake-Up Input	Handler Address	Priority
0	0x0	Highest
1	0x2	
2	0x4	
3	0x6	
4	0x8	
5	0xA	
6	0xC	
7	0xE	Lowest

If one or more wake-up inputs are already set, the sleep instruction will immediately cause the Sensor Controller to execute from the vector address of the highest priority.

Comparing the wake-up events listed in Table 20-17 with Figure 20-9 shows that all the possible wake-up events except AON_RTC_CH2_DLY can also drive the AUX domain wake-up event. The AUX domain wake-up event is driven by:

- Four programmable AUX wake-up events, configurable with respect to:
  - Event source (AUX_SYSIF:PROGWUnCFG.WU_SRC)
  - Polarity (AUX_SYSIF:PROGWUnCFG.POL)
  - Level- or edge-selectivity (AUX_SYSIF:WUFLAGSCLR)
- Four software events (AUX_SYSIF:SWWUTRIG)

The System CPU uses these events to perform handshaking with the Sensor Controller.

Hence, the Sensor Controller will typically:

- Configure one or more of the programmable AUX wake-up events
- Configure the Sensor Controller wake-up vector interface
- Request Power-Down operational mode
- Execute the *sleep* instruction
- Wake up:
  - Request Low-Power or Active operational mode
  - Clear the wake-up flag
  - Perform a task and possibly notify the System CPU
- Enable an AUX wake-up event
- Request Power-Down operational mode
- Execute *sleep* instruction

The requested system resources during *sleep* instruction are minimal, which lowers the overall system power consumption. Upon AUX wakeup, the system power controller externally forces Low-Power mode or Active AUX operational mode. The Sensor Controller must request the same operational mode before it clears the wake-up flag. For further details on AUX operational mode switching, see Section 20.2.

20.3.2.9 Access to AON Domain Registers

The Sensor Controller does not have access to the AON domain in general. Still, it can access certain system functionality. This access includes:

Read access to the following Battery Monitor and Temperature Sensor registers:
- AON_BATMON:BAT
- AON_BATMON:TEMP

For further description, see the AUX_SYSIF:BATMONBAT and AUX_SYSIF:BATMONTEMP registers in Section 20.8.9.

- Read access to the following Real-Time Clock registers:
  - AON_RTC:SEC.VALUE[15:0]
  - AON_RTC:SUBSEC.VALUE[31:16]

  For further description, see the AUX_SYSIF:RTCSEC and AUX_SYSIF:RTCSUBSEC registers in Section 20.8.9.
- Update Real-Time Clock subsecond increment:

  For further description, see the AUX_SYSIF:RTCSUBSECINC0, AUX_SYSIF:RTCSUBSECINC1, and AUX_SYSIF:RTCSUBSECINCCTL registers in Section 20.8.9.
- Clear Real-Time Clock channel 2 events:

  For further description, see the AUX_SYSIF:RTCEVCLR register in Section 20.8.9.
- Trigger recharge of the VDDR power rail and to detect if such event took place within a time interval:

  For further description, see Section 20.3.2.10.
- Detection of transitions in the MCU domain power state within a time interval:

  Such change may cause a non-accumulative change to the SCE clock rate. For further description, see the AUX_SYSIF:CLKSHIFTDET register in Section 20.8.9.

20.3.2.10 VDDR Recharge

Recharge of VDDR can occur when the Sensor Controller requests low-power or power-down operational modes. A recharge causes VDDS current spikes that may affect AUX measurements. The Sensor Controller must follow the routine shown in Figure 20-10 to avoid power noise caused by VDDR recharge during a sensitive measurement.

The routine is only valid when AON_PMCTL:RECHARGECFG.MODE is set to COMPARATOR, as this setting allows the Sensor Controller to execute tasks in Low-Power or Power-Down operational modes.

The Sensor Controller can also detect if the system power controller recharges VDDR within a certain time interval. The procedure is described in AUX_SYSIF:RECHARGEDET (see Section 20.8.9).

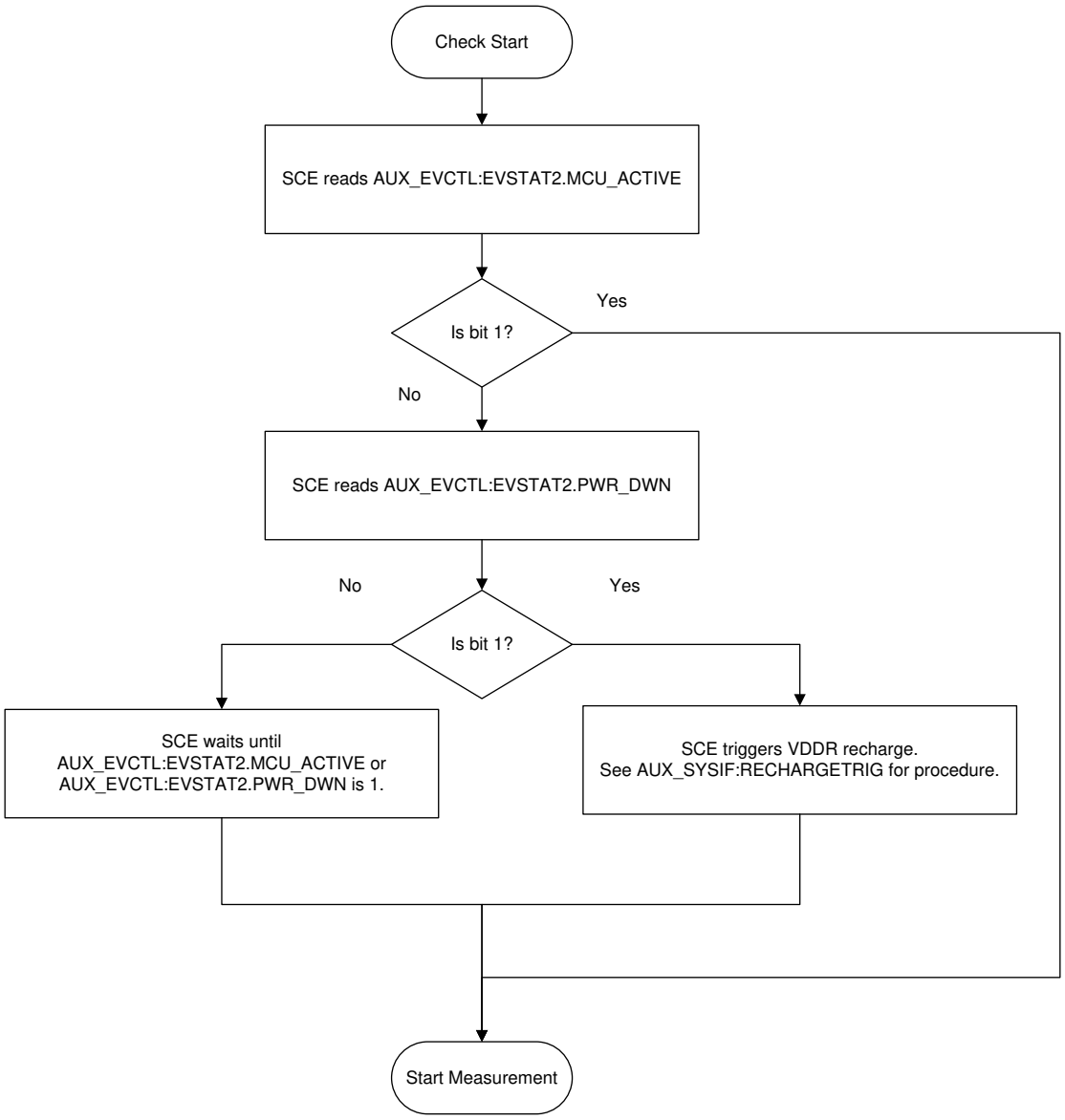


Figure 20-10. Avoid VDDR Recharge Power Noise

20.4 Digital Peripheral Modules

20.4.1 Overview

Table 20-19 lists the digital peripherals in the AUX domain together with respective operational rates and address space. For a description about operational rates, see Section 20.2.

Table 20-19. Digital Peripherals

Digital Peripheral	Operational Clock Rate(1)		
	AUX Bus Rate	AUX SCE Rate	AUX Timer2 Rate
AUX_SPIM	(X)	X	
AUX_TIMER2			X
AUX_TDC	X		
AUX_TIMER01	(X)	X	
AUX_SMPH	X		
AUX_AIODIO0		X	
AUX_AIODIO1		X	
AUX_AIODIO2		X	
AUX_AIODIO3		X	
AUX_DDI0_OSC	X		

(1) X = Default rate
(X) = Rate override capability
(XX) = Rate override capability for event synchronization

The Sensor Controller and the System CPU can share digital peripherals in the AUX domain. If this is required, it is recommended to use the hardware semaphores in SMPH to arbitrate access to the peripheral. For the same reason it is recommended that the System CPU and the Sensor Controller leaves the peripherals in reset state after usage. Table 20-21 lists how TI software assigns resources to the semaphores.

20.4.1.1 DDI Control-Configuration

AUX_DDI0_OSC (listed in Table 20-19) is not really a digital peripheral. As shown in Figure 20-1, both the System CPU and the Sensor Controller access DDI_0_OSC through the AUX_DDI0_OSC module to control and configure the system clocks. The AUX_DDI0_OSC module supports four different types of accesses to provide efficiency as it connects to the analog domain over a multicycle bus interface. The access types are:

- Direct
- Set
- Clear
- Masked

The Sensor Controller detects less access latency when the bus clock rate is higher than the SCE clock rate.

If the System CPU application requires access to this module, the application must use the TI provided DriverLib API functions. Sensor Controller Studio API functions includes access when necessary and the user need not consider this module when developing code.

For system clock description, see Section 20.2.

As noted in Section 20.3.2.2, SCE access to AUX_DDI0_OSC is restricted when device interconnect and firewall is configured to normal or contingency 1 modes, i.e. whenever PRCM:BUSSECCFG.BUS_CFG takes values other than 0xF9 and 0xF3. In this case, SCE may only access DDI half-words specified by AUX_SCE:NONSECDDIACC0-3. Other accesses will be blocked and SCE suspended.

20.4.2 Analog I/O Digital I/O (AIODIO)

20.4.2.1 Introduction

The AUX analog I/O digital I/O (AUX_AIODIOx) peripherals control the 32 AUX I/Os that map to separate package DIOs. The Sensor Controller is the primary owner of the AUX I/Os, though the System CPU can configure and use them. The only relevant System CPU use scenario is to configure the AUX I/Os before control is passed to the Sensor Controller.

The four AIODIO instances configure and control the 32 AUX I/Os. Each AIODIO peripheral manages 8 AUX I/Os, as shown in Table 20-20.

Table 20-20. AUX I/O Control

Peripheral	AUX I/O
AUX_AIODIO0	0 to 7
AUX_AIODIO1	8 to 15
AUX_AIODIO2	16 to 23
AUX_AIODIO3	24 to 31

Peripheral features are:

- 32 general-purpose I/Os with configurable I/O modes:
  - Digital input
  - Digital output
  - Open-drain digital output
  - Open-source digital output
- Up to eight I/Os support analog transfer.
- Configurable output data source:
  - Peripheral
  - Event
  - Register

Figure 20-11 shows the block diagram of a single AUX I/O(k) and how it connects to DIO(n).

20.4.2.2 Functional Description

20.4.2.2.1 Mapping to DIO Pins

The package-specific AUXIO–DIO mapping is listed in Table 15-2. To connect DIO(n) to AUX IO(k), the user must set IOC:IOCFGn.PORT_ID to AUX_IO. This action is not required when developing code in the Sensor Controller Studio, as the tool handles the pin mapping and the required IOC configuration.

20.4.2.2.2 Configuration

Section 20.4.2.2.3 describes the AUX I/O configuration. Direct user configuration is not required when developing code in Sensor Controller Studio. These users interact directly with the Sensor Controller Studio APIs.

20.4.2.2.3 GPIO Mode

AUX_AIODIOn:IOMODE (see Section 20.8.2) configures I/O mode for each AUX I/O as:

- Input
- Digital output
- Open-drain digital output
- Open-source digital output

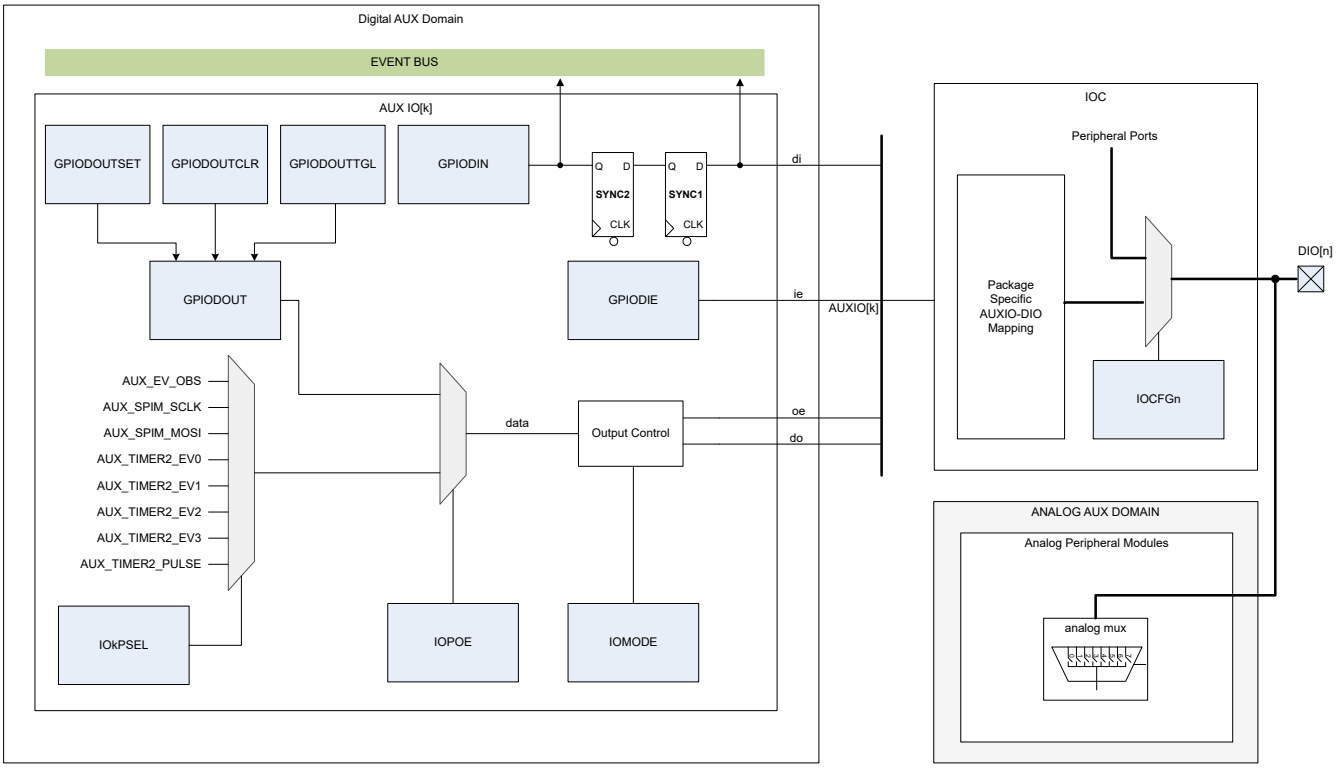


Figure 20-11. AUX I/O Block Diagram

20.4.2.2.4 Input Buffer

The digital input buffer must be enabled to use the AUX I/O as digital input. The buffer is enabled when the user sets the corresponding bit in AUX_AIODIOn:GPIODIE.

The AUX I/Os are synchronized by default at the SCE clock rate and when the System CPU writes the AIODIO instance. The latter must be confined to initialize before the AUX peripheral or Sensor Controller function to get constant synchronization clock rate.

Up to eight AUX I/Os enable analog transfer between DIOs and the analog AUX peripherals. To enable analog transfer, set the I/O mode to *input* and disable the digital input buffer. To determine which AUX I/O enables analog transfer, see Table 15-2.

20.4.2.2.5 Data Output Source

Each AUX I/O propagates data as per I/O mode output configuration. The data source is controlled by AUX_AIODIOn:IOPOE as:

- **Direct mode:** The data source is the internal AUX_AIODIOn:GPIODOUT register.
- **Peripheral mode:** The data source is selected by AUX_AIODIOn:IOnPSEL.SRC.

For the register description, see Section 20.8.2.

20.4.3 Semaphore (SMPH)

20.4.3.1 Introduction

The AUX semaphore (AUX_SMPH) peripheral contains eight hardware semaphores. The System CPU and the Sensor Controller use the semaphores to implement resource ownership.

20.4.3.2 Functional Description

The hardware semaphore functionality is simple. To reserve a resource such as the rights to access and use a module, the System CPU application requests a specific semaphore by read operation. The read value determines the ownership:

- 0 = The Sensor Controller owns the semaphore and the resource associated with it.
- 1 = The System CPU owns the semaphore and the resource associated with it.

To reserve a resource such as the rights to access and use a module, the Sensor Controller task requests a specific semaphore by read operation. The read value determines the ownership:

- 0 = The System CPU owns the semaphore and the resource associated with it.
- 1 = The Sensor Controller owns the semaphore and the resource associated with it.

Only the owner of a semaphore shall release it by writing the value 1 to the semaphore. There is no hardware mechanism that forces this rule.

To avoid polling and to save power, the System CPU can request a semaphore by writing the SMPH_ID to AUX_SMPH:AUTOTAKE.SMPH_ID. The AUX_EVCTL:EVTOMCUFLAGS. The AUX_SMPH_AUTOTAKE_DONE event becomes 1 when that semaphore belongs to the System CPU. The Sensor Controller must not use this feature. There is no hardware mechanism that forces this rule.

20.4.3.3 Semaphore Allocation in TI Software

Table 20-21 summarizes how TI software assigns resources to the semaphores.

Given the resource-to-semaphore mapping listed in Table 20-21, the System CPU and the Sensor Controller must acquire the semaphores in the same order when the application requires more than one semaphore. Hence, if both the System CPU and the Sensor Controller application require the COMPA and Reference DAC peripherals, both applications must request SMPH_ID 4, then SMPH_ID 3, or the other way around.

Table 20-21. Semaphore Resource Allocation

SMPH_ID	Resource Allocation
0	Unassigned by TI
1	TDC
2	ADC + COMPB
3	COMPA + ISRC
4	Reference DAC
5	Timer2
6	Unassigned by TI
7	Unassigned by TI

20.4.4 SPI Master (SPIM)

20.4.4.1 Introduction

The AUX SPI Master (AUX_SPIM) peripheral enables the Sensor Controller with power-efficient SPI communication. The Sensor Controller is the primary owner of this peripheral, though it can be shared with or owned by the System CPU.

The SPI master handles the generation of SCLK signals, MOSI signals, and the sampling of MISO. The SPI chip select (CS) signal is not controlled by the peripheral; therefore, the Sensor Controller must control this pin separately.

Peripheral features are:

- Configurable SCLK frequency
- Configurable phase and polarity
- 8-bit and 16-bit transfer size support
- SCLK, MOSI, and MISO can connect to all AUX I/Os
- Unbuffered RX- and TX-shift register

Figure 20-12 shows the block diagram of the SPI master and how it connects to DIOs.

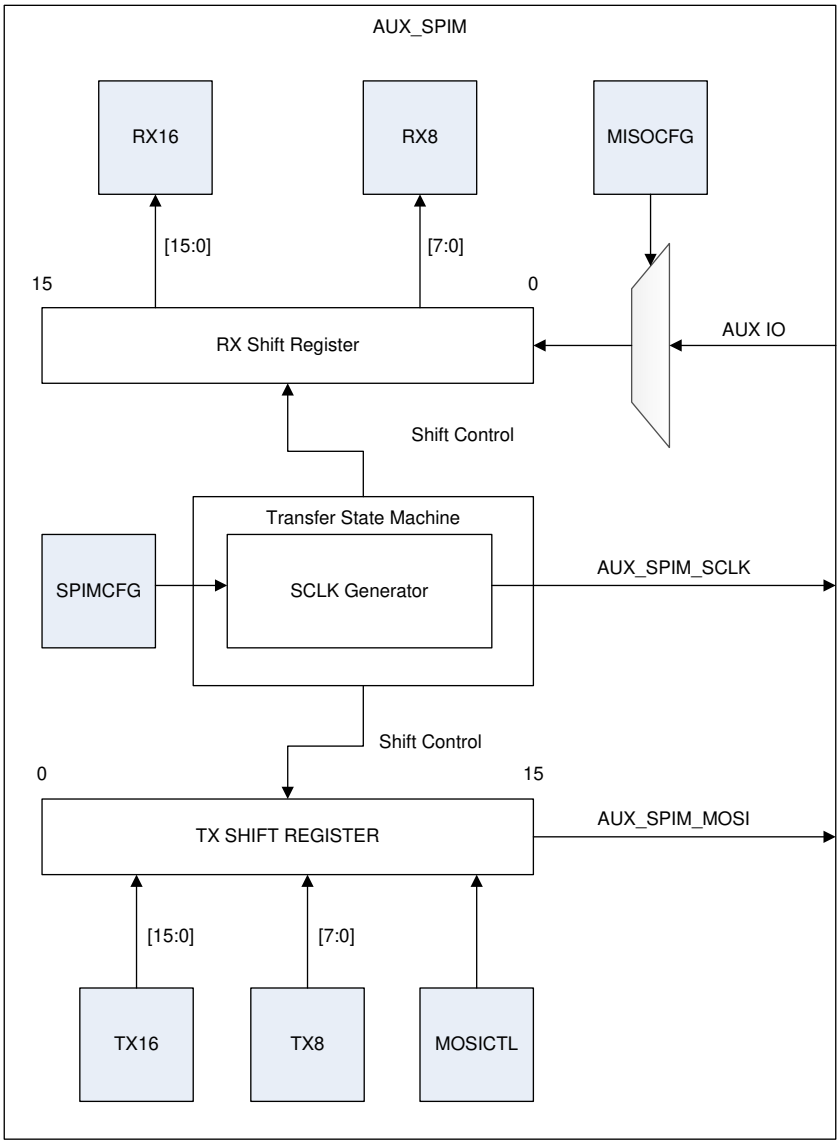


Figure 20-12. AUX_SPIM Block Diagram

20.4.4.2 Functional Description

20.4.4.2.1 TX and RX Operations

Write AUX_SPIM:TX8 or AUX_SPIM:TX16 to transfer 8- or 16-bit data between the SPI master and slaves. Data propagates first to the MSB of the I/O. When transfer completes, MOSI stays at value of the LSB.

An ongoing SPI transfer blocks register access:

- Read access to AUX_SPIM:SCLKIDLE completes when SCLK is idle.
- Read access to AUX_SPIM:DATAIDLE completes when the LSB-bit period is complete.
- Read access to AUX_SPIM:RX8 and AUX_SPIM:RX16 completes when the LSB is captured.
- Write access to any register completes when the LSB-bit period is complete.

The Sensor Controller halts when its access gets blocked, which is then exploited to reduce code size and to save power.

20.4.4.2.2 Configuration

Minimal configuration is required to set up the SPI master and the required I/O mapping.

The AUX_SPIM:SPIMCFG register configures the following:

- SCLK prescaler:
  - The SCLK is derived from the peripheral clock by division, given by the value of the DIV field.
- Phase of MOSI and MISO data signals:
  - The PHA field selects when to shift MOSI and when to sample MISO. MISO is always sampled by the SCLK edge in the middle of the bit period.
- SCLK polarity:
  - The POL field selects the idle state of SCLK.

Connect the peripheral SPI signals to AUX I/Os by:

- SCLK:
  - Determine the AUX I/O that connects to SCLK.
  - Set the AUX I/O peripheral output to AUX_SPIM_SCLK.
  - Enable peripheral output for the AUX I/O.
  - Set the I/O mode to digital output.
- MOSI:
  - Determine the AUX I/O that connects to MOSI.
  - Set the AUX I/O peripheral output to AUX_SPIM_MOSI.
  - Enable peripheral output for this AUX I/O.
  - Set the I/O mode to digital output.
- MISO:
  - Determine the AUX I/O that connects as MISO.
  - Set I/O mode to input.
  - Enable the digital input buffer.
  - Configure AUX_SPIM:MISOFCG to the chosen AUX I/O.

Chip select (CS) is controlled separately by the host. Configure the AUX I/O that functions as CS as follows:

- Determine the AUX I/O that functions as CS.
- Disable peripheral output for this AUX I/O.
- Set the I/O mode to digital output.

Sensor Controller Studio handles the necessary configuration for the user.

20.4.4.2.3 Timing Diagrams

Figure 20-13 and Figure 20-14 show how the configuration of AUX_SPIM:SPIMCFG fields affect the 8-bit data transfer as well as the duration of register access blocking. For the purpose of illustration, the AUX_SPIM:SPIMCFG.DIV is set to 0x4, which gives an SCLK period that is 10 times the peripheral clock period.

In both Figure 20-13 and Figure 20-14, the peripheral clock starts to toggle and blocking begins when the user writes AUX_SPIM:TX8, which indicates the start of transmission. The value of the POL field does not impact the timing of the waveforms; it only sets the IDLE state of SCLK. The value of the PHA field determines the phase of the MOSI and MISO signals with respect to SCLK. As seen in Figure 20-13 and Figure 20-14, this also affects the blocking duration of the AUX_SPIM:SCLKIDLE register access. When PHA = 1, the access completes faster because the SCLK reaches IDLE state earlier.

Note

Set the peripheral clock frequency with AUX_SYSIF:PEROPRATE.SPIM_OP_RATE as follows:

- When then the Sensor Controller owns the peripheral, it must be set to SCE_RATE.
- When the System CPU owns the peripheral, it must be set to BUS_RATE.

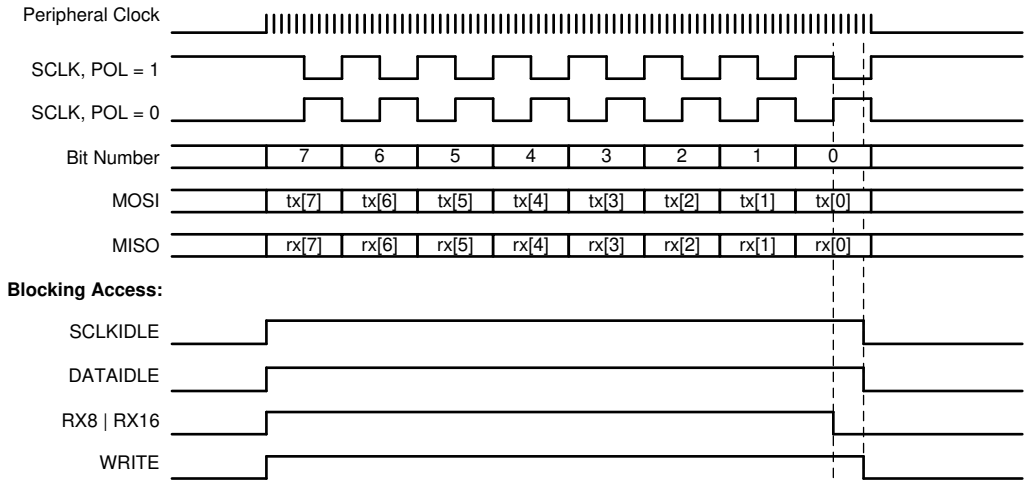


Figure 20-13. SPI Timing Diagram: PHA = 0

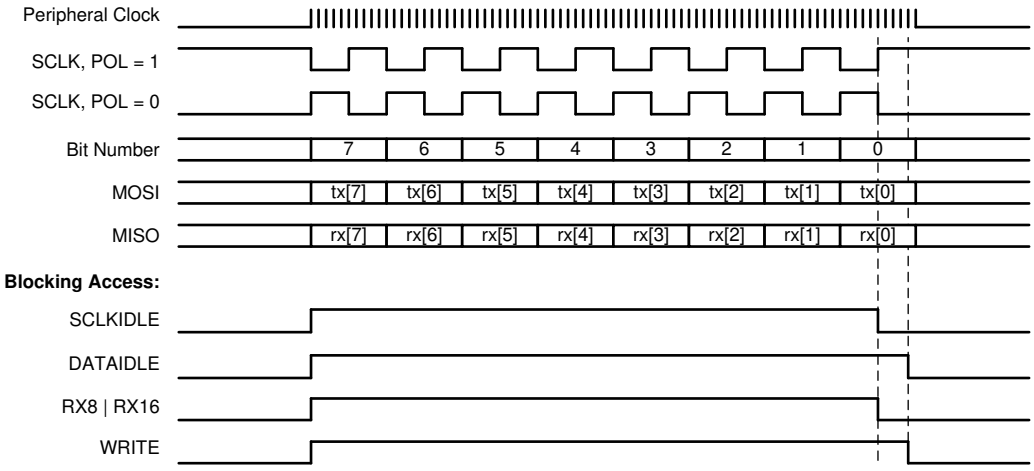


Figure 20-14. SPI Timing Diagram: PHA = 1

20.4.5 Time-to-Digital Converter (TDC)

20.4.5.1 Introduction

The AUX TDC (AUX_TDC) peripheral is a high-precision TDC used to measure the time between a configurable start event and another configurable stop event. The TDC counts at either 96 MHz or 48 MHz, using both edges of a selected counter clock source.

TDC peripheral features are:

- Configurable counter clock source
- Configurable reference clock source for on-chip clock calibration
- Configurable start event source and polarity
- Configurable stop event source and polarity
- Optional prescaling of start and stop events:
  - The prescaler can also be used for standalone, asynchronous pulse counting.
- Optional stop event ignore counter:
  - Several periods can be measured during a single TDC conversion.
- 25-bit TDC conversion counter with configurable saturation limit
- TDC measurement-done event

The TDC is typically used for the following:

- Frequency measurements
- Capacitive sensing, in conjunction with the analog AUX peripherals *ISRC* and *COMPA*
- Precise time between arbitrary event measurements
- Standalone, asynchronous pulse counting using the prescaler

The Sensor Controller and the System CPU share the TDC peripheral. TI's Power Manager uses the TDC peripheral for RCOSC calibration. Both masters must implement and honor resource sharing to use this peripheral. For details, see Section 20.4.3.3.

Figure 20-15 shows the block diagram of the TDC. It also shows the connections to the Oscillator Control (OSCCTL) module, which requires configuration to use the TDC.

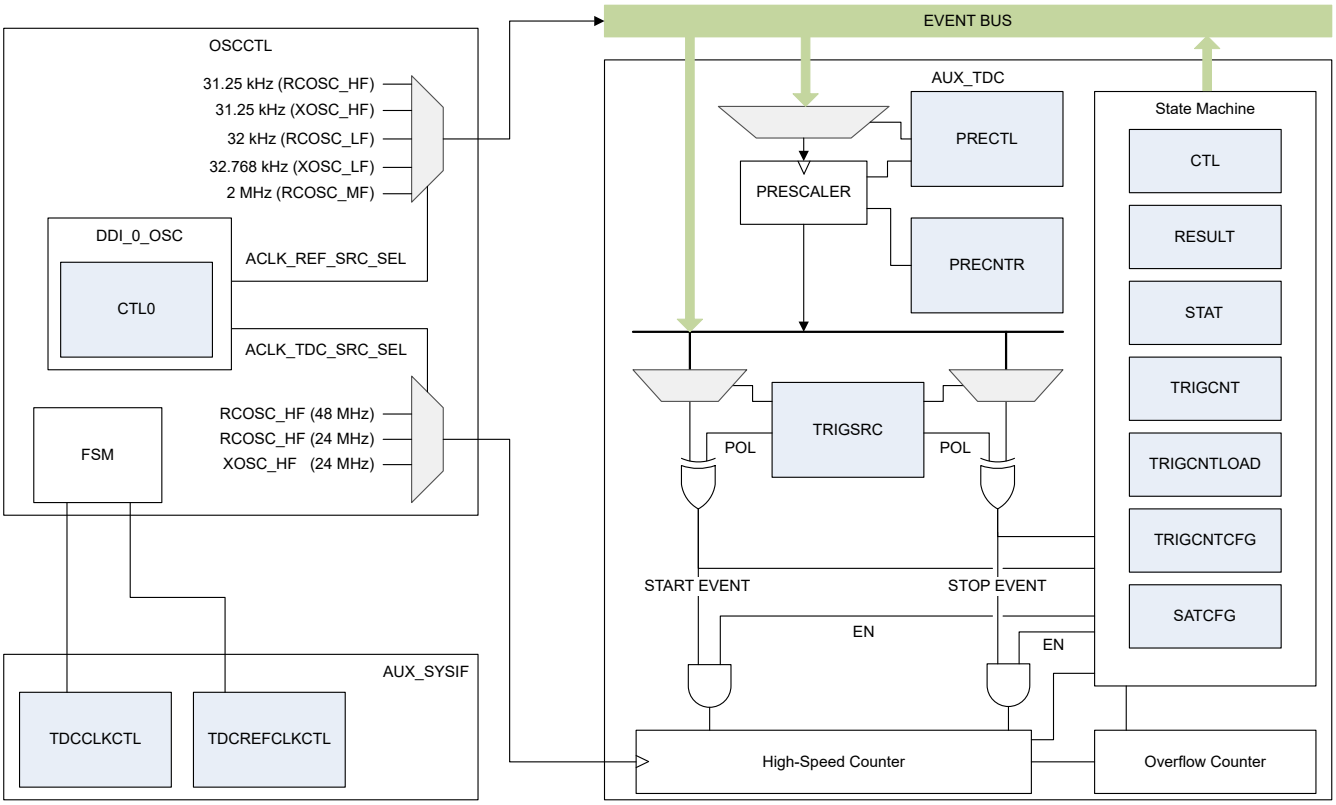


Figure 20-15. AUX_TDC Block Diagram

20.4.5.2 Functional Description

The TDC state machine shown in Figure 20-15 controls and monitors the TDC conversion process. The TDC state machine has the following user interfaces:

- Command
- Conversion Time Configuration
- Status and Results

The TDC state machine performs the conversion based on the configuration of the following:

- Clock source selection
- Start and stop event selection
- Prescaler configuration

Section 20.4.5.2.1 through Section 20.4.5.2.6 explain the interfaces and configuration.

Note

- The Sensor Controller must request active AUX operational mode to use the TDC.
- The System CPU and the Sensor Controller must configure the TDC while AUX_TDC:STAT.STATE = IDLE. Allowed exceptions are the following:
  - Update of AUX_TDC:TRIGCNT
  - Abort command

20.4.5.2.1 Command

The TDC state machine supports the following four user commands:

- Synchronous counter start: The TDC conversion starts when the start event edge occurs, which is useful for frequency measurements.
- Asynchronous counter start: The TDC conversion starts immediately when the start event level occurs. Precise edge-to-edge measurement is not assured unless the user controls the event source. In the latter case, the user must ensure that the start event does not arrive until AUX_TDC:STAT.STATE is less than 0x5, which takes approximately 420 ns.
- Abort: Forces the state machine to abort an ongoing TDC measurement. Do not write this command if AUX_TDC:STAT.STATE equals any of the following values:
  - CLR_CNT
  - WAIT_CLR_CNT_DONE

  Failure to comply may stall the TDC in these states.
- Clear: Clears results and status registers.

For further description, see AUX_TDC:CTL in Section 20.8.5.

20.4.5.2.2 Conversion Time Configuration

The user can configure the TDC state machine to:

- Saturate the conversion result at a selected value:

  This is equivalent to specifying a time-out. For details, see AUX_TDC:SATCFG in Section 20.8.5.
- Ignore AUX_TDC:TRIGCNTLOAD number of stop events during conversion.

Configure AUX_TDC:TRIGCNTLOAD and set AUX_TDC:TRIGCNTCFG.EN to enable the internal stop-counter while the TDC state machine is IDLE.

To override the current value of the internal stop-counter during a TDC conversion, write AUX_TDC:TRIGCNT. Any read-back of this register must be handled with care because the stop-counter can decrement while the user reads it. In this case, the user may read a value of 1 while there are no stop events left to ignore. The TDC measurement will ignore any updates of the stop-counter in this case. To safely update the stop-counter during a TDC conversion, the update must happen when AUX_TDC:TRIGCNT > 1.

20.4.5.2.3 Status and Result

The following state and conversion status flags are available in the AUX_TDC:STAT register:

- Saturation flag
- Complete flag
- State of TDC state machine

The TDC state machine updates AUX_TDC:RESULT when conversion completes on its own or when it ends by an abort command. At the same time, AUX_TDC:STAT.DONE transitions from low to high.

20.4.5.2.4 Clock Source Selection

20.4.5.2.4.1 Counter Clock

The TDC high-speed counter clock source is configurable by DDI_0_OSC: CTL0.ACLK_TDC_SRC_SEL, as listed in Table 20-22.

Table 20-22. TDC Counter Clock Source

ACLK_TDC_SRC_SEL	Oscillator	Frequency (MHz)
0x0	RCOSC_HF	48
0x1	RCOSC_HF	24
0x2	XOSC_HF	24
0x3	Not Used	Not Applicable

Configuration of DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL is done through the DDI Control-Configuration interface. See Section 20.4.1.1 for details.

The user must activate the selected counter clock source before TDC measurements can start:

- Set AUX_SYSIF:TDCCLKCTL.REQ to request the clock source.
- The clock source is active when AUX_SYSIF:TDCCLKCTL.ACK is set.

  Follow the procedure found in the TDCCLKCTL Register to activate the selected TDC counter clock.

Note

The user must deactivate the counter clock source when the TDC measurements are complete. Failure to do so will prevent the system from entering standby mode because the Oscillator Control module requests resources from the supply system. To remove the clock source request, clear AUX_SYSIF:TDCCLKCTL.REQ.

20.4.5.2.4.2 Reference Clock

Configure the reference clock to measure the frequency of on-chip oscillators. The reference clock source is configurable by DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL, as listed in Table 20-23.

Table 20-23. TDC Reference Clock Source

ACLK_REF_SRC_SEL	Oscillator	Frequency (kHz)
0x0	RCOSC_HF	31.25
0x1	XOSC_HF	31.25
0x2	RCOSC_LF	32
0x3	XOSC_LF	32.786
0x4	RCOSC_MF	2000
0x5–0x7	Not Used	Not Applicable

Configuration of DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL is done through the DDI Control-Configuration interface. For details, see Section 20.4.1.1.

The user must activate the selected reference clock source before a TDC measurements can start:

- Set AUX_SYSIF:TDCREFCLKCTL.REQ to request the reference clock source.
- The clock source is active when AUX_SYSIF:TDCREFCLKCTL.ACK is set.

Note

The user must deactivate any requests from high-frequency oscillators when the TDC measurements are complete. Failure to do so will prevent the system from entering standby mode because the Oscillator Control module requests resources from the supply system. Clear AUX_SYSIF:TDCREFCLKCTL.REQ to remove clock source request.

20.4.5.2.5 Start and Stop Events

The TDC conversion start and stop events are derived from events on the AUX event bus. As shown in Figure 20-15, the AUX_TDC:TRIGSRC register individually selects two events and corresponding polarity for this purpose. Both events are optionally gated by the TDC state machine, according to user command and configuration.

20.4.5.2.6 Prescaler

The prescaler depicted in Figure 20-15 is used to divide a high-frequency signal to comply with the timing requirements detailed in Section 20.4.5.3. The prescaler divides an event from the event bus to generate a lower-frequency event named *AUX_TDC_PRE*. The division ratio is either 16 or 64, and the resulting event waveform has a 50% duty cycle. The user must select AUX_TDC_PRE as both start and stop events for TDC conversion.

Initialize the prescaler with the sequence that follows:

1. Reset the prescaler by setting AUX_TDC:PRECTL.RESET_N to 0. This resets the internal counter, and clears the AUX_TDC_PRE event.
2. Configure prescaler event source, AUX_TDC:PRECTL.SRC.
3. Configure division ratio, AUX_TDC:PRECTL.RATIO.

Next, enable the prescaler event generation by setting AUX_TDC:PRECTL.RESET_N to 1.

The following limitations apply when the prescaler is used for event prescaling:

- The only supported TDC command is synchronous counter start.
- Prescaler input frequency must be less than 24 MHz.
- The TDC conversion result does not account for event prescaling, and software must scale the TDC result appropriately.

20.4.5.3 Supported Measurement Types

The TDC allows the user to measure time between two arbitrary selected events. This section (Section 20.4.5.3.1 through Section 20.4.5.3.4) identifies the most important measurement types and lists the associated event timing requirements. The latter must be ensured to assure correctness and repeatability of the measurements.

20.4.5.3.1 Measure Pulse Width

The following configuration is required to measure the time of high phase or low phase of an event source waveform. Use the sequence that follows:

1. Set AUX_TDC:TRIGSRC.START_SRC equal to AUX_TDC:TRIGSRC.STOP_SRC.
2. Decide on the event source phase to measure:
   - High phase:
     - Set AUX_TDC:TRIGSRC.START_POL = 0
     - Set AUX_TDC:TRIGSRC.STOP_POL = 1
   - Low phase:
     - Set AUX_TDC:TRIGSRC.START_POL = 1
     - Set AUX_TDC:TRIGSRC.STOP_POL = 0
3. Set AUX_TDC:TRIGCNTLOAD = 0 to configure the stop-counter to ignore zero stop events.
4. Set AUX_TDC:TRIGCNTCFG.EN = 1 to enable the stop-counter.

Figure 20-16 illustrates a generic event source waveform.

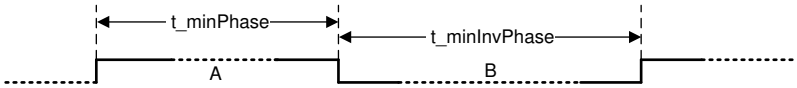


Figure 20-16. Phase Width Timing Requirements

The phase to measure is labeled A, and it is arbitrarily set to the high phase of the event source in Figure 20-16. The opposite phase, which is not measured, is labeled B. Table 20-24 lists the timing requirements for both of these phases.

Table 20-24. Phase Width Timing Requirements

Description	Label	Requirement	Condition
Minimum phase	t_minPhase	292 ns	DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL = 0x0
Minimum phase	t_minPhase	230 ns	DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL = (0x1 | 0x2)
Minimum inverted-phase	t_minInvPhase	126 ns	AUX_TDC:CTL.CMD = 0x1
Minimum inverted-phase	t_minInvPhase	42 ns	AUX_TDC:CTL.CMD = 0x2

If AUX_TDC:CTL.CMD = 0x2, the event source level must equal the level of inverted phase until AUX_TDC:STAT.STATE < 5. Failure to fulfill this requirement makes the TDC conversion start too late.

20.4.5.3.2 Measure Frequency

The following configuration is required to measure the frequency of an event source waveform. Use the sequence that follows:

- Set AUX_TDC:TRIGSRC.START_SRC equal to AUX_TDC:TRIGSRC.STOP_SRC.
- Set AUX_TDC:TRIGSRC.START_POL equal to AUX_TDC:TRIGSRC.STOP_POL.
- Set AUX_TDC:TRIGCNTLOAD to N to allow TDC conversion span N signal periods.
- Set AUX_TDC:TRIGCNTCFG.EN = 1 to enable the stop counter.
- Set AUX_TDC:CTL.CMD = 1 to start the TDC conversion.

Figure 20-17 shows a generic event source waveform.

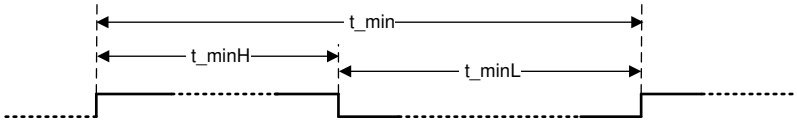


Figure 20-17. Frequency Measurement Waveform

The TDC measures correct signal frequency if the selected event source waveform matches the requirements listed in Table 20-25.

Table 20-25. Timing Requirements for Frequency Measurements

Description	Label	Requirement
Minimum low time	t_minL	208 ns
Minimum high time	t_minH	208 ns
Minimum period	t_min	416 ns

If an event waveform does not directly fulfill these requirements, the user must complete the following sequence:

1. Set AUX_TDC:TRIGSRC.START_SRC to AUX_TDC_PRE.
2. Initialize and enable the prescaler.

Fore more details, see Section 20.4.5.2.6.

20.4.5.3.3 Measure Time Between Edges of Different Events Sources

To measure time between events derived from different event sources, complete the following sequence:

1. Set AUX_TDC:TRIGSRC.START_SRC unequal to AUX_TDC:TRIGSRC.STOP_SRC.
2. Specify AUX_TDC:TRIGSRC.START_POL and AUX_TDC:TRIGSRC.STOP_POL.

The following subsections describe four ways to perform such measurements.

20.4.5.3.3.1 Asynchronous Counter Start – Ignore 0 Stop Events

In this scenario the user:

- Wants to measure the time between the start event and the first stop event
- Sets AUX_TDC:TRIGCNTCFG.EN = 0 to disable the stop counter
- Starts the TDC measurement by writing AUX_TDC:CTL.CMD = 2
- Assures that the start event is low until AUX_TDC.STAT.STATE = WAIT_START
- Assures that the stop event goes high later than the start event

Figure 20-18 shows the scenario for asynchronous counter start – ignore 0 stop events.

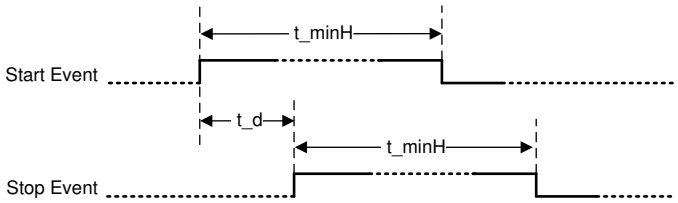


Figure 20-18. Arbitrary Time Measurement 1

Table 20-26 lists the associated timing requirements for this scenario.

Table 20-26. Arbitrary Time Measurement 1

Description	Label	Requirement
Minimum high time	t_minH	42 ns
Minimum delay	t_d	21 ns

20.4.5.3.3.2 Synchronous Counter Start – Ignore 0 Stop Events

In this scenario the user:

- Wants to measure the time between the start event that follows falling edge and the first stop event
- Sets AUX_TDC:TRIGCNTCFG.EN = 0 to disable the stop counter
- Starts the TDC measurement by writing AUX_TDC:CTL.CMD = 1
- Assures that the stop event goes high later than the start event

Figure 20-19 shows the scenario for synchronous counter start – ignore 0 stop events.

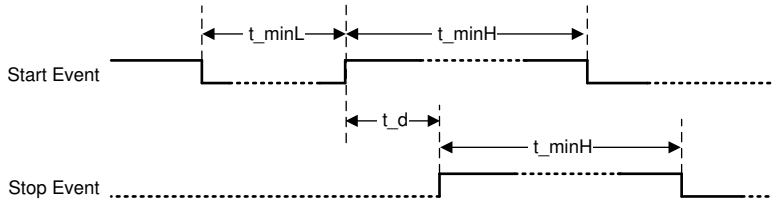


Figure 20-19. Arbitrary Time Measurement 2

Table 20-27 lists the associated timing requirements for this scenario.

Table 20-27. Arbitrary Time Measurement 2

Description	Label	Requirement
Minimum high time	t_minH	42 ns
Minimum low time	t_minL	126 ns
Minimum delay	t_d	21 ns

20.4.5.3.3.3 Asynchronous Counter Start – Ignore Stop Events

In this scenario the user:

- Wants to measure the time between the start event and the Nth stop event, where N > 0:
  - Sets AUX_TDC:TRIGCNTLOAD to N minus 1.
  - Sets AUX_TDC:TRIGCNTCFG.EN = 1 to enable the stop counter.
- Starts the TDC measurement by writing AUX_TDC:CTL.CMD = 2.
- Assures that the start event is low until AUX_TDC.STAT.STATE = WAIT_START.
- Assures that the stop event goes high later than the start event.

Figure 20-20 shows the scenario when N = 2, which ignores 1 stop event.

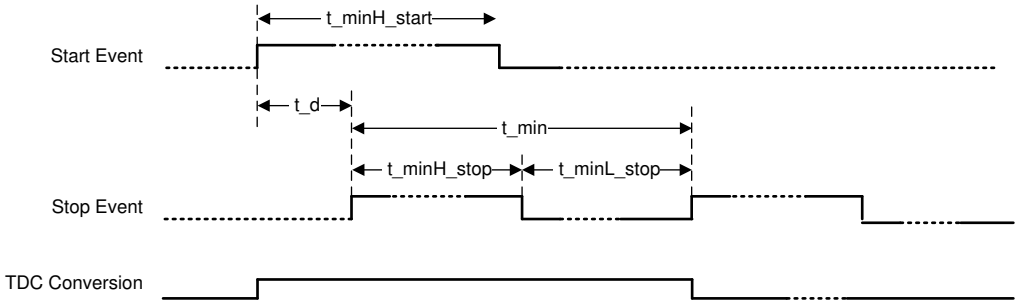


Figure 20-20. Arbitrary Time Measurement 3

Table 20-28 lists the associated timing requirements for this scenario.

Table 20-28. Arbitrary Time Measurement 3

Description	Label	Requirement
Minimum high time, start	t_minH_start	42 ns
Minimum high time, stop	t_minH_stop	42 ns
Minimum low time, stop	t_minL_Stop	168 ns
Minimum stop event period	t_min	210 ns
Minimum delay | TRIGCNTLOAD > 0	t_d	168 ns
Minimum delay | TRIGCNTLOAD = 0	t_d	292 ns

20.4.5.3.3.4 Synchronous Counter Start – Ignore Stop Events

In this scenario the user:

- Wants to measure the time between the start event that follows a falling edge and the Nth stop event, where N > 0.
  - Sets AUX_TDC:TRIGCNTLOAD to N minus 1.
  - Sets AUX_TDC:TRIGCNTCFG.EN = 1 to enable the stop counter.
- Starts the TDC measurement by writing AUX_TDC:CTL.CMD = 1.
- Assures that the stop event goes high later than the start event.

Figure 20-21 shows the scenario when N = 2, which ignores 1 stop event.

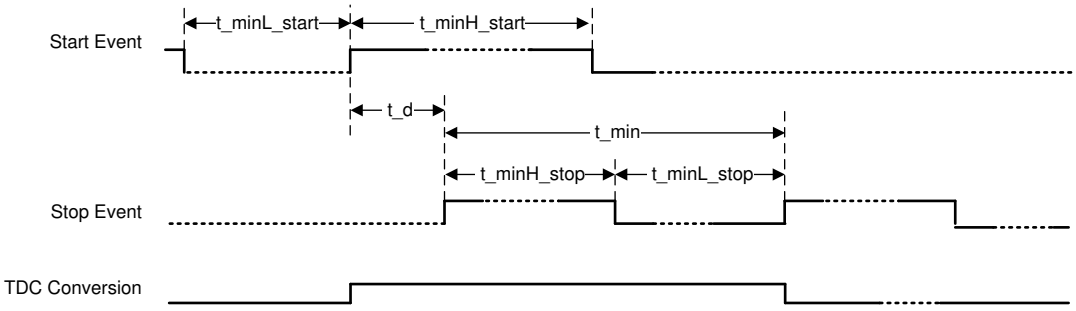


Figure 20-21. Arbitrary Time Measurement 4

Table 20-29 lists the associated timing requirements for this scenario.

Table 20-29. Arbitrary Time Measurement 4

Description	Label	Requirement
Minimum high time, start	t_minH_start	42 ns
Minimum low time, start	t_minL_start	126 ns
Minimum high time, stop	t_minH_stop	42 ns
Minimum low time, stop	t_minL_Stop	168 ns
Minimum stop event period	t_min	210 ns
Minimum delay | TRIGCNTLOAD > 0	t_d	168 ns
Minimum delay | TRIGCNTLOAD = 0	t_d	292 ns

20.4.5.3.4 Pulse Counting

The prescaler can function as a pulse counter when it is not used as event prescaler for TDC start events and stop events.

To initialize the prescaler for the purpose of pulse counting, use the sequence that follows:

1. Set AUX_TDC:PRECTL.RESET_N to 0 to reset the prescaler. This resets the internal counter, and clears the AUX_TDC_PRE event. However, the AUX_TDC_PRE event is not used when counting pulses.
2. Configure AUX_TDC:PRECTL.SRC to set the prescaler event source.

Set AUX_TDC:PRECTL.RESET_N to 1 to enable the prescaler for pulse counting.

The prescaler now requires three or four event pulses before it starts to count. The number of required pulses is dependent on the timing between reset release and event pulse arrival, as shown in Figure 20-22.

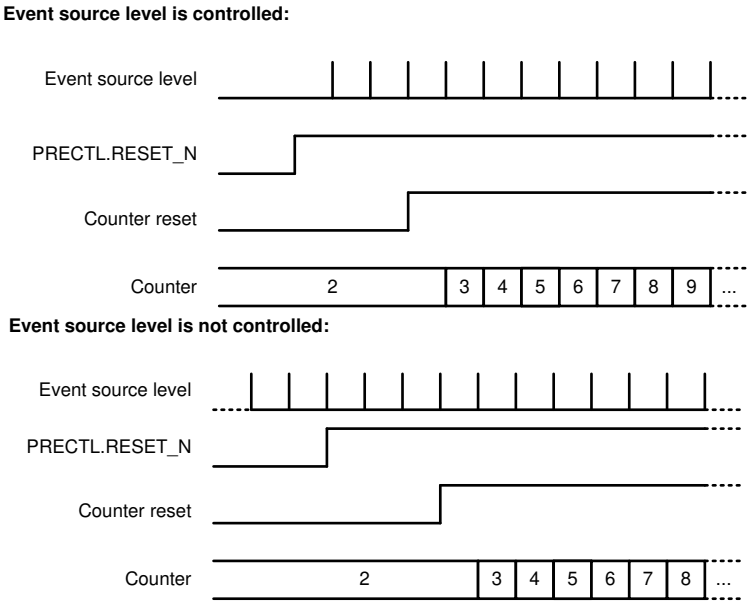


Figure 20-22. Pulse Counting

If the event pulse is inactive at the time of reset release, the counter starts to increment at the fourth event pulse. Hence, the counter value is as follows:

- If number of event pulses is less than 3, counter equals two.
- Otherwise, counter equals number of event pulses minus 1.

The situation becomes different if the user does not control the level of the event at reset release. In this case, the counter may require one more event pulse before it starts to increment. Hence, the counter value could behave as follows:

- If number of event pulses is less than 4, counter equals two.
- Otherwise, counter equals number of event pulses minus 2.

Because of this uncertainty, it is advised to control the level of the selected event at time of reset release.

For information about how to read the prescaler counter value, see AUX_TDC:PRECNTR in Section 20.8.5.

20.4.6 Timer01

20.4.6.1 Introduction

The AUX Timer0 and Timer1 (AUX_TIMER01) peripherals are two identical 16-bit compare timers. The Sensor Controller is the primary owner of these peripherals. However, Timer01 ownership can be shared with or owned by the System CPU.

Timer01 peripheral features are:

- Configurable counter target
- Optional clock source prescaling
- Configurable prescaler clock source:
  - Peripheral clock
  - Configurable event from AUX domain event bus
- One-shot and continuous counter modes:
  - Single or periodical event generation

The main purpose of the timers is to generate events for the AUX domain event bus subscribers. The timers are not used directly for waveform generation. Figure 20-23 shows the block diagram for a single timer.

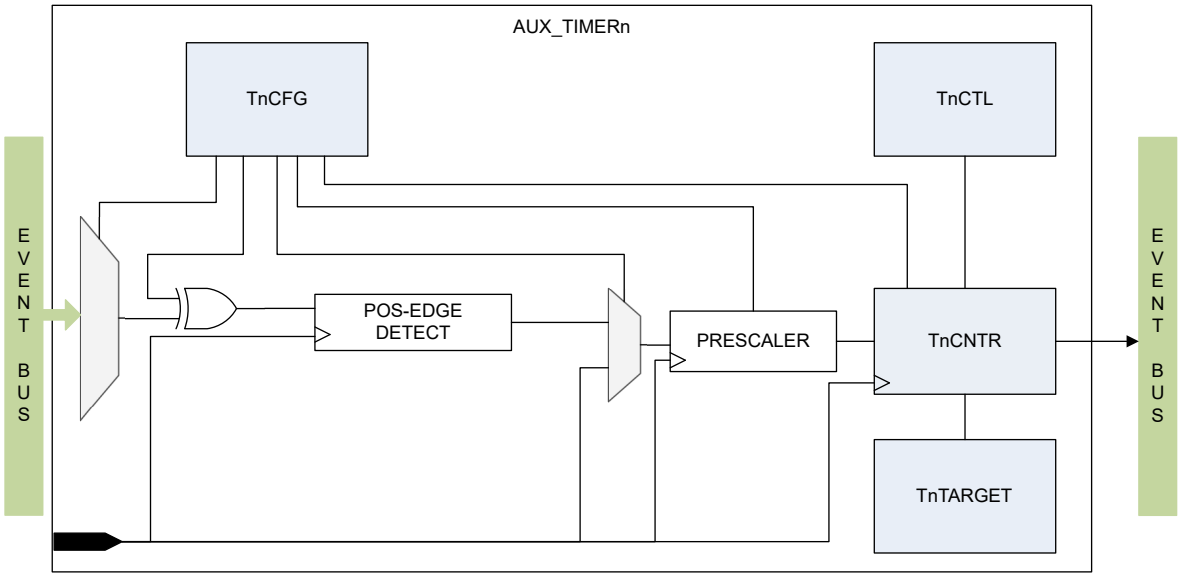


Figure 20-23. AUX_TIMER01 Block Diagram

20.4.6.2 Functional Description

The timers generate an event (AUX_TIMERn_EV) at the end of each counter period. The counter period is determined by the following:

- Counter clock source selection:
  - AUX_TIMER01:TnCFG.MODE selects the peripheral clock or a configurable event as counter clock source. Hence, it sets the counter clock source base frequency.
- Counter clock source prescaling:
  - AUX_TIMER01:TnCFG.PRE divides the counter clock source base frequency. The result is the counter frequency. Division by one is default.
- Counter target value:
  - The counter increments for each counter period as long as AUX_TIMER01:TnCNTR is less than AUX_TIMER01:TnTARGET.

The counter period ends when AUX_TIMER01:TnCNTR is equal to or greater than AUX_TIMER01:TnTARGET. The value of AUX_TIMER01:TnCFG.RELOAD decides if the counter restarts or idles.

Note

The peripheral clock frequency is set by AUX_SYSIF:PEROPRATE.TIMER01_OP_RATE.
- It must be set to SCE_RATE when then the Sensor Controller owns the peripheral.
- It must be set to BUS_RATE when the System CPU owns the peripheral.

20.4.7 Timer2

20.4.7.1 Introduction

The AUX Timer2 (AUX_TIMER2) peripheral is a single asynchronous 16-bit capture-compare timer. The Sensor Controller is the primary owner of this peripheral, though it can be shared with or owned by the System CPU.

Peripheral features are:

- Timer clock is independent of AUX operational mode
- Timer clock source prescaler
  - Counter clock frequency equals divided timer clock source frequency
- Configurable counter target:
  - Static value

  - Direct update
  - Shadow target update
- One-shot and continuous counter modes
- Four event outputs:
  - Manual control
  - Channel control
- Four capture-compare channels:
  - 15 different channel functions offer flexible event generation:
    - Simple event-on-capture
    - Simple event-on-compare
    - PWM
    - Period-pulse-width measurement
  - Channel functions can be one-shot or continuous
  - Channel event can be routed to arbitrary event output
  - Pipeline compare register
- Clock pulse propagation
  - Propagate a single clock pulse to AUX I/O

Figure 20-24 shows the block diagram of the timer with a single capture-compare channel.

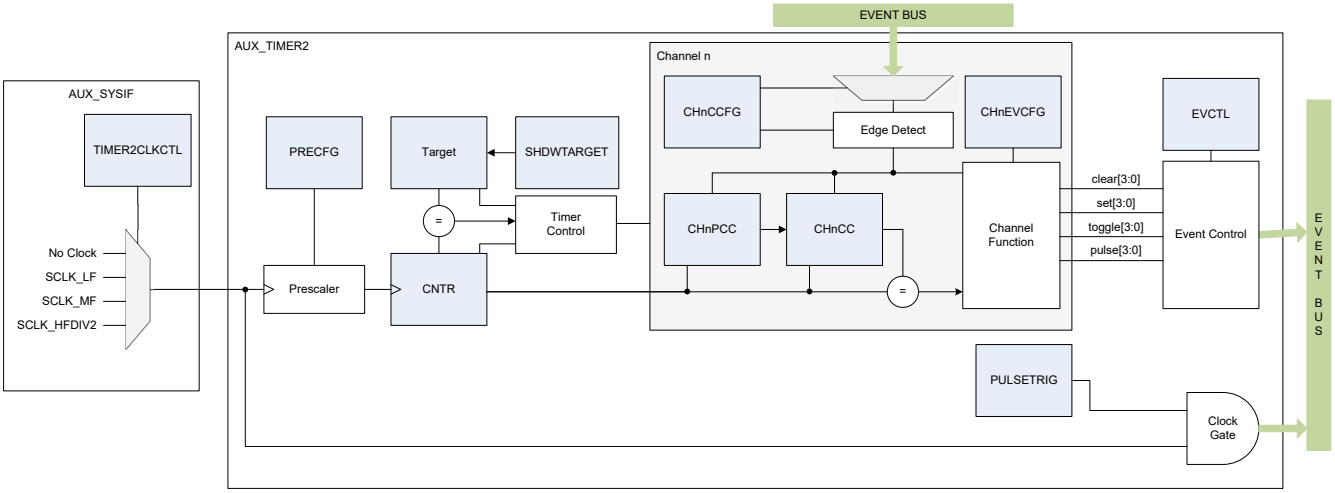


Figure 20-24. AUX_TIMER2 Block Diagram

20.4.7.2 Functional Description

20.4.7.2.1 Clock Source

The Timer2 clock runs independently of AUX operational mode. There are four possible sources for Timer2 clock:

- NO_CLOCK (default)
- SCLK_LF
- SCLK_MF
- SCLK_HFDIV2 – SCLK_HF / 2

Figure 20-24 shows that the clock source configuration is located within the AUX_SYSIF, which handles AUX domain clock and power management. Access to Timer2 is only possible when a clock is selected. To select a clock use the sequence that follows:

1. Wait until AUX_SYSIF:TIMER2CLKSWITCH.RDY = 1.
2. Set AUX_SYSIF:TIMER2CLKCTL.SRC to the appropriate value.
3. Wait until AUX_SYSIF:TIMER2CLKSWITCH.RDY = 1.

   Clock switch is complete, and AUX Timer2 receives the selected clock.

The AUX operational mode must be set to Active to select SCLK_HFDIV2. Only change operational mode when AUX_SYSIF:TIMER2CLKSWITCH.RDY = 1. There is no similar requirement when switching to other clock sources.

20.4.7.2.2 Clock Prescaler

The prescaler optionally divides the Timer2 clock. The divided clock determines the:

- Counter clock
- Channel event synchronization clock
- Event control clock
  - Channels update event output on this clock.

AUX_TIMER2:PRECFG.CLKDIV sets the division. Division ranges from 1 to 256. Registers are accessed at the Timer2 clock.

20.4.7.2.3 Counter

The value written to AUX_TIMER2:CTL.MODE determines the counter mode as follows:

- UP_ONCE: The timer counts from 0 to the selected target. The timer then becomes disabled.
- UP_PER: The timer counts from 0 to the selected target, repeatedly.
- UPDWN_PER: The timer counts from 0 to the selected target and decrements back to 0, repeatedly.

AUX_TIMER2:CTL.TARGET_EN selects either:

- Static counter target of 65535
- User configurable target value as given by AUX_TIMER2:TARGET

The target must be user configurable to allow shadow updates through the AUX_TIMER2:SHDWTARGET register.

20.4.7.2.4 Event Outputs

Timer2 has four event outputs:

- AUX_TIMER2_EV0 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0
- AUX_TIMER2_EV1 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1
- AUX_TIMER2_EV2 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2
- AUX_TIMER2_EV3 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3

The user can set and clear events manually by writing AUX_TIMER2:EVCTL. Manual update of an event output takes priority over automatic channel updates of the same event. Listed in decreasing order of priority, each event output can:

1. Clear
2. Set
3. Toggle
4. Pulse

   The event output is high for two counter clock periods, then goes low.

An event output may receive update requests from several channels at the same time. In this case, the event output is updated according to the previously stated priority list.

20.4.7.2.5 Channel Actions

Each channel implements 15 different channel actions. They are categorized as one-shot and continuous:

- A one-shot channel action performs a function only once before the timer disables the channel.
- A continuous channel action performs a function until the user disables the channel.

Table 20-30 lists the 15 channel actions.

Table 20-30. Channel Actions

Index	Channel Action	Category
0	Disable channel	One-shot
1	Set on capture, and then disable channel	One-shot
2	Clear on zero, toggle on compare, and then disable channel	One-shot
3	Set on zero, toggle on compare, and then disable channel	One-shot
4	Clear on compare, and then disable channel	One-shot
5	Set on compare, and then disable channel	One-shot
6	Toggle on compare, and then disable channel	One-shot
7	Pulse on compare, and then disable channel	One-shot
8	Period and pulse width measurement	Continuous
9	Set on capture repeatedly	Continuous
10	Clear on zero, toggle on compare repeatedly	Continuous
11	Set on zero, toggle on compare repeatedly	Continuous
12	Clear on compare repeatedly	Continuous
13	Set on compare repeatedly	Continuous
14	Toggle on compare repeatedly	Continuous
15	Pulse on compare repeatedly	Continuous

After configuration, the channel requests updates of enabled event outputs according to the channel action description in Table 20-30. There are three channel actions that require further description.

20.4.7.2.5.1 Period and Pulse Width Measurement

This channel action continuously captures period and pulse width of the signal selected by CHnCCFG.CAPT_SRC relative to the signal edge given by CHnCCFG.EDGE. The channel requests to set enabled events when CHnCC.VALUE contains signal period and CHnPCC.VALUE contains signal pulse width.

The channel function synchronizes the timer counter to the selected signal edge of the incoming signal. Hence:

- The counter restarts regularly, so other channel actions must be chosen with this in mind.
- The channels configured for this channel action cannot perform measurements simultaneously. The measurements are done in a time-interleaved manner.

For further description, see AUX_TIMER2:CHnEVCFG.CCACT in Section 20.8.7.

The timer measures signal period and pulse width of two different signals *A and B*. In the following example, it is assumed that both signals have periods less than the counter range. Hence, time-out detection as described in the register documentation is not required. Configure as follows:

- Channel 0:
  - AUX_TIMER2:CH0EVCFG. CCACT = PER_PULSE_WIDTH_MEAS
  - AUX_TIMER2:CH0EVCFG. EV0_GEN = 1
  - AUX_TIMER2:CH0CCFG.CAPT_SRC = Signal A
  - AUX_TIMER2:CH0CCFG.EDGE = RISING
- Channel 1:
  - AUX_TIMER2:CH1EVCFG. CCACT = PER_PULSE_WIDTH_MEAS
  - AUX_TIMER2:CH1EVCFG. EV1_GEN = 1
  - AUX_TIMER2:CH1CCFG.CAPT_SRC = Signal B
  - AUX_TIMER2:CH1CCFG.EDGE = FALLING
- Timer:
  - CTL.MODE = UP_PER

Figure 20-25 shows how the timer counter first synchronizes to signal A. Channel 0 then captures the high phase of signal A into CH0PCC at time t_0. Finally, the period of signal A is captured in CH0CC at time t_1. At the same time, Channel 0 sets the event output 0 high, and the timer counter starts to synchronize to signal B. Channel 1 then captures the low phase of signal B into CH1PCC at time t_2. Finally, the period of signal B is captured in CH1CC at time t_3. At the same time, channel 1 sets the event output 1 high, and the timer counter starts to synchronize to signal A. The sequence then repeats itself until it is stopped by the user.

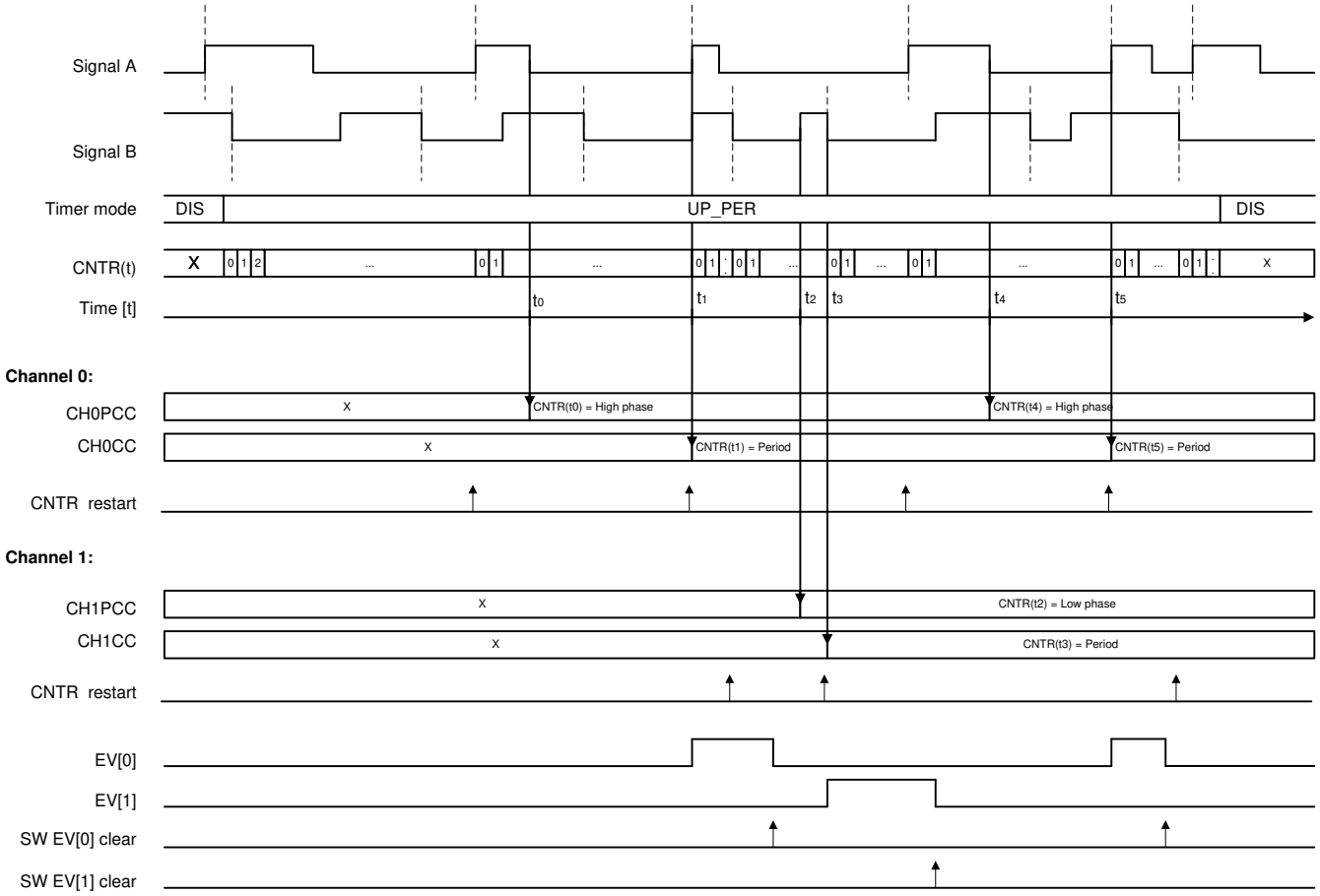


Figure 20-25. Period Pulse Width Measurement

20.4.7.2.5.2 Clear on Zero, Toggle on Compare Repeatedly

This channel action continuously:

- Clear enabled output events when AUX_TIMER2:CNTR = 0
- Toggle enabled output events when AUX_TIMER2:CNTR = CHnCC

The channel generates center-aligned PWM waveform when AUX_TIMER2:CTL.MODE = UPDWN_PER.

The channel copies a new value written in AUX_TIMER2:CHnPCC to AUX_TIMER2:CHnCC when AUX_TIMER2:CNTR becomes 0. This action prevents jitter on the edges of the generated PWM signal. Similarly, the timer copies a new value written in AUX_TIMER2:SHDWTARGET to AUX_TIMER2:TARGET when AUX_TIMER2:CNTR becomes 0. This action avoids period-jitter in PWM applications with time-varying periods.

For further description, see the description of the AUX_TIMER2:CHnEVCFG.CCACT register, Section 20.8.7.

The following example illustrates center-aligned PWM generation by channel 0 (see Figure 20-26). The waveform is synthesized on event output 0. The timer period is kept static, and the target value is set to half the period. Configure as follows:

- Channel 0:
  - AUX_TIMER2:CH0EVCFG.CCACT = CLR_ON_0_TGL_ON_CMP
  - AUX_TIMER2:CH0EVCFG.EV0_GEN = 1
  - AUX_TIMER2:CH0CC = C0
- Timer:
  - TARGET = PERIOD / 2
  - CTL.MODE = UPDWN_PER

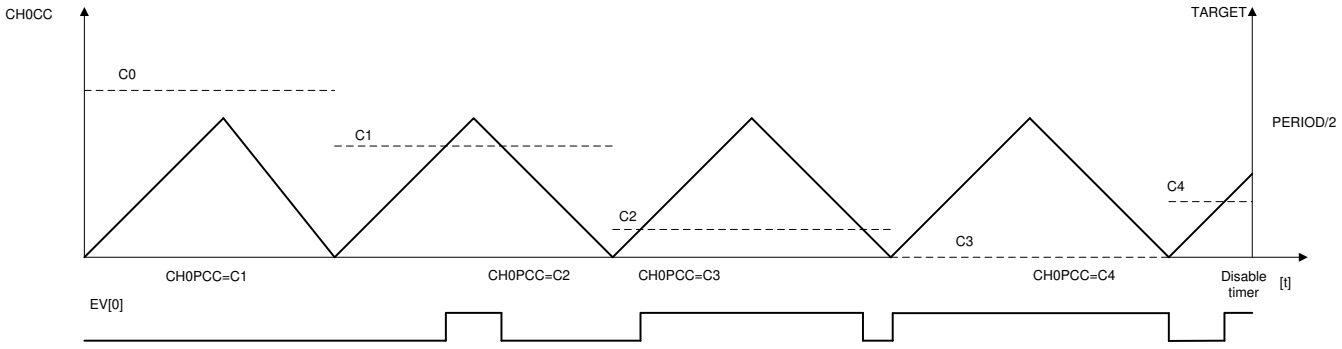


Figure 20-26. Center-Aligned PWM

The duty-cycle of the generated PWM waveform is controlled by AUX_TIMER2:CH0PCC updates. Waveform generation on event output 0 continues until stopped by the user.

20.4.7.2.5.3 Set on Zero, Toggle on Compare Repeatedly

This channel action continuously does the following:

- Set enabled output events when AUX_TIMER2:CNTR = 0
- Toggle enabled output events when AUX_TIMER2:CNTR = CHnCC

The channel generates an edge-aligned PWM waveform when AUX_TIMER2:CTL.MODE = UP_PER.

The channel copies a new value written in AUX_TIMER2:CHnPCC to AUX_TIMER2:CHnCC when AUX_TIMER2:CNTR becomes 0. This prevents jitter on the edges of the generated PWM signal. Similarly, the timer copies a new value written in AUX_TIMER2:SHDWTARGET to AUX_TIMER2:TARGET when AUX_TIMER2:CNTR becomes 0. This avoids period-jitter in PWM applications with time-varying period.

For further description, see the description of the AUX_TIMER2:CHnEVCFG.CCACT, Section 20.8.7.

The following example illustrates edge-aligned PWM generation by channel 0 (see Figure 20-27). The waveform is synthesized on event output 0. The timer period is kept static, and the target value is set to period minus 1. Configure as follows:

- Channel 0:
  - AUX_TIMER2:CH0EVCFG.CCACT = SET_ON_0_TGL_ON_CMP
  - AUX_TIMER2:CH0EVCFG.EV0_GEN = 1
  - AUX_TIMER2:CH0CC = C0
- Timer:
  - TARGET = PERIOD minus 1
  - CTL.MODE = UP_PER

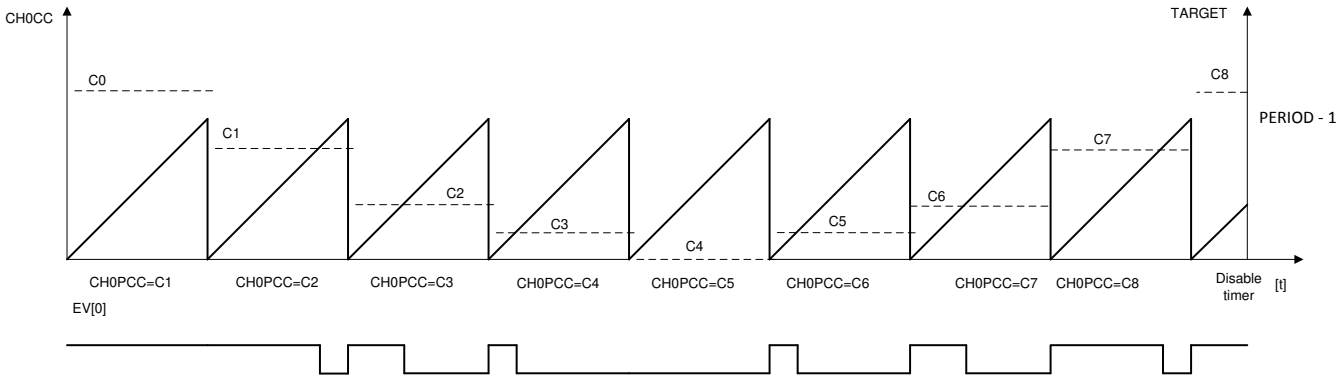


Figure 20-27. Edge-Aligned PWM

The duty-cycle of the generated PWM waveform is controlled by AUX_TIMER2:CH0PCC updates. Waveform generation on event output 0 continues until stopped by the user.

20.4.7.2.6 Asynchronous Bus Bridge

The AUX Timer2 is asynchronous to the AUX clock. It is accessed over an asynchronous bus bridge. The master side is clocked by the AUX bus clock, and the slave side is clocked by the Timer2 clock. The bridge has a single write buffer. An ongoing bus transfer stalls a new access, until the former completes.

To minimize access latency, TI recommends setting the Timer2 clock frequency to AUX clock frequency or higher during configuration.

20.5 Analog Peripheral Modules

20.5.1 Overview

Table 20-31 lists the analog peripherals in the AUX domain. The AUX Analog Interface (AUX_ANAIF) holds digital control and data interfaces to the ADC, ISRC, and DAC peripherals. The operational clock rates for these interfaces are listed in Table 20-31. See Section 20.2 for description about operational rates. The COMPA and COMPB modules are entirely controlled and configured through AUX_ADI4.

Analog peripherals in the AUX domain can be shared between the Sensor Controller and the System CPU. If this is required, it is recommended to use the hardware semaphores in AUX to arbitrate access to the peripherals. For the same reason it is recommended that the System CPU and the Sensor Controller leave the peripherals in reset state after usage. See Table 20-21 for how TI software assigns resources to the semaphores.

Table 20-31. Analog Peripherals

Analog Peripheral	Direct Digital Interface	Operational Clock Rate	
		BUS	SCE
ADC	AUX_ANAIF:ADCx	X	
ISRC	AUX_ANAIF:ISRCCTL	X	
DAC	AUX_ANAIF:DACx	(X)	X
COMPA	–	–	
COMPB	–	–	
AUX_ADI4	–	X	

20.5.1.1 ADI Control-Configuration

AUX_ADI4, which is listed in Table 20-31, is not really an analog peripheral. As shown in Figure 20-1, both the System CPU and the Sensor Controller access ADI_4_AUX through the AUX_ADI4 module to control and configure the analog AUX peripherals. The AUX_ADI4 module supports four different types of access to provide efficiency because it connects to the analog domain over a multicycle bus interface. The access types are:

- Direct
- Set
- Clear
- Masked

The Sensor Controller detects less access latency when bus clock rate is higher than the SCE clock rate.

If the System CPU application requires access to this module, the application must use the TI provided DriverLib API functions. Sensor Controller Studio API functions includes access when necessary and the user does not have to consider this module when developing code.

20.5.1.2 Block Diagram

Figure 20-28 shows the AUX analog block diagram.

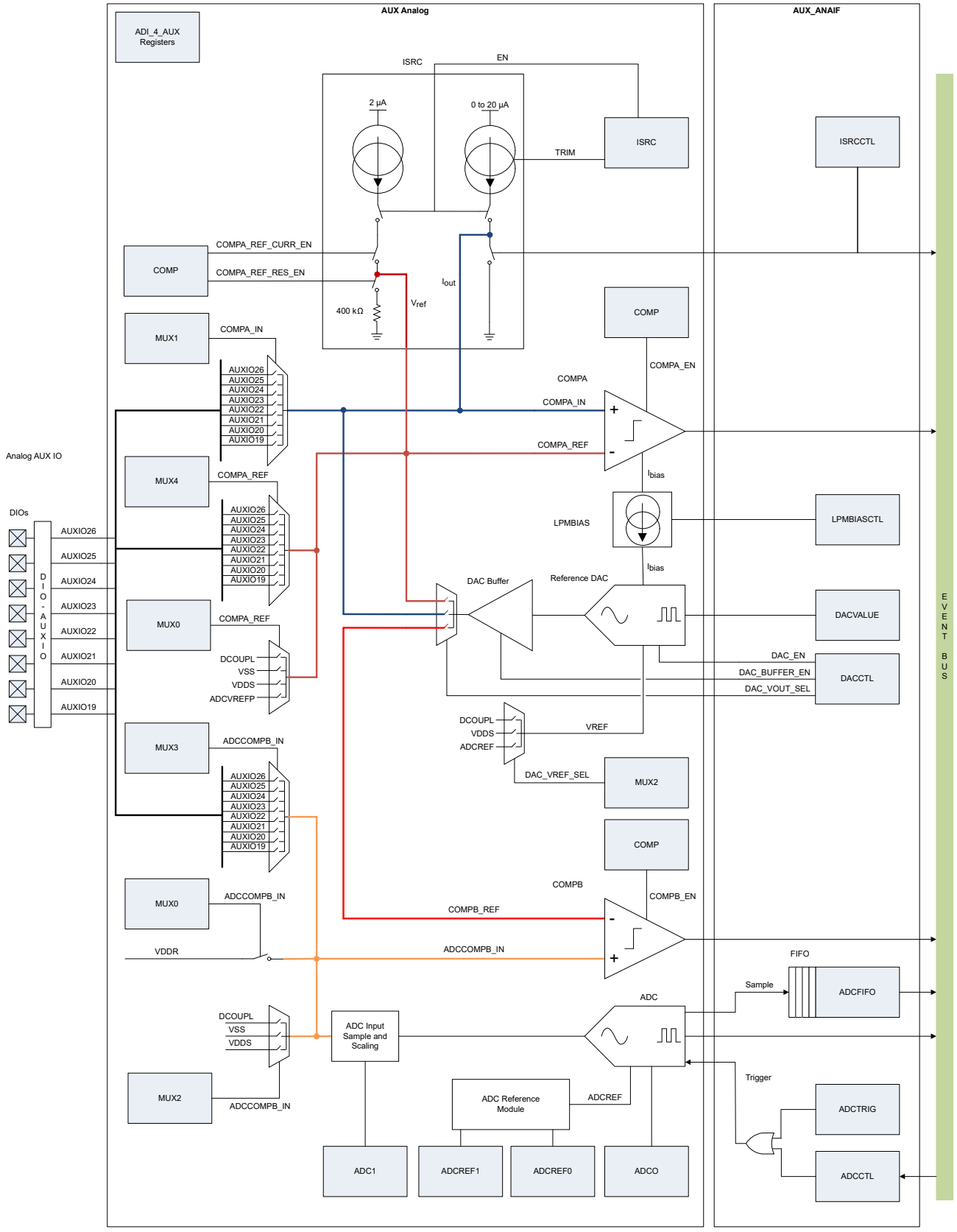


Figure 20-28. AUX Analog Block Diagram

20.5.2 Analog-to-Digital Converter (ADC)

20.5.2.1 Introduction

The AUX ADC peripheral has the following features:

- 12-bit general purpose successive-approximation type (SAR) ADC
- Samples analog-capable AUX I/O pins or internal chip voltages
- Sample rates up to 200 ksamples/s
- Manual or event-driven ADC trigger to start ADC sampling or conversion dependent of ADC operation mode:
  - ADC trigger starts sampling followed by conversion
  - Continuous sampling, ADC trigger starts conversion
- Configurable sampling time
- Fixed or relative ADC reference
- Four-element ADC sample FIFO with event interface

Figure 20-29 shows how the ADC connects to analog nodes and the digital control provided by AUX_ANAIF and ADI_4_AUX registers.

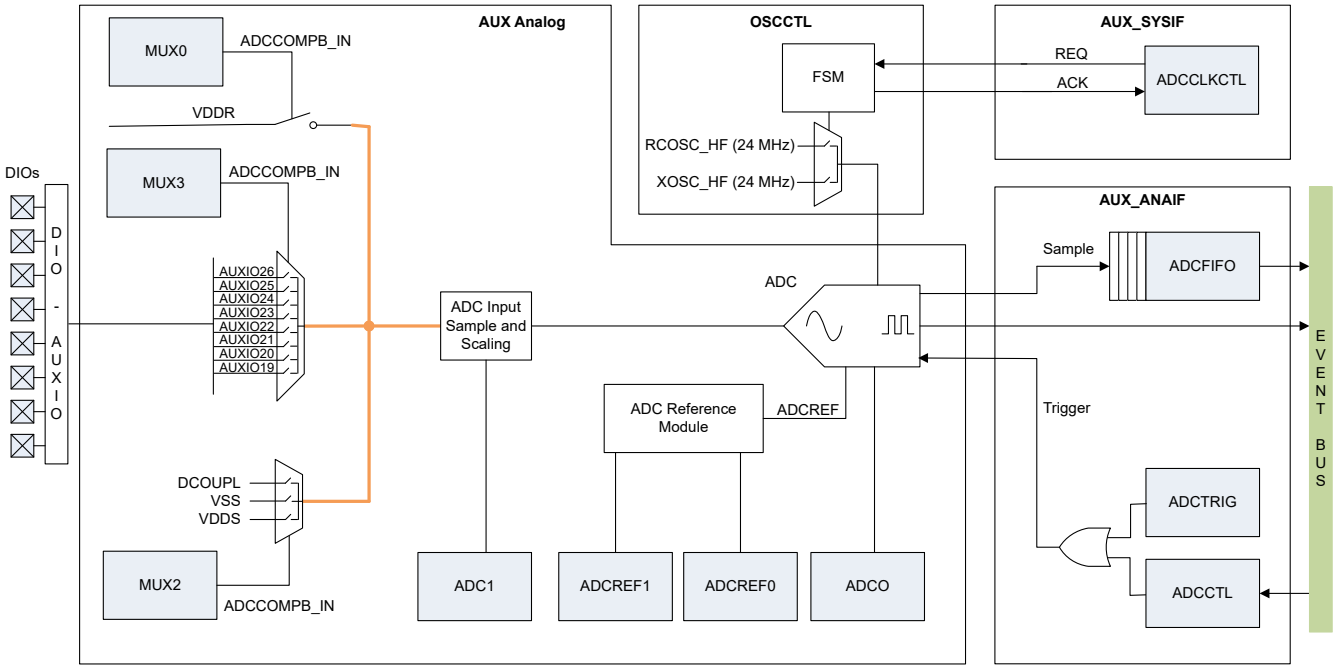


Figure 20-29. ADC Block Diagram

20.5.2.2 Functional Description

20.5.2.2.1 Input Selection and Scaling

The AUX analog-capable I/Os, AUXIO19 to AUXIO26, and some internal chip voltages can connect to the ADC input.

By default, the input is scaled down internally by a factor of 1408 / 4095 before it is used in the ADC. It is possible to disable down-scaling of the ADC input for increased resolution at the cost of reduced input range.

CAUTION

Disabling down-scaling changes the maximum input rating of the ADC input. Higher voltages can damage the ADC. See the device-specific data sheet.

The user must ensure that the ADC input has only one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.2.2.2 Reference Selection

Configuration of the ADC reference module must only happen while ADI_4_AUX.ADC0.RESET_N = 0. The ADC reference module must be enabled to perform ADC sample and conversion. The module provides the ADC reference voltage selected by ADI_4_AUX:ADCREF0.SRC, which can be a fixed voltage or a voltage relative to VDDS.

It is possible to turn off the ADC reference module to save power in synchronous ADC sample mode, as described in Section 20.5.2.2.3, which requires that the configurable sample period is longer than the ADC reference module power-up time. For a description, see ADI_4_AUX.ADCREF0.REF_ON_IDLE in Section 20.8.1.

Set ADI_4_AUX.ADCREF0.EN to enable the ADC reference module.

20.5.2.2.3 ADC Sample Mode

Configuration of sample mode and sample duration must only happen while ADI_4_AUX.ADC0.RESET_N = 0. The ADC input is a switched capacitor stage where the input voltage is sampled and then held before the conversion is done. The ADC operates in either synchronous or asynchronous mode, as described in the following:

- In synchronous mode, the ADC trigger enables sampling of the input for a programmable amount of time followed by conversion. This mode allows proper sampling of high-impedance sources. ADI_4_AUX:ADC0.SMPL_CYCLE_EXP sets the sampling duration.
- In asynchronous mode, the ADC continuously samples until the ADC trigger stops sampling and starts conversion. This mode allows jitter-free sampling at the expense of increased current consumption.

ADI_4_AUX.ADC0.SMPL_MODE selects the sample mode.

20.5.2.2.4 ADC Clock Source

The ADC core uses an internal 24 MHz clock for sampling and conversion. This internal clock source is required to enable the ADC core clock to use the ADC. For a description, see AUX_SYSIF:ADCCLKCTL in Section 20.8.9.

Once the ADC clock is requested, the clock source follows from Table 20-32.

Table 20-32. ADC Clock Source

RCOSC_HF	XOSC_HF	ADC Clock Source
Off	Off	RCOSC_HF
Off	On	XOSC_HF
On	Off	RCOSC_HF
On	On	XOSC_HF

For accurate, low-jitter sampling in asynchronous mode, software must ensure that SCLK_HF is sourced from the 24 MHz crystal.

Note

When the ADC clock is enabled, the system cannot go into standby or shutdown mode because the power controller requests a high-frequency clock source.

Finally, the user must set ADI_4_AUX:ADC0.EN to enable the analog ADC core and enable reaction to ADC triggers.

20.5.2.2.5 ADC Trigger

Configure the ADC start trigger with the sequence that follows:

1. Event-driven: AUX_ANAIF:ADCCTL selects an event source and polarity that will trigger the ADC
2. Manual: AUX_ANAIF:ADCCTL.START_SRC to NO_EVENT to manually trigger the ADC

The ADC sample mode, described in Section 20.5.2.2.3, controls the action when an ADC trigger occurs.

20.5.2.2.6 Sample FIFO

The ADC connects to a FIFO that can hold up to four ADC samples. Read AUX_ANAIF:ADCFIFO to get the oldest sample. It is possible to write dummy samples to the FIFO for debug purposes. The FIFO samples are not compensated for offset and gain errors in the ADC. The user must perform the compensation manually using the factory configuration data stored during chip production, see Section 11.4. The System CPU can use the AUX_ADC TI-DriverLib module to accomplish this task.

AUX_ANAIF:ADCFIFOSTAT provides FIFO status flags such as overflow, underflow, and utilization. All FIFO status flags and the ADC-conversion-done event connect to the AUX event bus.

To recover from an overflow or underflow condition, the user must flush the FIFO and re-enable the digital ADC interface, as described in AUX_ANAIF:ADCCTL.CMD (see Section 20.8.8).

Note

When debugging the software, showing the AUX_ANAIF_ADCFIFO register causes JTAG to read the FIFO, which pops the sample from the FIFO, and consequently the software cannot read it.

20.5.2.2.7 µDMA Interface

The µDMA Channel 7 is dedicated to transfer ADC samples from the ADC FIFO. Use the sequence that follows to set up a µDMA transfer:

1. Configure and enable the µDMA interface in AUX_EVCTL:DMACTL
   - Burst or single requests
2. µDMA transfer trigger
   - AUX_ADC_FIFO_NOT_EMPTY
   - AUX_ADC_FIFO_ALMOST_FULL
3. Configure the block size in the µDMA

The AUX_ADC_IRQ event sets when the µDMA completes data block transfer. AUX_ADC_IRQ is mapped to System CPU interrupt line 32 (for further description, see EVENT:CPUIRQSEL32 in Section 5.8.2).

Note

This event will also set when the ADC FIFO either overflows or underflows, as indicated by AUX_ANAIF:ADCFIFOSTAT.

20.5.2.2.8 Resource Ownership and Usage

ADC usage requires application ownership. For semaphore allocation in TI software, see Section 20.4.3.3.

The System CPU can use the peripheral in Active and Idle mode (power modes defined in TI's Power Manager). The Sensor Controller must request Active AUX operational mode to use the peripheral.

20.5.3 Comparator A (COMPA)

20.5.3.1 Introduction

The AUX COMPA peripheral is a high-performance continuous-time comparator that features:

- Input from analog-capable AUX I/O pins
- Reference from analog-capable AUX I/O pins or internal chip voltages

For capacitive touch sensing, COMPA is used together with the ISRC and TDC peripherals. ISRC will then drive an internal reference voltage for COMPA, nominally 0.8 V, while charging the capacitance that is also connected to the COMPA input. The time from start of charging until the COMPA output goes high is measured using the TDC.

Pulse counting is another application that involves COMPA and the TDC. In this application, the TDC prescaler counts COMPA events.

Figure 20-30 shows how COMPA connects to analog nodes and analog peripherals, as well as the digital interfaces. Proper reference DAC control and connections are described in Section 20.1.

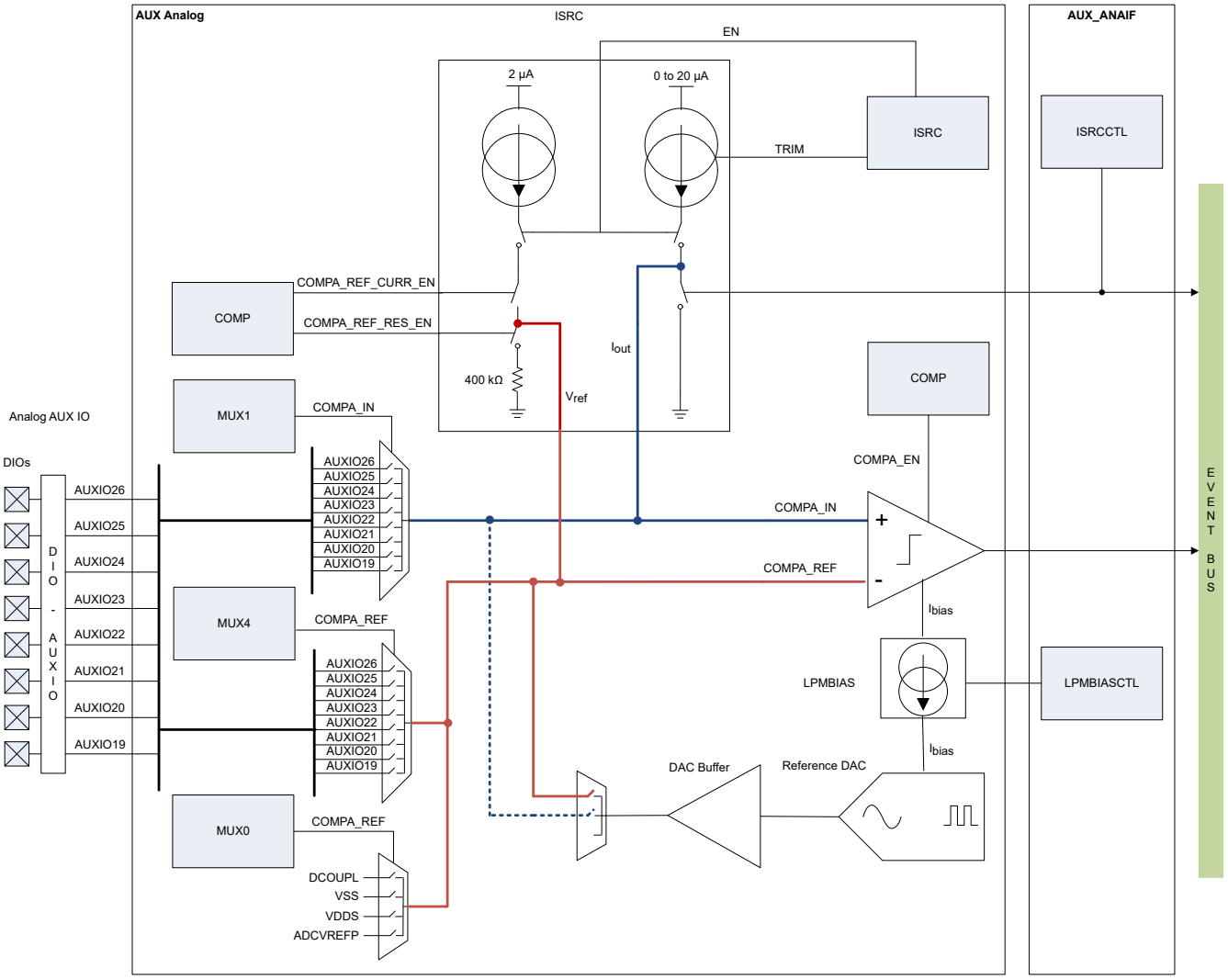


Figure 20-30. COMPA Block Diagram

20.5.3.2 Functional Description

20.5.3.2.1 Input Selection

The COMPA positive input can connect to:

- AUXIO19 through AUXIO26, which are analog-capable
- The programmable current output of the ISRC peripheral
- Reference DAC output

The reference DAC output connection is not intended for COMPA usage. Hence, the connection shows as dotted in Figure 20-30.

The user must ensure that the positive input of the COMPA peripheral has only one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.3.2.2 Reference Selection

The reference input for COMPA can connect to:

- AUXIO19 through AUXIO26, which are analog-capable
- The voltage reference output of the ISRC peripheral
- Reference DAC output

The user must ensure that the reference input for COMPA has only one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.3.2.3 LPM Bias and COMPA Enable

The COMPA peripheral requires bias current from the Low-Power Mode Bias (LPMBIAS) module when:

- The Sensor Controller owns and uses the peripheral in Low-Power and Power-Down operational modes
- The System CPU owns and uses the peripheral in Standby mode (power mode defined in TI's Power Manager).

Set AUX_ANAIF:LPMBIASCTL.EN to enable the LPM Bias module, then set ADI_4_AUX:COMP.COMPA_EN to enable COMPA.

20.5.3.2.4 Resource Ownership and Usage

COMPA usage requires application ownership of COMPA and the Reference DAC, if it generates the reference. For semaphore allocation in TI software, see Section 20.4.3.3.

The System CPU can use the peripheral in Active and Idle (power modes defined in TI's Power Manager). In this case, set AUX_SYSIF:EVSYNCRATE.AUX_COMPA_SYNC_RATE to BUS_RATE. To use the comparator in Standby mode, see Section 20.5.5.2.6.

Sensor Controller Studio provides the required API to use COMPA, and The Sensor Controller can use the peripheral in all AUX operational modes.

20.5.4 Comparator B (COMPB)

20.5.4.1 Introduction

The AUX COMPB peripheral is low-power clocked comparator that is updated at 32 kHz and features:

- Input from analog-capable AUX I/O pins
- Programmable reference

The primary application for COMPB is to continuously monitor slow signals and wake up the System CPU or the Sensor Controller.

Figure 20-31 shows how COMPB connects to analog nodes and analog peripherals, as well as the digital interfaces. Proper Reference DAC control and connections are described in Section 20.1.

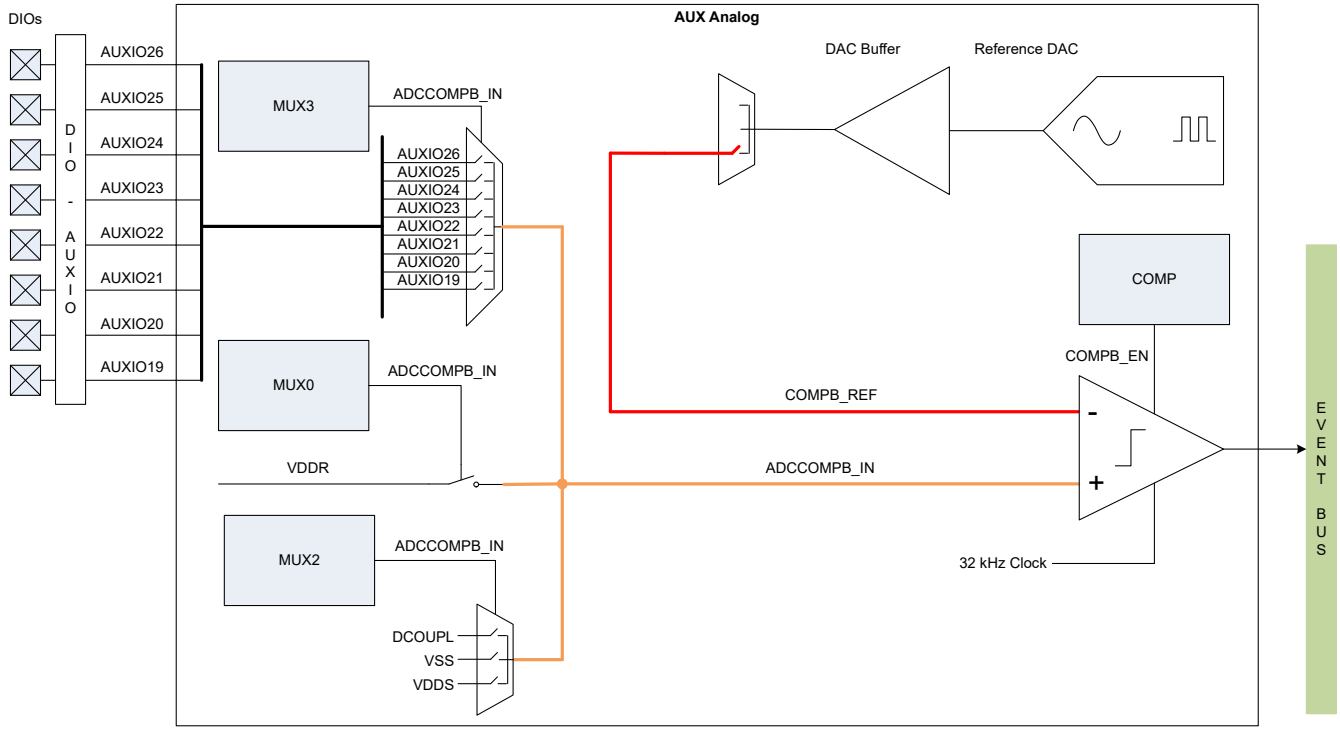


Figure 20-31. COMPB Block Diagram

20.5.4.2 Functional Description

20.5.4.2.1 Input Selection

The COMPB positive input can connect to:

- AUXIO19 through AUXIO26, which are analog-capable
- Internal power-supply voltages

The user must ensure that the COMPB positive input only has one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.4.2.2 Reference Selection

The Reference DAC provides a programmable COMPB voltage reference. Set AUX_ANAIF:DACCTL.DAC_VOUT_SEL to COMPB_REF to connect the analog nodes.

20.5.4.2.3 Resource Ownership and Usage

COMPB usage requires application ownership of both COMPB and the Reference DAC that generates the reference. For semaphore allocation in TI software, see Section 20.4.3.3.

The primary application for COMPB is to either wake up the System CPU or the Sensor Controller.

20.5.4.2.3.1 Sensor Controller Wakeup

Sensor Controller Studio provides the required API to use the COMPB event as Sensor Controller wakeup. The Sensor Controller can use the peripheral in all AUX operational modes.

20.5.4.2.3.2 System CPU Wakeup

When the Sensor Controller is not used by the application, the COMPB event can trigger System CPU wakeup in two ways:

- Direct wakeup: Select AUX_COMPB_ASYNC or AUX_COMPB_ASYNC_N as AON_EVENT:MCUWUSEL event source.
- Indirect wakeup: Select AUX_COMPB as AON_EVENT:MCUWUSEL event source, and configure the COMPB event polarity that sets the AUX_EVCTL:EVTOAONFLAGS.AUX_COMPB flag.

Direct wakeup infers minimum wake-up latency. It is however not possible to clear the COMPB event in AON_EVENT as is comes directly from the comparator.

Indirect wakeup infers latency that scales with the SCE clock rate, which depends on the AUX operational mode, as described in Section 20.2.1. The latency is given by a two-stage synchronizer and event flag capture and bounded upwards by three SCE clock periods. The System CPU can clear the event flag on wake up, and hence return to sleep while the COMPB event level that triggered wake up is active. For AUX event interface to the AON domain, see Section 20.6.6.

Because the Sensor Controller is not is not used in this scenario, the System CPU can configure COMPB event as wake-up event source for AUX. This assures low wake-up latency that scales with the 2 MHz clock. For a description, see Section 20.1 and the AUX_SYSIF:PROGWUnCFG registers in Section 20.8.9.

For indirect wakeup, the System CPU application must use the sequence that follows:

1. Set AUX_SYSIF:EVSYNCRATE.AUX_COMPB_SYNC_RATE to BUS_RATE.
2. Set AUX operational mode to Low-Power.
3. Configure the Reference DAC with valid Power-Down configuration (see Section 20.5.5.2.6).
4. Wait for Reference to settle.
5. Configure COMB event edge to trigger AUX wakeup.
6. Set AUX operational mode to Power-down, AUX_SYSIF:OPMODE.REQ = PDLP.
7. Upon wakeup, clear the wakeup flag as described in AUX_SYSIF:WUFLAGSCLR (see Section 20.8.9).

When the Sensor Controller is used by the application, it must provide the System CPU wakeup. Setup is done in Sensor Controller Studio.

20.5.5 Reference Digital-to-Analog Converter (DAC)

20.5.5.1 Introduction

The AUX Reference DAC peripheral has the following features:

- An 8-bit digital control word
- Configurable output range
- COMPA and COMPB reference generation
- Precharge analog-capable AUX I/O pins

The primary application for the Reference DAC is to provide configurable references for the comparators in various power modes. Hence, the Reference DAC is often used together with other analog peripherals.

Figure 20-32 shows how the Reference DAC connects to analog nodes and to the digital interfaces.

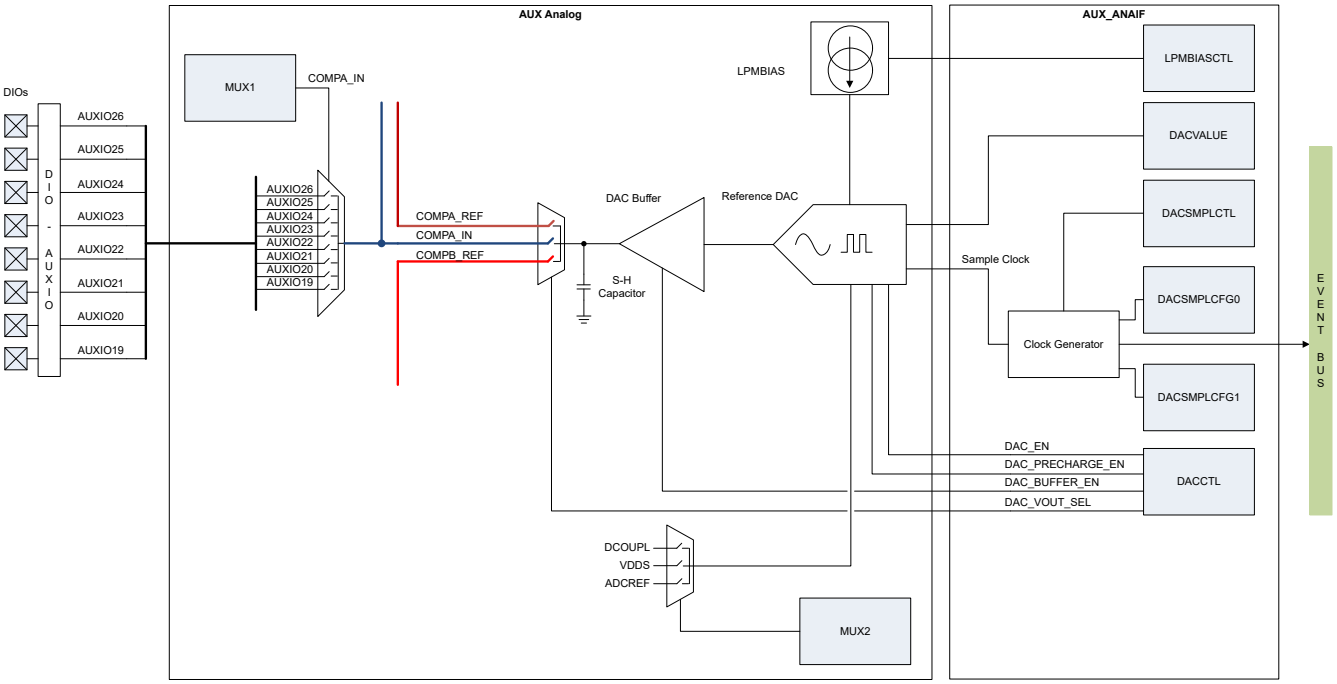


Figure 20-32. Reference DAC Block Diagram

20.5.5.2 Functional Description

The Reference DAC uses a configurable sample clock to translate an 8-bit digital code into an analog voltage stored in a Sample-and-Hold (S-H) capacitor. The S-H capacitor interfaces a selected load directly or through a buffer when driving higher loads.

20.5.5.2.1 Reference Selection

ADI_4_AUX:MUX2.DAC_VREF_SEL sets the DAC reference as:

- DCOUPL
- ADCREF
- VDDS

The ADI_4_AUX:MUX2.DAC_VREF_SEL register is only allowed to select one reference at a time. The reference gives the output voltage range.

20.5.5.2.2 Output Voltage Control and Range

The S-H capacitor voltage target is set by AUX_ANAIF:DACVALUE (see Section 20.8.8).

The Reference DAC transfer function exhibits nonlinearity.

Reference DAC calibration data is stored during chip production in the factory configuration, see FCFG1:DAC_CAL0 and FCFG1:DAC_CAL1 in Section 11.4.

20.5.5.2.3 Sample Clock

The DAC sample clock charges and maintains the DAC voltage stored in the S-H capacitor. The DAC sample clock waveform consists of a setup phase followed by a hold phase. The sample clock frequency is generally higher in the setup phase compared to the hold phase.

In the setup phase the sample-and-hold capacitor charges to the target voltage, while the hold phase maintains the S-H voltage. Power can be saved by reducing the sample clock frequency in the hold phase. Figure 20-33 shows the concept.

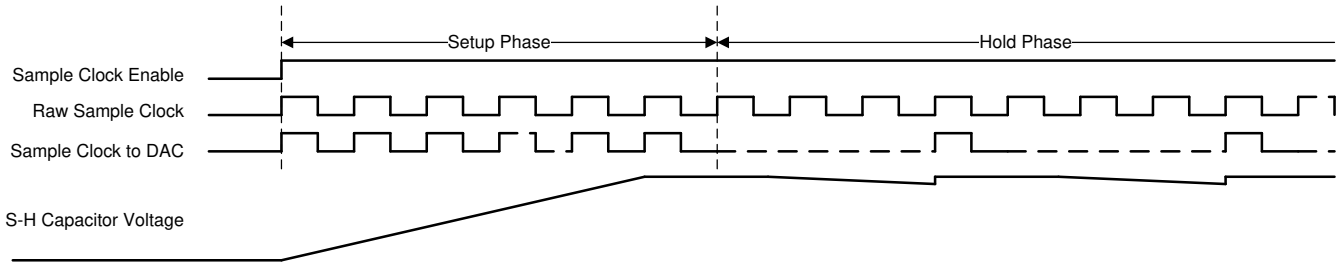


Figure 20-33. DAC Sample Clock Phases

The sample clock period is configured by AUX_ANAIF:DACSMPLCFG0 and AUX_ANAIF:DACSMPLCFG1 registers and enabled by the AUX_ANAIF:DACSMPLCTL register. The sample clock period scales with the peripheral clock frequency set by AUX_SYSIF:PEROPRATE.ANAIF_DAC_OP_RATE. Figure 20-34 shows how the sample clock period derives from the peripheral clock.

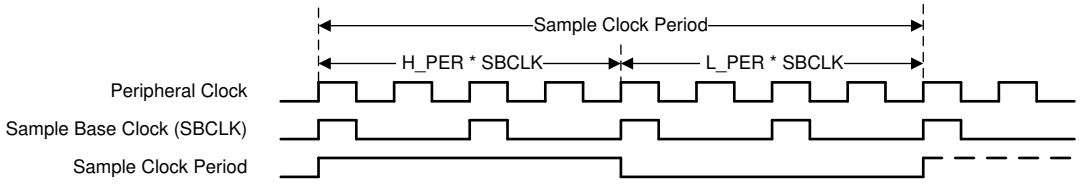


Figure 20-34. Sample Clock Period Configuration

AUX_ANAIF:DACSMPLCFG0 controls the clock division from the peripheral clock to the sample base clock. The AUX_ANAIF:DACSMPLCFG0 register is set to 2 in Figure 20-34.

AUX_ANAIF:DACSMPLCFG.H_PER and AUX_ANAIF:DACSMPLCFG.L_PER specify the duration of the high and low phase, respectively, of a single sample clock period in terms of the base clock period. They are both set to 2 periods in Figure 20-34.

A state machine can control the transition from setup phase to hold phase, or the user can control this manually.

20.5.5.2.3.1 Automatic Phase Control

Automatic phase transition requires that the setup phase needs less than 17 sample clock periods to charge the S-H capacitor to the target voltage. This is generally not the case when the Reference DAC drives an external load.

Configure the automatic phase control with the sequence as follows:

1. AUX_ANAIF:DACSMPLCFG1.SETUP_CNT specifies number of sample clock cycles in the setup phase.
2. AUX_ANAIF:DACSMPLCFG1.HOLD_INTERVAL specifies the number of inactive sample clock periods between each S-H refresh clock.

The transition sets the AUX_EVCTL:EVSTAT3.AUX_DAC_HOLD_ACTIVE event.

20.5.5.2.3.2 Manual Phase Control

The user must control the transition manually if the setup phase requires more than 16 sample clock periods to charge the S-H capacitor to the target value. Configure the manual phase control as follows:

1. Set AUX_ANAIF:DACSMPLCFG1.SETUP_CNT to 0.
2. Set AUX_ANAIF:DACSMPLCFG1.HOLD_INTERVAL initially to 0.

The user must update AUX_ANAIF:DACSMPLCFG1.HOLD_INTERVAL when the S-H capacitor target voltage has been reached.

Alternatively, the user can utilize the AUX_EVCTL:EVSTAT3.AUX_DAC_HOLD_ACTIVE and restart sample clock generation in a loop. This approach does not require any other hardware resources.

20.5.5.2.3.3 Operational Mode Dependency

There is only one allowed sample clock configuration in AUX Power-Down operational mode with AUX_SYSIF:OPMODEREQ.REQ equal to LPPD.

- Set all AUX_ANAIF:DACSMPLCFG1 fields to 0.

In the scenario, the user must enable the sample clock generation when AUX_SYSIF:OPMODEREQ.REQ = LP and then request LPPD.

Generation of the sample clock is not allowed when AUX_SYSIF:OPMODEREQ.REQ = PDA.

20.5.5.2.4 Output Selection

AUX_ANAIF:DACCTL.DAC_VOUT_SEL selects Reference DAC output connection:

- No Connection
- External load, COMPA_IN analog node
- COMPA reference input, COMPA_REF analog node
- COMPB reference input, COMPB_REF analog node

Selection of only one output connection at a time is allowed. The user must ensure that the connected analog node only has one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

The following subsections describe the different connections.

20.5.5.2.4.1 Buffer

The Reference DAC can interface the selected load through a buffer. The buffer must be enabled when the Reference DAC interfaces an external load. It is not recommended to interface internal loads with the buffer.

To enable the buffer, set AUX_ANAIF:DACCTL.DAC_BUFFER_EN to 1. To enable the buffer in LPM (Low Power Mode) also set AUX_ANAIF:LPMBIASCTL.EN to 1 to enable the bias module. The buffer cannot be used when AUX requests Power-Down operational mode.

20.5.5.2.4.2 External Load

The Reference DAC can drive an external load connected to the COMPA_IN analog node when the peripheral clock frequency is 2 MHz or 24 MHz. This is valid for:

- Active and Low-Power AUX operational mode
- Active and Idle mode (defined by TI's Power Manager)

The user must enable the buffer when the Reference DAC interfaces an external load.

Load support:

- Resistive loads > 10 MΩ
- From the AUX_DAC_Drvier specification: “If the DAC is operating at 2.0 V, the load is required to have a resistance >>10 MΩ or be purely capacitive . If this condition is not met, the CHP must be used to reduce the VOUT error.”
- Capacitive loads < 200 pF (always stable)
- Capacitive loads
- R+C

20.5.5.2.4.3 COMPA_REF

The Reference DAC can drive the COMPA_REF analog node in all power modes. In this case, the two COMPA_REF muxes shown in Figure 20-28 are disconnected from the COMPA_REF node. It is generally not recommended to use the buffer for this connection.

20.5.5.2.4.4 COMPB_REF

The Reference DAC can drive the COMPB_REF analog node in all power modes. It is generally not recommended to use the buffer for this connection.

20.5.5.2.5 LPM Bias

The Reference DAC requires bias current from the Low Power Mode Bias (LPMBIAS) module when:

- The Sensor Controller owns and uses the peripheral in Low-Power and Power-Down operational modes.
- The System CPU owns and uses the peripheral in Standby mode (defined by TI's Power Manager)

Set AUX_ANAIF:LPMBIASCTL.EN to enable the LPM Bias module, then set AUX_ANAIF:DACCTL.DAC_EN to enable the Reference DAC.

20.5.5.2.6 Resource Ownership and Usage

Reference DAC usage requires application ownership. For semaphore allocation in TI software, see Section 20.4.3.3.

Sensor Controller Studio provides the required API to use the Reference DAC, and The Sensor Controller can use the peripheral in all AUX operational modes.

The System CPU can use the peripheral in Active and Idle mode (power modes defined by TI's Power Manager). In this case, set AUX_SYSIF:PEROPRATE.ANAIF_DAC_OP_RATE to BUS_RATE.

The System CPU can use the peripheral in Standby mode (defined in TI's Power Manager) if the Sensor Controller is not used by the application. In this case, the System CPU must set AON_PMCTL.AUXSCECLK.PD_SRC to SCLK_LF and switch between Low-Power and Power-Down operational modes. The following sequence is required:

1. Set AUX_SYSIF:PEROPRATE.ANAIF_DAC_OP_RATE to SCE_RATE
2. Set AUX_SYSIF:OPMODEREQ to LP
3. Set the required AUX_ANAIF:DACVALUE
4. Configure AUX_ANAIF:DACCTL:
   a. Set DAC_PRECHARGE_EN to desired value
   b. Set DAC_VOUT_SEL to desired value
   c. Set AUX_ANAIF:DACCTL.DAC_BUFFER_EN to 0
5. Set AUX_ANAIF:LPMBIASCTL to 1
6. Wait 15 µs
7. Set AUX_ANAIF:DACSMPLCFG0.CLKDIV to 0
8. Configure AUX_ANAIF:DACSMPLCFG1:
   a. Set H_PER to 1
   b. Set L_PER to 0
   c. Set SETUP_CNT to 0
   d. Set HOLD_INTERVAL to 0
9. Set AUX_ANAIF:DACSMPLCTL to 1 to enable sample clock generation
10. Set AUX_ANAIF:DACSMPLCFG1.H_PER to 0
11. Wait for the required time to charge the S-H capacitor to the requested voltage
12. Set AUX_SYSIF:OPMODEREQ to PDLP
13. Enter sleep and wait for interrupt

20.5.6 Current Source (ISRC)

20.5.6.1 Introduction

The AUX Current Source (ISRC) module has the following features:

- Programmable current output: 0 to 20 µA
- Voltage reference output: nominally at 0.8 V

The primary application for the ISRC peripheral is capacitive touch sensing, where it is used together with the COMPA and TDC peripherals. In this application, ISRC drives an internal reference voltage for COMPA, nominally 0.8 V, while charging the capacitance connected to the COMPA input. The time from start of charging until the COMPA output goes high is measured using the TDC.

Figure 20-35 shows the ISRC analog connections and digital control. Proper Reference DAC control and connections are described in Section 20.1.

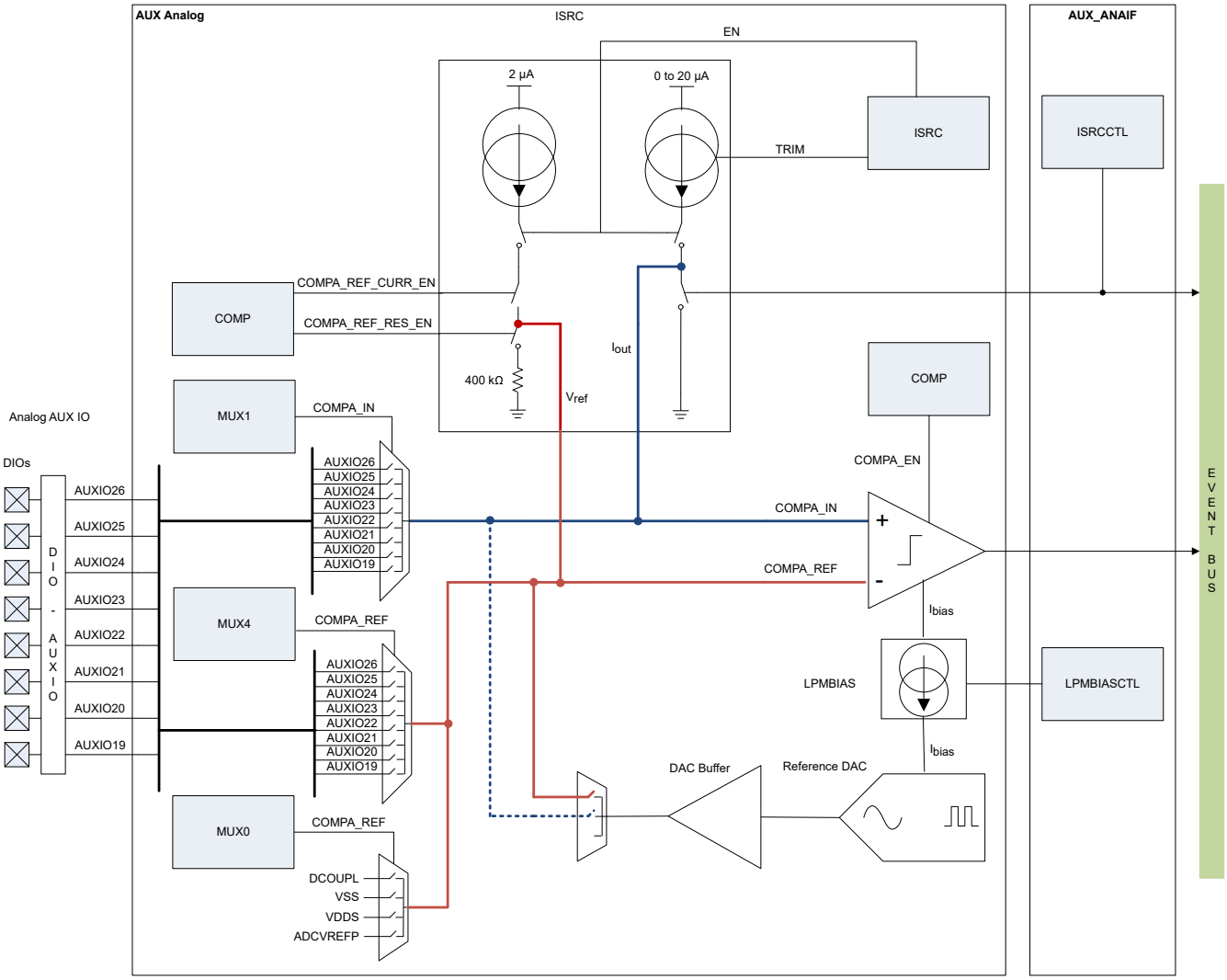


Figure 20-35. ISRC Block Diagram

20.5.6.2 Functional Description

20.5.6.2.1 Programmable Current

The programmable current output connects to the positive input of COMPA, which makes it possible to charge an external capacitance connected to an analog-capable AUX I/O. Physically, this output also connects to the reference DAC output. This connection is not intended for ISRC usage. Hence, the connection shows as a dotted line in Figure 20-35.

When ISRC is enabled, it drives the current configured by ADI_4_AUX:ISRC.TRIM onto the COMPA_IN analog node or to ground. The connection controlled by AUX_ANAIF:ISRCCTL.RESET_N.

The user must ensure that the COMPA positive input only has one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.6.2.2 Voltage Reference

The voltage reference output connects to the reference input of COMPA. The generated voltage reference is nominally at 0.8 V when using the internal 400 kΩ resistor. To generate and connect the reference voltage to COMPA reference input:

- Set ADI_4_AUX:COMP.COMPA_REF_RES_EN to enable the internal 400 kΩ resistor.
- Set ADI_4_AUX:COMP.COMPA_REF_CURR_EN to enable the 2 µA current.

It is also possible to generate other voltages by using an external resistor. To generate and connect the reference voltage to COMPA reference input in this case:

- Connect a resistor to an analog-capable AUX I/O.
- Connect that AUX I/O to COMPA reference input.
- Set ADI_4_AUX:COMP.COMPA_REF_CURR_EN to enable the 2 µA current.

The user must ensure that the COMPA reference input only has one driver at all times. To avoid shorting signals together internally, TI provides ROM-based functions that ensure a break-before-make switching of the internal muxes.

20.5.6.2.3 ISRC Enable

Set ADI_4_AUX:ISRC.EN to enable the module and to generate any current.

20.5.6.2.4 Temperature Dependency

The ISRC outputs are temperature dependent. The effect cancels out when ISRC both charges the COMPA positive input node and connects the internally generated 0.8 reference voltage to COMPA reference input.

20.5.6.2.5 Resource Ownership and Usage

ISRC usage requires application. For semaphore allocation in TI software, see Section 20.4.3.3.

The System CPU can use the peripheral in Active and Idle mode (power modes defined in TI's Power Manager). The Sensor Controller must request Active AUX operational mode to use the peripheral.

20.6 Event Routing and Usage

20.6.1 AUX Event Bus

The AUX domain event bus assembles events from:

- AUX peripherals
- AUX I/O pins
- Real-Time clock
- Battery Monitor and Temperature Sensor
- System Power and Clock Management

The event assembles into a 64-bit event bus that exists in two versions, one that is partly asynchronous and one that is fully synchronous to the AUX clock. The user can read the level of the synchronous events in the AUX_EVCTL:EVSTAT0 through AUX_EVCTL:EVSTAT3 registers.

20.6.1.1 Event Signals

Table 20-33 summarizes the event signals. For detailed event description, see the AUX_EVCTL:EVSTAT0 through AUX_EVCTL:EVSTAT3 registers in Section 20.8.3.

Table 20-33. AUX Event Bus

Index	Asynchronous Event Bus(1) (2)		Synchronous Event Bus(3) (4) (5)	
	Origin	Signal	SCE Rate	Bus Rate
[0:31]	DIO	AUXIO[0:31]	X	
32	AUX	MANUAL_EV	X	
33	System	AON_RTC_CH2	X	
34		AON_RTC_CH2_DLY	X	
35		AON_RTC_4KHZ	X	
36		AON_BATMON_BAT_UPD	X	
37		AON_BATMON_TEMP_UPD	X	
38		SCLK_LF	X	
39		PWR_DWN	X	
40		MCU_ACTIVE	X	
41		VDDR_RECHARGE	X	
42		ACLK_REF	X	
43		MCU_EV	X	
44		MCU_OBSMUX0	X	
45		MCU_OBSMUX1	X	
46	AUX	AUX_COMPA	X	(X)
47		AUX_COMPB	X	(X)
48		AUX_TIMER2_EV0	X	(X)
49		AUX_TIMER2_EV1	X	(X)
50		AUX_TIMER2_EV2	X	(X)
51		AUX_TIMER2_EV3	X	(X)
52		AUX_TIMER2_PULSE	X	(X)
53	AUX	AUX_TIMER1_EV	X	(X)
54		AUX_TIMER0_EV	X	(X)
55		AUX_TDC_DONE		X
56		AUX_ISRC_RESET_N		X
57		AUX_ADC_DONE		X
58		AUX_ADC_IRQ		X
59		AUX_ADC_FIFO_ALMOST_FULL		X
60		AUX_ADC_FIFO_NOT_EMPTY		X
61		AUX_SMPH_AUTOTAKE_DONE		X
62		DAC_HOLD_ACTIVE	X	(X)
63		TIMER2_CLKSWITCH_RDY		X

(1) Events not shaded have asynchronous origin referenced to the AUX clock.
(2) Events shaded in blue have synchronous origin referenced to the AUX clock.
(3) X = Default Rate
(4) (X) = Rate Override Capability
(5) These columns show the event update rate on the synchronous event bus.

The update rate is determined by either event synchronization or the operational rate of the synchronous peripheral that generates it.

Table 20-33 shows the following:

- Some events of asynchronous origin have configurable synchronization rate. The rate is configured by AUX_SYSIF:EVSYNCRATE.
- Some events of synchronous origin have configurable update rate. The rate is configured by AUX_SYSIF:PEROPRATE.

This configurability serves resource sharing between the System CPU and the Sensor Controller.

Because of synchronization there will always be a time delay between the asynchronous and the synchronous version of an event of asynchronous origin. The System CPU may require minimum synchronization latency, in which case the synchronization must happen at bus clock rate. A slow AUX SCE clock rate will then not affect the latency. On the other hand, when the SCE clock rate is less than the bus clock rate, any event synchronized at the bus clock rate will experience lower latency when the MCU domain is active. This may not be desirable for a Sensor Controller task or for Timer01. In this case, the event synchronization rate must equal SCE clock rate.

20.6.1.2 Event Subscribers

Table 20-34 lists the AUX event bus subscribers and to which version of the event bus they subscribe.

Table 20-34. AUX Event Subscribers

Subscribers	Asynchronous Event Bus	Synchronous Event Bus
Sensor Controller		X
Timer0 and Timer1		X
TDC State Machine	X	
TDC Prescaler	X	
TDC High-Speed Counter	X	
Timer2	X	
ADC Interface	X	
AUX Programmable Wakeup	X	

20.6.1.2.1 Event Detection

The subscribers detect events differently. In general, there is no assurance that an event will be long enough for detection.

20.6.1.2.1.1 Detection of Asynchronous Events

An asynchronous event is either detected by a synchronizer or the event is used as clock to set a detection flag, which then gets synchronized. Table 20-35 lists the detection schemes used for the subscribers.

Table 20-35. Asynchronous Event Detection

Subscribers	Detection Scheme
TDC Prescaler	Event as clock
TDC High-Speed Counter	Synchronizer
Timer 2	Synchronizer
ADC Interface	Event as clock
AUX Programmable Wakeup	Event as clock

Any event of asynchronous origin, such as AUX I/O or COMPA event, may be too short to allow proper synchronizer detection at a given clock frequency. Similarly, any Timer2 event may be too short to allow proper

synchronization to a slower AUX SCE clock. In general, the pulse width of an asynchronous event must be longer than the sample clock period to allow proper synchronizer detection.

To assure that the event subscriber catches the synchronized event, either of the following must be true:

- The operational clock rate of the event subscriber must be equal or higher than the synchronizer clock rate.
- The synchronized event duration must at least equal the effective clock period of the subscriber module.

When an asynchronous event is used as clock to set a detection flag, the tolerated event pulse width is a lot less than for synchronizers.

20.6.1.2.1.2 Detection of Synchronous Events

Similar requirement exist when the Sensor Controller and Timer01 uses events that have synchronous origin. If the operational rate of the event subscriber is lower than the operational rate of the event publisher, the subscriber may lose events. This happens when the event duration is shorter than the effective clock period of the subscriber module. The following consequences arise from this:

- As the ADC interface operates at SCLK_HFDIV2 clock rate, the Sensor Controller and the Timer01 operational rate must equal SCLK_HFDIV2 to subscribe to AUX_ADC_DONE event.
- If the Sensor Controller subscribes to AUX_TIMER0_EV or AUX_TIMER1_EV, then the timer operational rate must be equal or lower than the Sensor Controller operational rate.

20.6.2 Event Observation on External Pin

It is possible to observe the asynchronous event selected by AUX_EVCTL.EVOBSCFG on an external AUX I/O pin. To do so the user must configure the corresponding AUX I/O for peripheral mode output and set the AUX_EV_OBS event as data source.

This feature is useful for:

- Debug
- Setting the external pin level and simultaneously triggering an internal peripheral operation, such as TDC conversion

For AUX I/O configuration, see Section 20.4.2 and Figure 20-11.

20.6.3 Events From MCU Domain

The MCU event fabric defines the MCU_EV event at index 43 in Table 20-33. EVENT:AUXSEL0 configures the source for this event.

AUX_EVCTL:EVTOMCUFLAGS holds 16 event flags that connect to the MCU event fabric and System CPU. The event flag name is identical to the event on the synchronous event bus it derives from. Table 20-36 lists the event flags and their sensitivity type.

20.6.4 Events to MCU Domain

AUX_EVCTL:EVTOMCUFLAGS holds 16 event flags that connect to the MCU event fabric and System CPU. The event flag name is identical to the event on the synchronous event bus it derives from. Table 20-36 lists the event flags and their sensitivity type.

An event flag sets when the level or edge configured in AUX_EVCTL:EVTOMCUPOL occurs for that event. In order to clear a level-sensitive flag, the corresponding event must be cleared before the user clears the event flag. Write 1 to an event index in AUX_EVCTL:EVTOMCUFLAGSCLR to clear a level- or edge-sensitive flag.

Table 20-36. Events to MCU

Index	Event Flag Name	Sensitivity	Description
0	AUX_WU_EV	Level	AUX wakeup event(1)
1	AUX_COMPA	Edge	Comparator A event
2	AUX_COMPB	Edge	Comparator B event
3	AUX_TDC_DONE	Level	TDC conversion done or time-out
4	AUX_TIMER0_EV	Level	AUX Timer0 has reached its target value
5	AUX_TIMER1_EV	Level	AUX Timer1 has reached its target value
6	AUX_SMPH_AUTOTAKE_DONE	Level	A user-specified semaphore has been released
7	AUX_ADC_DONE	Level	ADC conversion is done
8	AUX_ADC_FIFO_ALMOST_FULL	Level	ADC FIFO almost full
9	MCU_OBSMUX0	Level	MCU observable event.
10	AUX_ADC_IRQ	Level	ADC IRQ(2)
11	AUX_TIMER2_EV0	Level	AUX Timer2 Event 0
12	AUX_TIMER2_EV1	Level	AUX Timer2 Event 1
13	AUX_TIMER2_EV2	Level	AUX Timer2 Event 2
14	AUX_TIMER2_EV3	Level	AUX Timer2 Event 3
15	AUX_TIMER2_PULSE	Level	AUX Timer2 single clock pulse

(1) Logical OR of the AUX wake-up events in the AUX_SYSIF:WUFLAGS register.
(2) ADC new sample available, FIFO underflow or overflow. If DMA is used: ADC DMA done, FIFO underflow or overflow.

The user can optionally enable the AUX_COMB event as a function of the 16 event flags in AUX_EVCTL:EVTOMCUFLAGS. The AUX_EVCTL:COMBEVTOMCUMASK configuration determines if a set event flag will set the AUX_COMB event. The AUX_COMB event is high as long as one or more of the event flags selected by AUX_EVCTL:COMBEVTOMCUMASK are set.

The MCU event fabric connects the 16 event flags and the AUX_COMB event as:

- DMA Triggers
- Radio Timer Capture
- System CPU Interrupts
- General-Purpose Timer Capture

See EVENT:* in Chapter 5 for all the possible connections.

20.6.5 Events From AON Domain

Events with index 32 through index 41 in Table 20-33 come from the AON domain. These events mainly serve two purposes:

- Wakeup: The events are used as AUX domain and Sensor Controller wake-up sources, either directly or down-divided by Timer01.
- Wait-Event: The Sensor Controller halts execution until a specific event level is reached.

Examples of wake-up usage are:

- Timer0 divides the AON_RTC_4KHZ event to generate AUX domain wakeup and Sensor Controller task execution every 10 ms.
- The AON_RTC_CH2 event wakes up the AUX domain. The AON_RTC_CH2_DLY event triggers Sensor Controller task execution.

Examples of wait-event usage are:

- A new value of AUX_SYSIF:BATMONBAT is ready when AON_BATMON_BAT_UPD is high.
- A new value of AUX_SYSIF:BATMONTEMP is ready when AON_BATMON_TEMP_UPD is high.

20.6.6 Events to AON Domain

AUX_EVCTL:EVTOAONFLAGS holds nine event flags that connect to the AON event fabric. The event flag name is identical to the event on the synchronous event bus it derives from. Table 20-37 lists the event flags and their sensitivity type.

Table 20-37. Events to AON

Index	Event Flag Name	Sensitivity	Description
0	SWEV0	NA	Maps directly to AUX_EVCTL:SWEVSET.SWEV0
1	SWEV1	NA	Maps directly to AUX_EVCTL:SWEVSET.SWEV1
2	SWEV2	NA	Maps directly to AUX_EVCTL:SWEVSET.SWEV2
3	AUX_COMPA	Edge	Comparator A event
4	AUX_COMPB	Edge	Comparator B event
5	AUX_ADC_DONE	Level	ADC conversion is done
6	AUX_TDC_DONE	Level	TDC conversion done or time-out
7	AUX_TIMER0_EV	Level	AUX Timer0 has reached its target value
8	AUX_TIMER1_EV	Level	AUX Timer1 has reached its target value

An event flag sets when the level or edge configured in AUX_EVCTL:EVTOAONPOL occurs for that event. The level of SWEV0 through SWEV2 event flags is controlled by register. To clear a level-sensitive flag, the corresponding event must be cleared before the user clears the event flag. Write 1 to an event index in AUX_EVCTL:EVTOAONFLAGSCLR to clear a level- or edge-sensitive flag.

The event flags SWEV0 and SWEV1 also connect directly to the System CPU.

The event flags are useful for:

- MCU domain wakeup (see AON_EVENT:MCUWUSEL in Section 5.8.1)
- Programmable Event to MCU (see AON_EVENT:EVTOMCUSEL in Section 5.8.1)
- RTC Capture Events (see AON_EVENT:RTCSEL in Section 5.8.1)
- System CPU interrupt:
  - SWEV0 maps to EVENT:CPUIRQSEL6
  - SWEV1 maps to EVENT:CPUIRQSEL13
- Software-defined handshake protocol

The AUX_TIMER2_EV* events are also routed directly to AON_EVENT without intermediate synchronization to the AUX clock.

20.6.7 µDMA Interface

The µDMA channel 7 is dedicated to transfer ADC samples from the ADC FIFO. For further description, see Section 20.5.2.2.7.

The System CPU and Sensor Controller can request µDMA transfer on channel 8 through AUX_EVCTL:DMASWREQ register. This requires properly setup of µDMA channel 8.

20.7 Sensor Controller Alias Register Space

Most peripheral addresses are aliased down to the lower 256 words (512 bytes) in the AUX memory space. Only the Sensor Controller can access the alias register space. The Sensor Controller uses an alias address whenever possible (see Table 20-38). Usage improves instruction execution by one SCE clock cycle compared to using the non-aliased 16-bit peripheral address.

Table 20-38. Alias Register Mapping

Register Bank	Register Name	Original Address	Alias Address
AUX_SPIM	SPIMCFG	0x400C 1000 + 0x00	0
AUX_SPIM	MISOCFG	0x400C 1000 + 0x04	1
AUX_SPIM	MOSICTL	0x400C 1000 + 0x08	2
AUX_SPIM	TX8	0x400C 1000 + 0x0C	3
AUX_SPIM	TX16	0x400C 1000 + 0x10	4
AUX_SPIM	RX8	0x400C 1000 + 0x14	5
AUX_SPIM	RX16	0x400C 1000 + 0x18	6
AUX_SPIM	SCLKIDLE	0x400C 1000 + 0x1C	7
AUX_SPIM	DATAIDLE	0x400C 1000 + 0x20	8
AUX_MAC	OP0S	0x400C 2000 + 0x00	9
AUX_MAC	OP0U	0x400C 2000 + 0x04	10
AUX_MAC	OP1SMUL	0x400C 2000 + 0x08	11
AUX_MAC	OP1UMUL	0x400C 2000 + 0x0C	12
AUX_MAC	OP1SMAC	0x400C 2000 + 0x10	13
AUX_MAC	OP1UMAC	0x400C 2000 + 0x14	14
AUX_MAC	OP1SADD16	0x400C 2000 + 0x18	15
AUX_MAC	OP1UADD16	0x400C 2000 + 0x1C	16
AUX_MAC	OP1SADD32	0x400C 2000 + 0x20	17
AUX_MAC	OP1UADD32	0x400C 2000 + 0x24	18
AUX_MAC	CLZ	0x400C 2000 + 0x28	19
AUX_MAC	CLS	0x400C 2000 + 0x2C	20
AUX_MAC	ACCSHIFT	0x400C 2000 + 0x30	21
AUX_MAC	ACCRESET	0x400C 2000 + 0x34	22
AUX_MAC	ACC15_0	0x400C 2000 + 0x38	23
AUX_MAC	ACC31_16	0x400C 2000 + 0x78	24
AUX_MAC	ACC39_32	0x400C 2000 + 0x9C	25
AUX_TIMER2	CTL	0x400C 3000 + 0x00	26
AUX_TIMER2	TARGET	0x400C 3000 + 0x04	27
AUX_TIMER2	SHDWTARGET	0x400C 3000 + 0x08	28
AUX_TIMER2	CNTR	0x400C 3000 + 0x0C	29
AUX_TIMER2	PRECFG	0x400C 3000 + 0x10	30
AUX_TIMER2	EVCTL	0x400C 3000 + 0x14	31
AUX_TIMER2	PULSETRIG	0x400C 3000 + 0x18	32
AUX_TIMER2	CH0EVCFG	0x400C 3000 + 0x80	33
AUX_TIMER2	CH1EVCFG	0x400C 3000 + 0x90	34
AUX_TIMER2	CH2EVCFG	0x400C 3000 + 0xA0	35
AUX_TIMER2	CH3EVCFG	0x400C 3000 + 0xB0	36
AUX_TIMER2	CH0CCFG	0x400C 3000 + 0x84	37
AUX_TIMER2	CH1CCFG	0x400C 3000 + 0x94	38
AUX_TIMER2	CH2CCFG	0x400C 3000 + 0xA4	39

Table 20-38. Alias Register Mapping (continued)

Register Bank	Register Name	Original Address	Alias Address
AUX_TIMER2	CH3CCFG	0x400C 3000 + 0xB4	40
AUX_TIMER2	CH0PCC	0x400C 3000 + 0x88	41
AUX_TIMER2	CH1PCC	0x400C 3000 + 0x98	42
AUX_TIMER2	CH2PCC	0x400C 3000 + 0xA8	43
AUX_TIMER2	CH3PCC	0x400C 3000 + 0xB8	44
AUX_TIMER2	CH0CC	0x400C 3000 + 0x8C	45
AUX_TIMER2	CH1CC	0x400C 3000 + 0x9C	46
AUX_TIMER2	CH2CC	0x400C 3000 + 0xAC	47
AUX_TIMER2	CH3CC	0x400C 3000 + 0xBC	48
AUX_TDC	CTL	0x400C 4000 + 0x00	49
AUX_TDC	STAT	0x400C 4000 + 0x04	50
AUX_TDC	RESULT_LO	0x400C 4000 + 0x08	51
AUX_TDC	RESULT_HI	0x400C 4000 + 0x0A	52
AUX_TDC	SATCFG	0x400C 4000 + 0x0C	53
AUX_TDC	TRIGSRC	0x400C 4000 + 0x10	54
AUX_TDC	TRIGCNT	0x400C 4000 + 0x14	55
AUX_TDC	TRIGCNTLOAD	0x400C 4000 + 0x18	56
AUX_TDC	TRIGCNTCFG	0x400C 4000 + 0x1C	57
AUX_TDC	PRECTL	0x400C 4000 + 0x20	58
AUX_TDC	PRECNTR	0x400C 4000 + 0x24	59
AUX_EVCTL	SCEWEVCFG0	0x400C 5000 + 0x10	60
AUX_EVCTL	SCEWEVCFG1	0x400C 5000 + 0x14	61
AUX_EVCTL	DMACTL	0x400C 5000 + 0x18	62
AUX_EVCTL	DMASWREQ	0x400C 5000 + 0x1C	63
AUX_EVCTL	SWEVSET	0x400C 5000 + 0x20	64
AUX_EVCTL	EVTOAONFLAGS	0x400C 5000 + 0x24	65
AUX_EVCTL	EVTOAONPOL	0x400C 5000 + 0x28	66
AUX_EVCTL	EVTOAONFLAGSCLR	0x400C 5000 + 0x2C	67
AUX_EVCTL	EVTOMCUFLAGS	0x400C 5000 + 0x30	68
AUX_EVCTL	EVTOMCUPOL	0x400C 5000 + 0x34	69
AUX_EVCTL	EVTOMCUFLAGSCLR	0x400C 5000 + 0x38	70
AUX_EVCTL	COMBEVTOMCUMASK	0x400C 5000 + 0x3C	71
AUX_EVCTL	EVOBSCFG	0x400C 5000 + 0x40	72
AUX_EVCTL	PROGDLY	0x400C 5000 + 0x44	73
AUX_EVCTL	MANUAL	0x400C 5000 + 0x48	74
AUX_EVCTL	EVSTAT0L	0x400C 5000 + 0x4C	75
AUX_EVCTL	EVSTAT0H	0x400C 5000 + 0x50	76
AUX_EVCTL	EVSTAT1L	0x400C 5000 + 0x54	77
AUX_EVCTL	EVSTAT1H	0x400C 5000 + 0x58	78
AUX_EVCTL	EVSTAT2L	0x400C 5000 + 0x5C	79
AUX_EVCTL	EVSTAT2H	0x400C 5000 + 0x60	80
AUX_EVCTL	EVSTAT3L	0x400C 5000 + 0x64	81
AUX_EVCTL	EVSTAT3H	0x400C 5000 + 0x68	82
AUX_SYSIF	OPMODEREQ	0x400C 6000 + 0x00	83
AUX_SYSIF	OPMODEACK	0x400C 6000 + 0x04	84

Table 20-38. Alias Register Mapping (continued)

Register Bank	Register Name	Original Address	Alias Address
AUX_SYSIF	PROGWU0CFG	0x400C 6000 + 0x08	85
AUX_SYSIF	PROGWU1CFG	0x400C 6000 + 0x0C	86
AUX_SYSIF	PROGWU2CFG	0x400C 6000 + 0x10	87
AUX_SYSIF	PROGWU3CFG	0x400C 6000 + 0x14	88
AUX_SYSIF	SWWUTRIG	0x400C 6000 + 0x18	89
AUX_SYSIF	WUFLAGS	0x400C 6000 + 0x1C	90
AUX_SYSIF	WUFLAGSCLR	0x400C 6000 + 0x20	91
AUX_SYSIF	WUGATE	0x400C 6000 + 0x24	92
AUX_SYSIF	VECCFG0	0x400C 6000 + 0x28	93
AUX_SYSIF	VECCFG1	0x400C 6000 + 0x2C	94
AUX_SYSIF	VECCFG2	0x400C 6000 + 0x30	95
AUX_SYSIF	VECCFG3	0x400C 6000 + 0x34	96
AUX_SYSIF	VECCFG4	0x400C 6000 + 0x38	97
AUX_SYSIF	VECCFG5	0x400C 6000 + 0x3C	98
AUX_SYSIF	VECCFG6	0x400C 6000 + 0x40	99
AUX_SYSIF	VECCFG7	0x400C 6000 + 0x44	100
AUX_SYSIF	EVSYNCRATE	0x400C 6000 + 0x48	101
AUX_SYSIF	PEROPRATE	0x400C 6000 + 0x4C	102
AUX_SYSIF	ADCCLKCTL	0x400C 6000 + 0x50	103
AUX_SYSIF	TDCCLKCTL	0x400C 6000 + 0x54	104
AUX_SYSIF	TDCREFCLKCTL	0x400C 6000 + 0x58	105
AUX_SYSIF	TIMER2CLKCTL	0x400C 6000 + 0x5C	106
AUX_SYSIF	TIMER2CLKSTAT	0x400C 6000 + 0x60	107
AUX_SYSIF	TIMER2CLKSWITCH	0x400C 6000 + 0x64	108
AUX_SYSIF	TIMER2DBGCTL	0x400C 6000 + 0x68	109
AUX_SYSIF	BGAPREQ	0x400C 6000 + 0x6C	110
AUX_SYSIF	CLKSHIFTDET	0x400C 6000 + 0x70	111
AUX_SYSIF	RECHARGETRIG	0x400C 6000 + 0x74	112
AUX_SYSIF	RECHARGEDET	0x400C 6000 + 0x78	113
AUX_SYSIF	RTCSUBSECINC0	0x400C 6000 + 0x7C	114
AUX_SYSIF	RTCSUBSECINC1	0x400C 6000 + 0x80	115
AUX_SYSIF	RTCSUBSECINCCTL	0x400C 6000 + 0x84	116
AUX_SYSIF	RTCSEC	0x400C 6000 + 0x88	117
AUX_SYSIF	RTCSUBSEC	0x400C 6000 + 0x8C	118
AUX_SYSIF	RTCEVCLR	0x400C 6000 + 0x90	119
AUX_SYSIF	BATMONBAT	0x400C 6000 + 0x94	120
AUX_SYSIF	BATMONTEMP	0x400C 6000 + 0x9C	121
AUX_SYSIF	TIMERHALT	0x400C 6000 + 0xA0	122
AUX_SYSIF	OPMODESTAT	0x400C 6000 + 0xAC	123
AUX_SYSIF	TIMER2BRIDGE	0x400C 6000 + 0xB0	124
AUX_TIMER01	T0CFG	0x400C 7000 + 0x00	125
AUX_TIMER01	T1CFG	0x400C 7000 + 0x10	126
AUX_TIMER01	T0CTL	0x400C 7000 + 0x04	127
AUX_TIMER01	T1CTL	0x400C 7000 + 0x14	128
AUX_TIMER01	T0TARGET	0x400C 7000 + 0x08	129

Table 20-38. Alias Register Mapping (continued)

Register Bank	Register Name	Original Address	Alias Address
AUX_TIMER01	T1TARGET	0x400C 7000 + 0x18	130
AUX_TIMER01	T0CNTR	0x400C 7000 + 0x0C	131
AUX_TIMER01	T1CNTR	0x400C 7000 + 0x1C	132
AUX_SMPH	SMPH0	0x400C 8000 + 0x00	133
AUX_SMPH	SMPH1	0x400C 8000 + 0x04	134
AUX_SMPH	SMPH2	0x400C 8000 + 0x08	135
AUX_SMPH	SMPH3	0x400C 8000 + 0x0C	136
AUX_SMPH	SMPH4	0x400C 8000 + 0x10	137
AUX_SMPH	SMPH5	0x400C 8000 + 0x14	138
AUX_SMPH	SMPH6	0x400C 8000 + 0x18	139
AUX_SMPH	SMPH7	0x400C 8000 + 0x1C	140
AUX_SMPH	AUTOTAKE	0x400C 8000 + 0x20	141
AUX_ANAIF	ADCCTL	0x400C 9000 + 0x10	142
AUX_ANAIF	ADCFIFOSTAT	0x400C 9000 + 0x14	143
AUX_ANAIF	ADCFIFO	0x400C 9000 + 0x18	144
AUX_ANAIF	ADCTRIG	0x400C 9000 + 0x1C	145
AUX_ANAIF	ISRCCTL	0x400C 9000 + 0x20	146
AUX_ANAIF	DACCTL	0x400C 9000 + 0x30	147
AUX_ANAIF	LPMBIASCTL	0x400C 9000 + 0x34	148
AUX_ANAIF	DACSMPLCTL	0x400C 9000 + 0x38	149
AUX_ANAIF	DACSMPLCFG0	0x400C 9000 + 0x3C	150
AUX_ANAIF	DACSMPLCFG1	0x400C 9000 + 0x40	151
AUX_ANAIF	DACVALUE	0x400C 9000 + 0x44	152
AUX_ANAIF	DACSTAT	0x400C 9000 + 0x48	153
AUX_ADI4	SET01	0x400C B000 + 0x10	154
AUX_ADI4	SET23	0x400C B000 + 0x12	155
AUX_ADI4	SET45	0x400C B000 + 0x14	156
AUX_ADI4	SET67	0x400C B000 + 0x16	157
AUX_ADI4	SET89	0x400C B000 + 0x18	158
AUX_ADI4	SET1011	0x400C B000 + 0x1A	159
AUX_ADI4	SET1213	0x400C B000 + 0x1C	160
AUX_ADI4	SET1415	0x400C B000 + 0x1E	161
AUX_ADI4	CLR01	0x400C B000 + 0x20	162
AUX_ADI4	CLR23	0x400C B000 + 0x22	163
AUX_ADI4	CLR45	0x400C B000 + 0x24	164
AUX_ADI4	CLR67	0x400C B000 + 0x26	165
AUX_ADI4	CLR89	0x400C B000 + 0x28	166
AUX_ADI4	CLR1011	0x400C B000 + 0x2A	167
AUX_ADI4	CLR1213	0x400C B000 + 0x2C	168
AUX_ADI4	CLR1415	0x400C B000 + 0x2E	169
AUX_AIODIO0	IOMODEL	0x400C C000 + 0x40	170
AUX_AIODIO0	IOMODEH	0x400C C000 + 0x44	171
AUX_AIODIO1	IOMODEL	0x400C D000 + 0x40	172
AUX_AIODIO1	IOMODEH	0x400C D000 + 0x44	173
AUX_AIODIO2	IOMODEL	0x400C E000 + 0x40	174

Table 20-38. Alias Register Mapping (continued)

Register Bank	Register Name	Original Address	Alias Address
AUX_AIODIO2	IOMODEH	0x400C E000 + 0x44	175
AUX_AIODIO3	IOMODEL	0x400C F000 + 0x40	176
AUX_AIODIO3	IOMODEH	0x400C F000 + 0x44	177
AUX_AIODIO0	GPIODIE	0x400C C000 + 0x04	178
AUX_AIODIO1	GPIODIE	0x400C D000 + 0x04	179
AUX_AIODIO2	GPIODIE	0x400C E000 + 0x04	180
AUX_AIODIO3	GPIODIE	0x400C F000 + 0x04	181
AUX_AIODIO0	IOPOE	0x400C C000 + 0x08	182
AUX_AIODIO1	IOPOE	0x400C D000 + 0x08	183
AUX_AIODIO2	IOPOE	0x400C E000 + 0x08	184
AUX_AIODIO3	IOPOE	0x400C F000 + 0x08	185
AUX_AIODIO0	GPIODOUT	0x400C C000 + 0x0C	186
AUX_AIODIO1	GPIODOUT	0x400C D000 + 0x0C	187
AUX_AIODIO2	GPIODOUT	0x400C E000 + 0x0C	188
AUX_AIODIO3	GPIODOUT	0x400C F000 + 0x0C	189
AUX_AIODIO0	GPIODIN	0x400C C000 + 0x10	190
AUX_AIODIO1	GPIODIN	0x400C D000 + 0x10	191
AUX_AIODIO2	GPIODIN	0x400C E000 + 0x10	192
AUX_AIODIO3	GPIODIN	0x400C F000 + 0x10	193
AUX_AIODIO0	GPIODOUTSET	0x400C C000 + 0x14	194
AUX_AIODIO1	GPIODOUTSET	0x400C D000 + 0x14	195
AUX_AIODIO2	GPIODOUTSET	0x400C E000 + 0x14	196
AUX_AIODIO3	GPIODOUTSET	0x400C F000 + 0x14	197
AUX_AIODIO0	GPIODOUTCLR	0x400C C000 + 0x18	198
AUX_AIODIO1	GPIODOUTCLR	0x400C D000 + 0x18	199
AUX_AIODIO2	GPIODOUTCLR	0x400C E000 + 0x18	200
AUX_AIODIO3	GPIODOUTCLR	0x400C F000 + 0x18	201
AUX_AIODIO0	GPIODOUTTGL	0x400C C000 + 0x1C	202
AUX_AIODIO1	GPIODOUTTGL	0x400C D000 + 0x1C	203
AUX_AIODIO2	GPIODOUTTGL	0x400C E000 + 0x1C	204
AUX_AIODIO3	GPIODOUTTGL	0x400C F000 + 0x1C	205
AUX_AIODIO0	IO0PSEL	0x400C C000 + 0x20	206
AUX_AIODIO0	IO1PSEL	0x400C C000 + 0x24	207
AUX_AIODIO0	IO2PSEL	0x400C C000 + 0x28	208
AUX_AIODIO0	IO3PSEL	0x400C C000 + 0x2C	209
AUX_AIODIO0	IO4PSEL	0x400C C000 + 0x30	210
AUX_AIODIO0	IO5PSEL	0x400C C000 + 0x34	211
AUX_AIODIO0	IO6PSEL	0x400C C000 + 0x38	212
AUX_AIODIO0	IO7PSEL	0x400C C000 + 0x3C	213
AUX_AIODIO1	IO0PSEL	0x400C D000 + 0x20	214
AUX_AIODIO1	IO1PSEL	0x400C D000 + 0x24	215
AUX_AIODIO1	IO2PSEL	0x400C D000 + 0x28	216
AUX_AIODIO1	IO3PSEL	0x400C D000 + 0x2C	217
AUX_AIODIO1	IO4PSEL	0x400C D000 + 0x30	218
AUX_AIODIO1	IO5PSEL	0x400C D000 + 0x34	219

Table 20-38. Alias Register Mapping (continued)

Register Bank	Register Name	Original Address	Alias Address
AUX_AIODIO1	IO6PSEL	0x400C D000 + 0x38	220
AUX_AIODIO1	IO7PSEL	0x400C D000 + 0x3C	221
AUX_AIODIO2	IO0PSEL	0x400C E000 + 0x20	222
AUX_AIODIO2	IO1PSEL	0x400C E000 + 0x24	223
AUX_AIODIO2	IO2PSEL	0x400C E000 + 0x28	224
AUX_AIODIO2	IO3PSEL	0x400C E000 + 0x2C	225
AUX_AIODIO2	IO4PSEL	0x400C E000 + 0x30	226
AUX_AIODIO2	IO5PSEL	0x400C E000 + 0x34	227
AUX_AIODIO2	IO6PSEL	0x400C E000 + 0x38	228
AUX_AIODIO2	IO7PSEL	0x400C E000 + 0x3C	229
AUX_AIODIO3	IO0PSEL	0x400C F000 + 0x20	230
AUX_AIODIO3	IO1PSEL	0x400C F000 + 0x24	231
AUX_AIODIO3	IO2PSEL	0x400C F000 + 0x28	232
AUX_AIODIO3	IO3PSEL	0x400C F000 + 0x2C	233
AUX_AIODIO3	IO4PSEL	0x400C F000 + 0x30	234
AUX_AIODIO3	IO5PSEL	0x400C F000 + 0x34	235
AUX_AIODIO3	IO6PSEL	0x400C F000 + 0x38	236
AUX_AIODIO3	IO7PSEL	0x400C F000 + 0x3C	237

20.8 AUX Domain Sensor Controller and Peripherals Registers

Table 20-39 lists the registers in the AUX subsystem. See Chapter 3 for the memory-map.

Table 20-39. AUX Domain Sensor Controller and Peripherals Registers

Module	Name	Section
AUX_ADI4	ADI Master	Section 20.8.1
AUX_AIODIO	Analog Digital IOs	Section 20.8.2
AUX_EVCTL	Event Controller	Section 20.8.3
AUX_SMPH	Semaphores	Section 20.8.4
AUX_TDC	Time-to-Digital Converter	Section 20.8.5
AUX_TIMER01	Timer01	Section 20.8.6
AUX_TIMER2	Timer2	Section 20.8.7
AUX_ANAIF	Analog Interface	Section 20.8.8
AUX_SYSIF	System Interface	Section 20.8.9
AUX_SPIM	SPI Master	Section 20.8.10
AUX_MAC	Math Accelerator	Section 20.8.11

20.8.1 ADI_4_AUX Registers

Table 20-40 lists the memory-mapped registers for the ADI_4_AUX registers. All register offset addresses not listed in Table 20-40 should be considered as reserved locations and the register contents should not be modified.

Table 20-40. ADI_4_AUX Registers

Offset	Acronym	Register Name	Section
0h	MUX0	Internal	Section 20.8.1.1
1h	MUX1	Internal	Section 20.8.1.2
2h	MUX2	Internal	Section 20.8.1.3
3h	MUX3	Internal	Section 20.8.1.4
4h	ISRC	Current Source	Section 20.8.1.5
5h	COMP	Comparator	Section 20.8.1.6
7h	MUX4	Internal	Section 20.8.1.7
8h	ADC0	ADC Control 0	Section 20.8.1.8
9h	ADC1	ADC Control 1	Section 20.8.1.9
Ah	ADCREF0	ADC Reference 0	Section 20.8.1.10
Bh	ADCREF1	ADC Reference 1	Section 20.8.1.11
Eh	LPMBIAS	Internal	Section 20.8.1.12

Complex bit access types are encoded to fit into small table cells. Table 20-41 shows the codes that are used for access types in this section.

Table 20-41. ADI_4_AUX Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.1.1 MUX0 Register (Offset = 0h) [Reset = 00h]

MUX0 is shown in Table 20-42.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-42. MUX0 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0h	Reserved
6	ADCCOMPB_IN	R/W	0h	Internal. Only to be used through TI provided API.
5-4	RESERVED	R	0h	Reserved
3-0	COMPA_REF	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.2 MUX1 Register (Offset = 1h) [Reset = 00h]

MUX1 is shown in Table 20-43.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-43. MUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	COMPA_IN	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.3 MUX2 Register (Offset = 2h) [Reset = 00h]

MUX2 is shown in Table 20-44.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-44. MUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-3	ADCCOMPB_IN	R/W	0h	Internal. Only to be used through TI provided API.
2-0	DAC_VREF_SEL	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.4 MUX3 Register (Offset = 3h) [Reset = 00h]

MUX3 is shown in Table 20-45.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-45. MUX3 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	ADCCOMPB_IN	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.5 ISRC Register (Offset = 4h) [Reset = 00h]

ISRC is shown in Table 20-46.

Return to the Summary Table.

Current Source
Strength and trim control for current source. Only to be used through TI provided API.

Table 20-46. ISRC Register Field Descriptions

Bit	Field	Type	Reset	Description
7-2	TRIM	R/W	0h	Adjust current from current source. Output currents may be combined to get desired total current. 0h = No current connected 1h = 0P25U : 0.25 uA 2h = 0P5U : 0.5 uA 4h = 1P0U : 1.0 uA 8h = 2P0U : 2.0 uA 10h = 4P5U : 4.5 uA 20h = 11P75U : 11.75 uA
1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Current source enable

20.8.1.6 COMP Register (Offset = 5h) [Reset = 00h]

COMP is shown in Table 20-47.

Return to the Summary Table.

Comparator
Control COMPA and COMPB comparators. Only to be used through TI provided API.

Table 20-47. COMP Register Field Descriptions

Bit	Field	Type	Reset	Description
7	COMPA_REF_RES_EN	R/W	0h	Enables 400kΩ resistance from COMPA reference node to ground. Used with COMPA_REF_CURR_EN to generate voltage reference for cap-sense.
6	COMPA_REF_CURR_EN	R/W	0h	Enables 2uA IPTAT current from ISRC to COMPA reference node. Requires ISRC.EN = 1. Used with COMPA_REF_RES_EN to generate voltage reference for cap-sense.
5-3	LPM_BIAS_WIDTH_TRIM	R/W	0h	Internal. Only to be used through TI provided API.
2	COMPB_EN	R/W	0h	COMPB enable
1	RESERVED	R	0h	Reserved
0	COMPA_EN	R/W	0h	COMPA enable

20.8.1.7 MUX4 Register (Offset = 7h) [Reset = 00h]

MUX4 is shown in Table 20-48.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-48. MUX4 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-0	COMPA_REF	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.8 ADC0 Register (Offset = 8h) [Reset = 00h]

ADC0 is shown in Table 20-49.

Return to the Summary Table.

ADC Control 0
ADC Sample Control. Only to be used through TI provided API.

Table 20-49. ADC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	SMPL_MODE	R/W	0h	ADC Sampling mode: 0: Synchronous mode 1: Asynchronous mode The ADC does a sample-and-hold before conversion. In synchronous mode the sampling starts when the ADC clock detects a rising edge on the trigger signal. Jitter/uncertainty will be inferred in the detection if the trigger signal originates from a domain that is asynchronous to the ADC clock. SMPL_CYCLE_EXP determines the the duration of sampling. Conversion starts immediately after sampling ends. In asynchronous mode the sampling is continuous when enabled. Sampling ends and conversion starts immediately with the rising edge of the trigger signal. Sampling restarts when the conversion has finished. Asynchronous mode is useful when it is important to avoid jitter in the sampling instant of an externally driven signal
6-3	SMPL_CYCLE_EXP	R/W	0h	Controls the sampling duration before conversion when the ADC is operated in synchronous mode (SMPL_MODE = 0). The setting has no effect in asynchronous mode. The sampling duration is given as $2^{SMPL\_CYCLE\_EXP + 1}$ / 6 us. 3h = 2P7_US : 16x 6 MHz clock periods = 2.7us 4h = 5P3_US : 32x 6 MHz clock periods = 5.3us 5h = 10P6_US : 64x 6 MHz clock periods = 10.6us 6h = 21P3_US : 128x 6 MHz clock periods = 21.3us 7h = 42P6_US : 256x 6 MHz clock periods = 42.6us 8h = 85P3_US : 512x 6 MHz clock periods = 85.3us 9h = 170_US : 1024x 6 MHz clock periods = 170us Ah = 341_US : 2048x 6 MHz clock periods = 341us Bh = 682_US : 4096x 6 MHz clock periods = 682us Ch = 1P37_MS : 8192x 6 MHz clock periods = 1.37ms Dh = 2P73_MS : 16384x 6 MHz clock periods = 2.73ms Eh = 5P46_MS : 32768x 6 MHz clock periods = 5.46ms Fh = 10P9_MS : 65536x 6 MHz clock periods = 10.9ms
2	RESERVED	R	0h	Reserved
1	RESET_N	R/W	0h	Reset ADC digital subchip, active low. ADC must be reset every time it is reconfigured. 0: Reset 1: Normal operation
0	EN	R/W	0h	ADC Enable 0: Disable 1: Enable

20.8.1.9 ADC1 Register (Offset = 9h) [Reset = 00h]

ADC1 is shown in Table 20-50.

Return to the Summary Table.

ADC Control 1
ADC Comparator Control. Only to be used through TI provided API.

Table 20-50. ADC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-1	RESERVED	R	0h	Reserved
0	SCALE_DIS	R/W	0h	Internal. Only to be used through TI provided API.

20.8.1.10 ADCREF0 Register (Offset = Ah) [Reset = 00h]

ADCREF0 is shown in Table 20-51.

Return to the Summary Table.

ADC Reference 0
Control reference used by the ADC. Only to be used through TI provided API.

Table 20-51. ADCREF0 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0h	Reserved
6	REF_ON_IDLE	R/W	0h	Enable ADCREF in IDLE state. 0: Disabled in IDLE state 1: Enabled in IDLE state Keep ADCREF enabled when ADC0.SMPL_MODE = 0. Recommendation: Enable ADCREF always when ADC0.SMPL_CYCLE_EXP is less than 0x6 (21.3us sampling time).
5	IOMUX	R/W	0h	Internal. Only to be used through TI provided API.
4	EXT	R/W	0h	Internal. Only to be used through TI provided API.
3	SRC	R/W	0h	ADC reference source: 0: Fixed reference = 4.3V 1: Relative reference = VDDS
2-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	ADC reference module enable: 0: ADC reference module powered down 1: ADC reference module enabled

20.8.1.11 ADCREF1 Register (Offset = Bh) [Reset = 00h]

ADCREF1 is shown in Table 20-52.

Return to the Summary Table.

ADC Reference 1
Control reference used by the ADC. Only to be used through TI provided API.

Table 20-52. ADCREF1 Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0h	Reserved
5-0	VTRIM	R/W	0h	Trim output voltage of ADC fixed reference (64 steps, 2's complement). Applies only for ADCREF0.SRC = 0. Examples: 0x00 - nominal voltage 1.43V 0x01 - nominal + 0.4% 1.435V 0x3F - nominal - 0.4% 1.425V 0x1F - maximum voltage 1.6V 0x20 - minimum voltage 1.3V

20.8.1.12 LPMBIAS Register (Offset = Eh) [Reset = 00h]

LPMBIAS is shown in Table 20-53.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-53. LPMBIAS Register Field Descriptions

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0h	Reserved
5-0	LPM_TRIM_IOUT	R/W	0h	Internal. Only to be used through TI provided API.

20.8.2 AUX_AIODIO Registers

Table 20-54 lists the memory-mapped registers for the AUX_AIODIO registers. All register offset addresses not listed in Table 20-54 should be considered as reserved locations and the register contents should not be modified.

Table 20-54. AUX_AIODIO Registers

Offset	Acronym	Register Name	Section
0h	IOMODE	Input Output Mode	Section 20.8.2.1
4h	GPIODIE	General Purpose Input Output Digital Input Enable	Section 20.8.2.2
8h	IOPOE	Input Output Peripheral Output Enable	Section 20.8.2.3
Ch	GPIODOUT	General Purpose Input Output Data Out	Section 20.8.2.4
10h	GPIODIN	General Purpose Input Output Data In	Section 20.8.2.5
14h	GPIODOUTSET	General Purpose Input Output Data Out Set	Section 20.8.2.6
18h	GPIODOUTCLR	General Purpose Input Output Data Out Clear	Section 20.8.2.7
1Ch	GPIODOUTTGL	General Purpose Input Output Data Out Toggle	Section 20.8.2.8
20h	IO0PSEL	Input Output 0 Peripheral Select	Section 20.8.2.9
24h	IO1PSEL	Input Output 1 Peripheral Select	Section 20.8.2.10
28h	IO2PSEL	Input Output 2 Peripheral Select	Section 20.8.2.11
2Ch	IO3PSEL	Input Output 3 Peripheral Select	Section 20.8.2.12
30h	IO4PSEL	Input Output 4 Peripheral Select	Section 20.8.2.13
34h	IO5PSEL	Input Output 5 Peripheral Select	Section 20.8.2.14
38h	IO6PSEL	Input Output 6 Peripheral Select	Section 20.8.2.15
3Ch	IO7PSEL	Input Output 7 Peripheral Select	Section 20.8.2.16
40h	IOMODEL	Input Output Mode Low	Section 20.8.2.17
44h	IOMODEH	Input Output Mode High	Section 20.8.2.18

Complex bit access types are encoded to fit into small table cells. Table 20-55 shows the codes that are used for access types in this section.

Table 20-55. AUX_AIODIO Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.2.1 IOMODE Register (Offset = 0h) [Reset = 00000000h]

IOMODE is shown in Table 20-56.

Return to the Summary Table.

Input Output Mode
This register controls pull-up, pull-down, and output mode for AUXIO that are controlled by instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-56. IOMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	IO7	R/W	0h	Selects mode for AUXIO[8i+7]. 0h = Output Mode: When IOPOE bit 7 is 0: GPIODOUT bit 7 drives AUXIO[8i+7]. When IOPOE bit 7 is 1: The signal selected by IO7PSEL.SRC drives AUXIO[8i+7]. 1h = Input Mode: When GPIODIE bit 7 is 0: AUXIO[8i+7] is enabled for analog signal transfer. When GPIODIE bit 7 is 1: AUXIO[8i+7] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 7 is 0: - If GPIODOUT bit 7 is 0: AUXIO[8i+7] is driven low. - If GPIODOUT bit 7 is 1: AUXIO[8i+7] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 7 is 1: - If signal selected by IO7PSEL.SRC is 0: AUXIO[8i+7] is driven low. - If signal selected by IO7PSEL.SRC is 1: AUXIO[8i+7] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 7 is 0: - If GPIODOUT bit 7 is 0: AUXIO[8i+7] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 7 is 1: AUXIO[8i+7] is driven high. When IOPOE bit 7 is 1: - If signal selected by IO7PSEL.SRC is 0: AUXIO[8i+7] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO7PSEL.SRC is 1: AUXIO[8i+7] is driven high.

Table 20-56. IOMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-12	IO6	R/W	0h	Selects mode for AUXIO[8i+6]. 0h = Output Mode: When IOPOE bit 6 is 0: GPIODOUT bit 6 drives AUXIO[8i+6]. When IOPOE bit 6 is 1: The signal selected by IO6PSEL.SRC drives AUXIO[8i+6]. 1h = Input Mode: When GPIODIE bit 6 is 0: AUXIO[8i+6] is enabled for analog signal transfer. When GPIODIE bit 6 is 1: AUXIO[8i+6] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 6 is 0: - If GPIODOUT bit 6 is 0: AUXIO[8i+6] is driven low. - If GPIODOUT bit 6 is 1: AUXIO[8i+6] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 6 is 1: - If signal selected by IO6PSEL.SRC is 0: AUXIO[8i+6] is driven low. - If signal selected by IO6PSEL.SRC is 1: AUXIO[8i+6] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 6 is 0: - If GPIODOUT bit 6 is 0: AUXIO[8i+6] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 6 is 1: AUXIO[8i+6] is driven high. When IOPOE bit 6 is 1: - If signal selected by IO6PSEL.SRC is 0: AUXIO[8i+6] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO6PSEL.SRC is 1: AUXIO[8i+6] is driven high.
11-10	IO5	R/W	0h	Selects mode for AUXIO[8i+5]. 0h = Output Mode: When IOPOE bit 5 is 0: GPIODOUT bit 5 drives AUXIO[8i+5]. When IOPOE bit 5 is 1: The signal selected by IO5PSEL.SRC drives AUXIO[8i+5]. 1h = Input Mode: When GPIODIE bit 5 is 0: AUXIO[8i+5] is enabled for analog signal transfer. When GPIODIE bit 5 is 1: AUXIO[8i+5] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 5 is 0: - If GPIODOUT bit 5 is 0: AUXIO[8i+5] is driven low. - If GPIODOUT bit 5 is 1: AUXIO[8i+5] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 5 is 1: - If signal selected by IO5PSEL.SRC is 0: AUXIO[8i+5] is driven low. - If signal selected by IO5PSEL.SRC is 1: AUXIO[8i+5] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 5 is 0: - If GPIODOUT bit 5 is 0: AUXIO[8i+5] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 5 is 1: AUXIO[8i+5] is driven high. When IOPOE bit 5 is 1: - If signal selected by IO5PSEL.SRC is 0: AUXIO[8i+5] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO5PSEL.SRC is 1: AUXIO[8i+5] is driven high.

Table 20-56. IOMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	IO4	R/W	0h	Selects mode for AUXIO[8i+4]. 0h = Output Mode: When IOPOE bit 4 is 0: GPIODOUT bit 4 drives AUXIO[8i+4]. When IOPOE bit 4 is 1: The signal selected by IO4PSEL.SRC drives AUXIO[8i+4]. 1h = Input Mode: When GPIODIE bit 4 is 0: AUXIO[8i+4] is enabled for analog signal transfer. When GPIODIE bit 4 is 1: AUXIO[8i+4] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 4 is 0: - If GPIODOUT bit 4 is 0: AUXIO[8i+4] is driven low. - If GPIODOUT bit 4 is 1: AUXIO[8i+4] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 4 is 1: - If signal selected by IO4PSEL.SRC is 0: AUXIO[8i+4] is driven low. - If signal selected by IO4PSEL.SRC is 1: AUXIO[8i+4] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 4 is 0: - If GPIODOUT bit 4 is 0: AUXIO[8i+4] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 4 is 1: AUXIO[8i+4] is driven high. When IOPOE bit 4 is 1: - If signal selected by IO4PSEL.SRC is 0: AUXIO[8i+4] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO4PSEL.SRC is 1: AUXIO[8i+4] is driven high.
7-6	IO3	R/W	0h	Selects mode for AUXIO[8i+3]. 0h = Output Mode: When IOPOE bit 3 is 0: GPIODOUT bit 3 drives AUXIO[8i+3]. When IOPOE bit 3 is 1: The signal selected by IO3PSEL.SRC drives AUXIO[8i+3]. 1h = Input Mode: When GPIODIE bit 3 is 0: AUXIO[8i+3] is enabled for analog signal transfer. When GPIODIE bit 3 is 1: AUXIO[8i+3] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 3 is 0: - If GPIODOUT bit 3 is 0: AUXIO[8i+3] is driven low. - If GPIODOUT bit 3 is 1: AUXIO[8i+3] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 3 is 1: - If signal selected by IO3PSEL.SRC is 0: AUXIO[8i+3] is driven low. - If signal selected by IO3PSEL.SRC is 1: AUXIO[8i+3] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 3 is 0: - If GPIODOUT bit 3 is 0: AUXIO[8i+3] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 3 is 1: AUXIO[8i+3] is driven high. When IOPOE bit 3 is 1: - If signal selected by IO3PSEL.SRC is 0: AUXIO[8i+3] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO3PSEL.SRC is 1: AUXIO[8i+3] is driven high.

Table 20-56. IOMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	IO2	R/W	0h	Select mode for AUXIO[8i+2]. 0h = Output Mode: When IOPOE bit 2 is 0: GPIODOUT bit 2 drives AUXIO[8i+2]. When IOPOE bit 2 is 1: The signal selected by IO2PSEL.SRC drives AUXIO[8i+2]. 1h = Input Mode: When GPIODIE bit 2 is 0: AUXIO[8i+2] is enabled for analog signal transfer. When GPIODIE bit 2 is 1: AUXIO[8i+2] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 2 is 0: - If GPIODOUT bit 2 is 0: AUXIO[8i+2] is driven low. - If GPIODOUT bit 2 is 1: AUXIO[8i+2] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 2 is 1: - If signal selected by IO2PSEL.SRC is 0: AUXIO[8i+2] is driven low. - If signal selected by IO2PSEL.SRC is 1: AUXIO[8i+2] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 2 is 0: - If GPIODOUT bit 2 is 0: AUXIO[8i+2] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 2 is 1: AUXIO[8i+2] is driven high. When IOPOE bit 2 is 1: - If signal selected by IO2PSEL.SRC is 0: AUXIO[8i+2] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO2PSEL.SRC is 1: AUXIO[8i+2] is driven high.
3-2	IO1	R/W	0h	Select mode for AUXIO[8i+1]. 0h = Output Mode: When IOPOE bit 1 is 0: GPIODOUT bit 1 drives AUXIO[8i+1]. When IOPOE bit 1 is 1: The signal selected by IO1PSEL.SRC drives AUXIO[8i+1]. 1h = Input Mode: When GPIODIE bit 1 is 0: AUXIO[8i+1] is enabled for analog signal transfer. When GPIODIE bit 1 is 1: AUXIO[8i+1] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 1 is 0: - If GPIODOUT bit 1 is 0: AUXIO[8i+1] is driven low. - If GPIODOUT bit 1 is 1: AUXIO[8i+1] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 1 is 1: - If signal selected by IO1PSEL.SRC is 0: AUXIO[8i+1] is driven low. - If signal selected by IO1PSEL.SRC is 1: AUXIO[8i+1] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 1 is 0: - If GPIODOUT bit 1 is 0: AUXIO[8i+1] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 1 is 1: AUXIO[8i+1] is driven high. When IOPOE bit 1 is 1: - If signal selected by IO1PSEL.SRC is 0: AUXIO[8i+1] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO1PSEL.SRC is 1: AUXIO[8i+1] is driven high.

Table 20-56. IOMODE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	IO0	R/W	0h	Select mode for AUXIO[8i+0]. 0h = Output Mode: When IOPOE bit 0 is 0: GPIODOUT bit 0 drives AUXIO[8i+0]. When IOPOE bit 0 is 1: The signal selected by IO0PSEL.SRC drives AUXIO[8i+0]. 1h = Input Mode: When GPIODIE bit 0 is 0: AUXIO[8i+0] is enabled for analog signal transfer. When GPIODIE bit 0 is 1: AUXIO[8i+0] is enabled for digital input. 2h = Open-Drain Mode: When IOPOE bit 0 is 0: - If GPIODOUT bit 0 is 0: AUXIO[8i+0] is driven low. - If GPIODOUT bit 0 is 1: AUXIO[8i+0] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. When IOPOE bit 0 is 1: - If signal selected by IO0PSEL.SRC is 0: AUXIO[8i+0] is driven low. - If signal selected by IO0PSEL.SRC is 1: AUXIO[8i+0] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. 3h = Open-Source Mode: When IOPOE bit 0 is 0: - If GPIODOUT bit 0 is 0: AUXIO[8i+0] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If GPIODOUT bit 0 is 1: AUXIO[8i+0] is driven high. When IOPOE bit 0 is 1: - If signal selected by IO0PSEL.SRC is 0: AUXIO[8i+0] is tri-stated or pulled. This depends on IOC:IOCFGn.PULL_CTL. - If signal selected by IO0PSEL.SRC is 1: AUXIO[8i+0] is driven high.

20.8.2.2 GPIODIE Register (Offset = 4h) [Reset = 00000000h]

GPIODIE is shown in Table 20-57.

Return to the Summary Table.

General Purpose Input Output Digital Input Enable
This register controls input buffers for AUXIO that are controlled by instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-57. GPIODIE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to enable digital input buffer for AUXIO[8i+n]. Write 0 to bit index n in this bit vector to disable digital input buffer for AUXIO[8i+n]. You must enable the digital input buffer for AUXIO[8i+n] to read the pin value in GPIODIN. You must disable the digital input buffer for analog input or pins that float to avoid current leakage.

20.8.2.3 IOPOE Register (Offset = 8h) [Reset = 00000000h]

IOPOE is shown in Table 20-58.

Return to the Summary Table.

Input Output Peripheral Output Enable
This register selects the output source for AUXIO that are controlled by instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-58. IOPOE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to configure AUXIO[8i+n] to be driven from source given in [IOnPSEL.*]. Write 0 to bit index n in this bit vector to configure AUXIO[8i+n] to be driven from bit n in GPIODOUT.

20.8.2.4 GPIODOUT Register (Offset = Ch) [Reset = 00000000h]

GPIODOUT is shown in Table 20-59.

Return to the Summary Table.

General Purpose Input Output Data Out
The output data register is used to set data on AUXIO that are controlled by instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-59. GPIODOUT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to set AUXIO[8i+n]. Write 0 to bit index n in this bit vector to clear AUXIO[8i+n]. You must clear bit n in IOPOE to connect bit n in this bit vector to AUXIO[8i+n].

20.8.2.5 GPIODIN Register (Offset = 10h) [Reset = 00000000h]

GPIODIN is shown in Table 20-60.

Return to the Summary Table.

General Purpose Input Output Data In
This register provides synchronized input data for AUXIO that are controlled by instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth

Table 20-60. GPIODIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R	0h	Bit n in this bit vector contains the value for AUXIO[8i+n] when GPIODIE bit n is set. Otherwise, bit n is read as 0.

20.8.2.6 GPIODOUTSET Register (Offset = 14h) [Reset = 00000000h]

GPIODOUTSET is shown in Table 20-61.

Return to the Summary Table.

General Purpose Input Output Data Out Set
Set bits in GPIODOUT in instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-61. GPIODOUTSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to set GPIODOUT bit n. Read value is 0.

20.8.2.7 GPIODOUTCLR Register (Offset = 18h) [Reset = 00000000h]

GPIODOUTCLR is shown in Table 20-62.

Return to the Summary Table.

General Purpose Input Output Data Out Clear
Clear bits in GPIODOUT instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-62. GPIODOUTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to clear GPIODOUT bit n. Read value is 0.

20.8.2.8 GPIODOUTTGL Register (Offset = 1Ch) [Reset = 00000000h]

GPIODOUTTGL is shown in Table 20-63.

Return to the Summary Table.

General Purpose Input Output Data Out Toggle
Toggle bits in GPIODOUT in instance i of AUX_AIODIO. Hence, in formulas below i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-63. GPIODOUTTGL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	IO7_0	R/W	0h	Write 1 to bit index n in this bit vector to toggle GPIODOUT bit n. Read value is 0.

20.8.2.9 IO0PSEL Register (Offset = 20h) [Reset = 00000000h]

IO0PSEL is shown in Table 20-64.

Return to the Summary Table.

Input Output 0 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+0] when IOPOE bit 0 is 1.
To avoid glitches on AUXIO[8i+0] you must configure this register while IOPOE bit 0 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-64. IO0PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+0] when IOPOE bit 0 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.10 IO1PSEL Register (Offset = 24h) [Reset = 00000000h]

IO1PSEL is shown in Table 20-65.

Return to the Summary Table.

Input Output 1 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+1] when IOPOE bit 1 is 1.
To avoid glitches on AUXIO[8i+1] you must configure this register while IOPOE bit 1 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-65. IO1PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+1] when IOPOE bit 1 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.11 IO2PSEL Register (Offset = 28h) [Reset = 00000000h]

IO2PSEL is shown in Table 20-66.

Return to the Summary Table.

Input Output 2 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+2] when IOPOE bit 2 is 1.
To avoid glitches on AUXIO[8i+2] you must configure this register while IOPOE bit 2 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-66. IO2PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+2] when IOPOE bit 2 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.12 IO3PSEL Register (Offset = 2Ch) [Reset = 00000000h]

IO3PSEL is shown in Table 20-67.

Return to the Summary Table.

Input Output 3 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+3] when IOPOE bit 3 is 1.
To avoid glitches on AUXIO[8i+3] you must configure this register while IOPOE bit 3 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-67. IO3PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+3] when IOPOE bit 3 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.13 IO4PSEL Register (Offset = 30h) [Reset = 00000000h]

IO4PSEL is shown in Table 20-68.

Return to the Summary Table.

Input Output 4 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+4] when IOPOE bit 4 is 1.
To avoid glitches on AUXIO[8i+4] you must configure this register while IOPOE bit 4 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-68. IO4PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+4] when IOPOE bit 4 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.14 IO5PSEL Register (Offset = 34h) [Reset = 00000000h]

IO5PSEL is shown in Table 20-69.

Return to the Summary Table.

Input Output 5 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+5] when IOPOE bit 5 is 1.
To avoid glitches on AUXIO[8i+5] you must configure this register while IOPOE bit 5 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-69. IO5PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+5] when IOPOE bit 5 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.15 IO6PSEL Register (Offset = 38h) [Reset = 00000000h]

IO6PSEL is shown in Table 20-70.

Return to the Summary Table.

Input Output 6 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+6] when IOPOE bit 6 is 1.
To avoid glitches on AUXIO[8i+6] you must configure this register while IOPOE bit 6 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-70. IO6PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+6] when IOPOE bit 6 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.16 IO7PSEL Register (Offset = 3Ch) [Reset = 00000000h]

IO7PSEL is shown in Table 20-71.

Return to the Summary Table.

Input Output 7 Peripheral Select
This register selects a peripheral signal that connects to AUXIO[8i+7] when IOPOE bit 7 is 1.
To avoid glitches on AUXIO[8i+7] you must configure this register while IOPOE bit 7 is 0.
In the formulas i = 0 for AUX_AIODIO0, i = 1 for AUX_AIODIO1, and so forth.

Table 20-71. IO7PSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select a peripheral signal that connects to AUXIO[8i+7] when IOPOE bit 7 is set. 0h = Peripheral output mux selects event selected by AUX_EVCTL:EVOBSCFG 1h = Peripheral output mux selects AUX_SPIM SCLK. 2h = Peripheral output mux selects AUX_SPIM MOSI. 3h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0. 4h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1. 5h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2. 6h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3. 7h = Peripheral output mux selects asynchronous version of AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE.

20.8.2.17 IOMODEL Register (Offset = 40h) [Reset = 00000000h]

IOMODEL is shown in Table 20-72.

Return to the Summary Table.

Input Output Mode Low
This is an alias register for IOMODE.IO0 thru IOMODE.IO3.

Table 20-72. IOMODEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-6	IO3	R/W	0h	See IOMODE.IO3.
5-4	IO2	R/W	0h	See IOMODE.IO2.
3-2	IO1	R/W	0h	See IOMODE.IO1.
1-0	IO0	R/W	0h	See IOMODE.IO0.

20.8.2.18 IOMODEH Register (Offset = 44h) [Reset = 00000000h]

IOMODEH is shown in Table 20-73.

Return to the Summary Table.

Input Output Mode High
This is an alias register for IOMODE.IO4 thru IOMODE.IO7.

Table 20-73. IOMODEH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-6	IO7	R/W	0h	See IOMODE.IO7.
5-4	IO6	R/W	0h	See IOMODE.IO6.
3-2	IO5	R/W	0h	See IOMODE.IO5.
1-0	IO4	R/W	0h	See IOMODE.IO4.

20.8.3 AUX_EVCTL Registers

Table 20-74 lists the memory-mapped registers for the AUX_EVCTL registers. All register offset addresses not listed in Table 20-74 should be considered as reserved locations and the register contents should not be modified.

Table 20-74. AUX_EVCTL Registers

Offset	Acronym	Register Name	Section
0h	EVSTAT0	Event Status 0	Section 20.8.3.1
4h	EVSTAT1	Event Status 1	Section 20.8.3.2
8h	EVSTAT2	Event Status 2	Section 20.8.3.3
Ch	EVSTAT3	Event Status 3	Section 20.8.3.4
10h	SCEWEVCFG0	Sensor Controller Engine Wait Event Configuration 0	Section 20.8.3.5
14h	SCEWEVCFG1	Sensor Controller Engine Wait Event Configuration 1	Section 20.8.3.6
18h	DMACTL	Direct Memory Access Control	Section 20.8.3.7
20h	SWEVSET	Software Event Set	Section 20.8.3.8
24h	EVTOAONFLAGS	Events To AON Flags	Section 20.8.3.9
28h	EVTOAONPOL	Events To AON Polarity	Section 20.8.3.10
2Ch	EVTOAONFLAGSCLR	Events To AON Clear	Section 20.8.3.11
30h	EVTOMCUFLAGS	Events to MCU Flags	Section 20.8.3.12
34h	EVTOMCUPOL	Event To MCU Polarity	Section 20.8.3.13
38h	EVTOMCUFLAGSCLR	Events To MCU Flags Clear	Section 20.8.3.14
3Ch	COMBEVTOMCUMASK	Combined Event To MCU Mask	Section 20.8.3.15
40h	EVOBSCFG	Event Observation Configuration	Section 20.8.3.16
44h	PROGDLY	Programmable Delay	Section 20.8.3.17
48h	MANUAL	Manual	Section 20.8.3.18
4Ch	EVSTAT0L	Event Status 0 Low	Section 20.8.3.19
50h	EVSTAT0H	Event Status 0 High	Section 20.8.3.20
54h	EVSTAT1L	Event Status 1 Low	Section 20.8.3.21
58h	EVSTAT1H	Event Status 1 High	Section 20.8.3.22
5Ch	EVSTAT2L	Event Status 2 Low	Section 20.8.3.23
60h	EVSTAT2H	Event Status 2 High	Section 20.8.3.24
64h	EVSTAT3L	Event Status 3 Low	Section 20.8.3.25
68h	EVSTAT3H	Event Status 3 High	Section 20.8.3.26

Complex bit access types are encoded to fit into small table cells. Table 20-75 shows the codes that are used for access types in this section.

Table 20-75. AUX_EVCTL Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W0C	W 0C	Write 0 to clear
Reset or Default Value		
-n		Value after reset or the default value

20.8.3.1 EVSTAT0 Register (Offset = 0h) [Reset = 00000000h]

EVSTAT0 is shown in Table 20-76.

Return to the Summary Table.

Event Status 0
Register holds events 0 thru 15 of the 64-bit event bus that is synchronous to AUX clock. All events read through this register are synchronized at SCE clock rate, unless otherwise noted. The following subscribers use the asynchronous version of events in this register.
- AUX_TIMER2.
- AUX_ANAIF.
- AUX_TDC.
- AUX_SYSIF.
- AUX_AIODIOn.
- EVOBSCFG.

Table 20-76. EVSTAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUXIO15	R	0h	AUXIO15 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 7.
14	AUXIO14	R	0h	AUXIO14 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 6.
13	AUXIO13	R	0h	AUXIO13 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 5.
12	AUXIO12	R	0h	AUXIO12 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 4.
11	AUXIO11	R	0h	AUXIO11 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 3.
10	AUXIO10	R	0h	AUXIO10 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 2.
9	AUXIO9	R	0h	AUXIO9 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 1.
8	AUXIO8	R	0h	AUXIO8 pin level, read value corresponds to AUX_AIODIO1:GPIODIN bit 0.
7	AUXIO7	R	0h	AUXIO7 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 7.
6	AUXIO6	R	0h	AUXIO6 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 6.
5	AUXIO5	R	0h	AUXIO5 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 5.
4	AUXIO4	R	0h	AUXIO4 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 4.
3	AUXIO3	R	0h	AUXIO3 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 3.
2	AUXIO2	R	0h	AUXIO2 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 2.
1	AUXIO1	R	0h	AUXIO1 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 1.
0	AUXIO0	R	0h	AUXIO0 pin level, read value corresponds to AUX_AIODIO0:GPIODIN bit 0.

20.8.3.2 EVSTAT1 Register (Offset = 4h) [Reset = 00000000h]

EVSTAT1 is shown in Table 20-77.

Return to the Summary Table.

Event Status 1
Register holds events 16 thru 31 of the 64-bit event bus that is synchronous to AUX clock. All events read through this register are synchronized at SCE clock rate, unless otherwise noted. The following subscribers use the asynchronous version of events in this register.
- AUX_TIMER2.
- AUX_ANAIF.
- AUX_TDC.
- AUX_SYSIF.
- AUX_AIODIOn.
- EVOBSCFG.

Table 20-77. EVSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUXIO31	R	0h	AUXIO31 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 7.
14	AUXIO30	R	0h	AUXIO30 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 6.
13	AUXIO29	R	0h	AUXIO29 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 5.
12	AUXIO28	R	0h	AUXIO28 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 4.
11	AUXIO27	R	0h	AUXIO27 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 3.
10	AUXIO26	R	0h	AUXIO26 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 2.
9	AUXIO25	R	0h	AUXIO25 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 1.
8	AUXIO24	R	0h	AUXIO24 pin level, read value corresponds to AUX_AIODIO3:GPIODIN bit 0.
7	AUXIO23	R	0h	AUXIO23 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 7.
6	AUXIO22	R	0h	AUXIO22 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 6.
5	AUXIO21	R	0h	AUXIO21 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 5.
4	AUXIO20	R	0h	AUXIO20 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 4.
3	AUXIO19	R	0h	AUXIO19 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 3.
2	AUXIO18	R	0h	AUXIO18 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 2.
1	AUXIO17	R	0h	AUXIO17 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 1.
0	AUXIO16	R	0h	AUXIO16 pin level, read value corresponds to AUX_AIODIO2:GPIODIN bit 0.

20.8.3.3 EVSTAT2 Register (Offset = 8h) [Reset = 00000000h]

EVSTAT2 is shown in Table 20-78.

Return to the Summary Table.

Event Status 2
Register holds events 32 thru 47 of the 64-bit event bus that is synchronous to AUX clock. All events read through this register are synchronized at SCE clock rate, unless otherwise noted. The following subscribers use the asynchronous version of events in this register.
- AUX_TIMER2.
- AUX_ANAIF.
- AUX_TDC.
- AUX_SYSIF.
- AUX_AIODIOn.
- EVOBSCFG.

Table 20-78. EVSTAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_COMPB	R	0h	Comparator B output. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_COMPB_SYNC_RATE sets the synchronization rate for this event.
14	AUX_COMPA	R	0h	Comparator A output. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_COMPA_SYNC_RATE sets the synchronization rate for this event.
13	MCU_OBSMUX1	R	0h	Observation input 1 from IOC. This event is configured by IOC:OBSAUXOUTPUT.SEL1.
12	MCU_OBSMUX0	R	0h	Observation input 0 from IOC. This event is configured by IOC:OBSAUXOUTPUT.SEL0 and can be overridden by IOC:OBSAUXOUTPUT.SEL_MISC.
11	MCU_EV	R	0h	Event from EVENT configured by EVENT:AUXSEL0.
10	ACLK_REF	R	0h	TDC reference clock. It is configured by DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL and enabled by AUX_SYSIF:TDCREFCLKCTL.REQ.
9	VDDR_RECHARGE	R	0h	Event is high during VDDR recharge.
8	MCU_ACTIVE	R	0h	Event is high while system(MCU, AUX, or JTAG domains) is active or transitions to active (GLDO or DCDC power supply state). Event is not high during VDDR recharge.
7	PWR_DWN	R	0h	Event is high while system(MCU, AUX, or JTAG domains) is in powerdown (uLDO power supply).
6	SCLK_LF	R	0h	SCLK_LF clock
5	AON_BATMON_ TEMP_UPD	R	0h	Event is high for two SCLK_MF clock periods when there is an update of AON_BATMON:TEMP.
4	AON_BATMON_ BAT_UPD	R	0h	Event is high for two SCLK_MF clock periods when there is an update of AON_BATMON:BAT.
3	AON_RTC_4KHZ	R	0h	AON_RTC:SUBSEC.VALUE bit 19. AON_RTC:CTL.RTC_4KHZ_EN enables this event.
2	AON_RTC_CH2_DLY	R	0h	AON_RTC:EVFLAGS.CH2 delayed by AON_RTC:CTL.EV_DELAY configuration.
1	AON_RTC_CH2	R	0h	AON_RTC:EVFLAGS.CH2.
0	MANUAL_EV	R	0h	Programmable event. See MANUAL for description.

20.8.3.4 EVSTAT3 Register (Offset = Ch) [Reset = 00000000h]

EVSTAT3 is shown in Table 20-79.

Return to the Summary Table.

Event Status 3
Register holds events 48 thru 63 of the 64-bit event bus that is synchronous to AUX clock. All events read through this register are synchronized at SCE clock rate, unless otherwise noted. The following subscribers use the asynchronous version of events in this register.
- AUX_TIMER2.
- AUX_ANAIF.
- AUX_TDC .
- AUX_SYSIF.
- AUX_AIODIOn.
- EVOBSCFG.

Table 20-79. EVSTAT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_TIMER2_CLKSWITCH_RDY	R	0h	AUX_SYSIF:TIMER2CLKSWITCH.RDY
14	AUX_DAC_HOLD_ACTIVE	R	0h	AUX_ANAIF:DACSTAT.HOLD_ACTIVE
13	AUX_SMPH_AUTOTAKE_DONE	R	0h	See AUX_SMPH:AUTOTAKE.SMPH_ID for description.
12	AUX_ADC_FIFO_NOT_EMPTY	R	0h	AUX_ANAIF:ADCFIFOSTAT.EMPTY negated
11	AUX_ADC_FIFO_ALMOST_FULL	R	0h	AUX_ANAIF:ADCFIFOSTAT.ALMOST_FULL
10	AUX_ADC_IRQ	R	0h	The logical function for this event is configurable. When DMACTL.EN = 1 : Event = UDMA0 Channel 7 done event OR AUX_ANAIF:ADCFIFOSTAT.OVERFLOW OR AUX_ANAIF:ADCFIFOSTAT.UNDERFLOW When DMACTL.EN = 0 : Event = (NOT AUX_ANAIF:ADCFIFOSTAT.EMPTY) OR AUX_ANAIF:ADCFIFOSTAT.OVERFLOW OR AUX_ANAIF:ADCFIFOSTAT.UNDERFLOW Bit 7 in UDMA0:DONEMASK must be 0.
9	AUX_ADC_DONE	R	0h	AUX_ANAIF ADC conversion done event. Event is synchronized at AUX bus rate.
8	AUX_ISRC_RESET_N	R	0h	AUX_ANAIF:ISRCCTL.RESET_N
7	AUX_TDC_DONE	R	0h	AUX_TDC:STAT.DONE
6	AUX_TIMER0_EV	R	0h	AUX_TIMER0_EV event, see AUX_TIMER01:T0TARGET for description.
5	AUX_TIMER1_EV	R	0h	AUX_TIMER1_EV event, see AUX_TIMER01:T1TARGET for description.
4	AUX_TIMER2_PULSE	R	0h	AUX_TIMER2 pulse event. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_TIMER2_SYNC_RATE sets the synchronization rate for this event.
3	AUX_TIMER2_EV3	R	0h	AUX_TIMER2 event output 3. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_TIMER2_SYNC_RATE sets the synchronization rate for this event.
2	AUX_TIMER2_EV2	R	0h	AUX_TIMER2 event output 2. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_TIMER2_SYNC_RATE sets the synchronization rate for this event.

Table 20-79. EVSTAT3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	AUX_TIMER2_EV1	R	0h	AUX_TIMER2 event output 1. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_TIMER2_SYNC_RATE sets the synchronization rate for this event.
0	AUX_TIMER2_EV0	R	0h	AUX_TIMER2 event output 0. Configuration of AUX_SYSIF:EVSYNCRATE.AUX_TIMER2_SYNC_RATE sets the synchronization rate for this event.

20.8.3.5 SCEWEVCFG0 Register (Offset = 10h) [Reset = 00000000h]

SCEWEVCFG0 is shown in Table 20-80.

Return to the Summary Table.

Sensor Controller Engine Wait Event Configuration 0
Configuration of this register and SCEWEVCFG1 controls bit index 7 in AUX_SCE:WUSTAT.EV_SIGNALS. This bit can be used by AUX_SCE WEV0, WEV1, BEV0 and BEV1 instructions.
When COMB_EV_EN = 0:
AUX_SCE:WUSTAT.EV_SIGNALS (7) = EV0_SEL event
When COMB_EV_EN = 1:
AUX_SCE:WUSTAT.EV_SIGNALS (7) = (EV0_SEL event) OR (SCEWEVCFG1.EV1_SEL event)
Bit fields SCEWEVCFG1.EV0_POL and SCEWEVCFG1.EV1_POL control the polarity of selected events.
Event combination is useful when there is a need to wait for a certain condition with timeout.

Table 20-80. SCEWEVCFG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	COMB_EV_EN	R/W	0h	Event combination control: 0: Disable event combination. 1: Enable event combination.

Table 20-80. SCEWEVCFG0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	EV0_SEL	R/W	0h	Select the event source from the synchronous event bus to be used in event equation. 0h = EVSTAT0.AUXIO0 1h = EVSTAT0.AUXIO1 2h = EVSTAT0.AUXIO2 3h = EVSTAT0.AUXIO3 4h = EVSTAT0.AUXIO4 5h = EVSTAT0.AUXIO5 6h = EVSTAT0.AUXIO6 7h = EVSTAT0.AUXIO7 8h = EVSTAT0.AUXIO8 9h = EVSTAT0.AUXIO9 Ah = EVSTAT0.AUXIO10 Bh = EVSTAT0.AUXIO11 Ch = EVSTAT0.AUXIO12 Dh = EVSTAT0.AUXIO13 Eh = EVSTAT0.AUXIO14 Fh = EVSTAT0.AUXIO15 10h = EVSTAT1.AUXIO16 11h = EVSTAT1.AUXIO17 12h = EVSTAT1.AUXIO18 13h = EVSTAT1.AUXIO19 14h = EVSTAT1.AUXIO20 15h = EVSTAT1.AUXIO21 16h = EVSTAT1.AUXIO22 17h = EVSTAT1.AUXIO23 18h = EVSTAT1.AUXIO24 19h = EVSTAT1.AUXIO25 1Ah = EVSTAT1.AUXIO26 1Bh = EVSTAT1.AUXIO27 1Ch = EVSTAT1.AUXIO28 1Dh = EVSTAT1.AUXIO29 1Eh = EVSTAT1.AUXIO30 1Fh = EVSTAT1.AUXIO31 20h = Programmable delay event as described in PROGDLY 21h = EVSTAT2.AON_RTC_CH2 22h = EVSTAT2.AON_RTC_CH2_DLY 23h = EVSTAT2.AON_RTC_4KHZ 24h = EVSTAT2.AON_BATMON_BAT_UPD 25h = EVSTAT2.AON_BATMON_TEMP_UPD 26h = EVSTAT2.SCLK_LF 27h = EVSTAT2.PWR_DWN 28h = EVSTAT2.MCU_ACTIVE 29h = EVSTAT2.VDDR_RECHARGE 2Ah = EVSTAT2.ACLK_REF 2Bh = EVSTAT2.MCU_EV 2Ch = EVSTAT2.MCU_OBSMUX0 2Dh = EVSTAT2.MCU_OBSMUX1 2Eh = EVSTAT2.AUX_COMPA 2Fh = EVSTAT2.AUX_COMPB 30h = EVSTAT3.AUX_TIMER2_EV0 31h = EVSTAT3.AUX_TIMER2_EV1 32h = EVSTAT3.AUX_TIMER2_EV2 33h = EVSTAT3.AUX_TIMER2_EV3 34h = EVSTAT3.AUX_TIMER2_PULSE 35h = EVSTAT3.AUX_TIMER1_EV 36h = EVSTAT3.AUX_TIMER0_EV 37h = EVSTAT3.AUX_TDC_DONE 38h = EVSTAT3.AUX_ISRC_RESET_N 39h = EVSTAT3.AUX_ADC_DONE 3Ah = EVSTAT3.AUX_ADC_IRQ 3Bh = EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Eh = EVSTAT3.AUX_DAC_HOLD_ACTIVE

Table 20-80. SCEWEVCFG0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Fh = EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY

20.8.3.6 SCEWEVCFG1 Register (Offset = 14h) [Reset = 00000000h]

SCEWEVCFG1 is shown in Table 20-81.

Return to the Summary Table.

Sensor Controller Engine Wait Event Configuration 1
See SCEWEVCFG0 for description.

Table 20-81. SCEWEVCFG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV0_POL	R/W	0h	Polarity of SCEWEVCFG0.EV0_SEL event. When SCEWEVCFG0.COMB_EV_EN is 0: 0: Non-inverted. 1: Non-inverted. When SCEWEVCFG0.COMB_EV_EN is 1. 0: Non-inverted. 1: Inverted.
6	EV1_POL	R/W	0h	Polarity of EV1_SEL event. When SCEWEVCFG0.COMB_EV_EN is 0: 0: Non-inverted. 1: Non-inverted. When SCEWEVCFG0.COMB_EV_EN is 1. 0: Non-inverted. 1: Inverted.

Table 20-81. SCEWEVCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	EV1_SEL	R/W	0h	Select the event source from the synchronous event bus to be used in event equation. 0h = EVSTAT0.AUXIO0 1h = EVSTAT0.AUXIO1 2h = EVSTAT0.AUXIO2 3h = EVSTAT0.AUXIO3 4h = EVSTAT0.AUXIO4 5h = EVSTAT0.AUXIO5 6h = EVSTAT0.AUXIO6 7h = EVSTAT0.AUXIO7 8h = EVSTAT0.AUXIO8 9h = EVSTAT0.AUXIO9 Ah = EVSTAT0.AUXIO10 Bh = EVSTAT0.AUXIO11 Ch = EVSTAT0.AUXIO12 Dh = EVSTAT0.AUXIO13 Eh = EVSTAT0.AUXIO14 Fh = EVSTAT0.AUXIO15 10h = EVSTAT1.AUXIO16 11h = EVSTAT1.AUXIO17 12h = EVSTAT1.AUXIO18 13h = EVSTAT1.AUXIO19 14h = EVSTAT1.AUXIO20 15h = EVSTAT1.AUXIO21 16h = EVSTAT1.AUXIO22 17h = EVSTAT1.AUXIO23 18h = EVSTAT1.AUXIO24 19h = EVSTAT1.AUXIO25 1Ah = EVSTAT1.AUXIO26 1Bh = EVSTAT1.AUXIO27 1Ch = EVSTAT1.AUXIO28 1Dh = EVSTAT1.AUXIO29 1Eh = EVSTAT1.AUXIO30 1Fh = EVSTAT1.AUXIO31 20h = Programmable delay event as described in PROGDLY 21h = EVSTAT2.AON_RTC_CH2 22h = EVSTAT2.AON_RTC_CH2_DLY 23h = EVSTAT2.AON_RTC_4KHZ 24h = EVSTAT2.AON_BATMON_BAT_UPD 25h = EVSTAT2.AON_BATMON_TEMP_UPD 26h = EVSTAT2.SCLK_LF 27h = EVSTAT2.PWR_DWN 28h = EVSTAT2.MCU_ACTIVE 29h = EVSTAT2.VDDR_RECHARGE 2Ah = EVSTAT2.ACLK_REF 2Bh = EVSTAT2.MCU_EV 2Ch = EVSTAT2.MCU_OBSMUX0 2Dh = EVSTAT2.MCU_OBSMUX1 2Eh = EVSTAT2.AUX_COMPA 2Fh = EVSTAT2.AUX_COMPB 30h = EVSTAT3.AUX_TIMER2_EV0 31h = EVSTAT3.AUX_TIMER2_EV1 32h = EVSTAT3.AUX_TIMER2_EV2 33h = EVSTAT3.AUX_TIMER2_EV3 34h = EVSTAT3.AUX_TIMER2_PULSE 35h = EVSTAT3.AUX_TIMER1_EV 36h = EVSTAT3.AUX_TIMER0_EV 37h = EVSTAT3.AUX_TDC_DONE 38h = EVSTAT3.AUX_ISRC_RESET_N 39h = EVSTAT3.AUX_ADC_DONE 3Ah = EVSTAT3.AUX_ADC_IRQ 3Bh = EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Eh = EVSTAT3.AUX_DAC_HOLD_ACTIVE

Table 20-81. SCEWEVCFG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Fh = EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY

20.8.3.7 DMACTL Register (Offset = 18h) [Reset = 00000000h]

DMACTL is shown in Table 20-82.

Return to the Summary Table.

Direct Memory Access Control

Table 20-82. DMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	REQ_MODE	R/W	0h	UDMA0 Request mode 0h = Burst requests are generated on UDMA0 channel 7 when the condition configured in SEL is met. 1h = Single requests are generated on UDMA0 channel 7 when the condition configured in SEL is met.
1	EN	R/W	0h	uDMA ADC interface enable. 0: Disable UDMA0 interface to ADC. 1: Enable UDMA0 interface to ADC.
0	SEL	R/W	0h	Select FIFO watermark level required to trigger a UDMA0 transfer of ADC FIFO data. 0h = UDMA0 trigger event will be generated when there are samples in the ADC FIFO. 1h = UDMA0 trigger event will be generated when the ADC FIFO is almost full (3/4 full).

20.8.3.8 SWEVSET Register (Offset = 20h) [Reset = 00000000h]

SWEVSET is shown in Table 20-83.

Return to the Summary Table.

Software Event Set
Set software event flags from AUX domain to AON and MCU domains. CPUs in MCU domain can read the event flags from EVTOAONFLAGS and clear them in EVTOAONFLAGSCLR.
Use of these event flags is software-defined.

Table 20-83. SWEVSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	SWEV2	W	0h	Software event flag 2. 0: No effect. 1: Set software event flag 2.
1	SWEV1	W	0h	Software event flag 1. 0: No effect. 1: Set software event flag 1.
0	SWEV0	W	0h	Software event flag 0. 0: No effect. 1: Set software event flag 0.

20.8.3.9 EVTOAONFLAGS Register (Offset = 24h) [Reset = 00000000h]

EVTOAONFLAGS is shown in Table 20-84.

Return to the Summary Table.

Events To AON Flags
This register contains a collection of event flags routed to AON_EVENT.
To clear an event flag, write to EVTOAONFLAGSCLR or write 0 to event flag in this register.

Table 20-84. EVTOAONFLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	AUX_TIMER1_EV	R/W0C	0h	This event flag is set when level selected by EVTOAONPOL.AUX_TIMER1_EV occurs on EVSTAT3.AUX_TIMER1_EV.
7	AUX_TIMER0_EV	R/W0C	0h	This event flag is set when level selected by EVTOAONPOL.AUX_TIMER0_EV occurs on EVSTAT3.AUX_TIMER0_EV.
6	AUX_TDC_DONE	R/W0C	0h	This event flag is set when level selected by EVTOAONPOL.AUX_TDC_DONE occurs on EVSTAT3.AUX_TDC_DONE.
5	AUX_ADC_DONE	R/W0C	0h	This event flag is set when level selected by EVTOAONPOL.AUX_ADC_DONE occurs on EVSTAT3.AUX_ADC_DONE.
4	AUX_COMPB	R/W0C	0h	This event flag is set when edge selected by EVTOAONPOL.AUX_COMPB occurs on EVSTAT2.AUX_COMPB.
3	AUX_COMPA	R/W0C	0h	This event flag is set when edge selected by EVTOAONPOL.AUX_COMPA occurs on EVSTAT2.AUX_COMPA.
2	SWEV2	R/W0C	0h	This event flag is set when software writes a 1 to SWEVSET.SWEV2.
1	SWEV1	R/W0C	0h	This event flag is set when software writes a 1 to SWEVSET.SWEV1.
0	SWEV0	R/W0C	0h	This event flag is set when software writes a 1 to SWEVSET.SWEV0.

20.8.3.10 EVTOAONPOL Register (Offset = 28h) [Reset = 00000000h]

EVTOAONPOL is shown in Table 20-85.

Return to the Summary Table.

Events To AON Polarity
Event source polarity configuration for EVTOAONFLAGS.

Table 20-85. EVTOAONPOL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	AUX_TIMER1_EV	R/W	0h	Select the level of EVSTAT3.AUX_TIMER1_EV that sets EVTOAONFLAGS.AUX_TIMER1_EV. 0h = High level 1h = Low level
7	AUX_TIMER0_EV	R/W	0h	Select the level of EVSTAT3.AUX_TIMER0_EV that sets EVTOAONFLAGS.AUX_TIMER0_EV. 0h = High level 1h = Low level
6	AUX_TDC_DONE	R/W	0h	Select level of EVSTAT3.AUX_TDC_DONE that sets EVTOAONFLAGS.AUX_TDC_DONE. 0h = High level 1h = Low level
5	AUX_ADC_DONE	R/W	0h	Select the level of EVSTAT3.AUX_ADC_DONE that sets EVTOAONFLAGS.AUX_ADC_DONE. 0h = High level 1h = Low level
4	AUX_COMPB	R/W	0h	Select the edge of EVSTAT2.AUX_COMPB that sets EVTOAONFLAGS.AUX_COMPB. 0h = Rising edge 1h = Falling edge
3	AUX_COMPA	R/W	0h	Select the edge of EVSTAT2.AUX_COMPA that sets EVTOAONFLAGS.AUX_COMPA. 0h = Rising edge 1h = Falling edge
2-0	RESERVED	R	0h	Reserved

20.8.3.11 EVTOAONFLAGSCLR Register (Offset = 2Ch) [Reset = 00000000h]

EVTOAONFLAGSCLR is shown in Table 20-86.

Return to the Summary Table.

Events To AON Clear
Clear event flags in EVTOAONFLAGS.
In order to clear a level sensitive event flag, the event must be deasserted.

Table 20-86. EVTOAONFLAGSCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8	AUX_TIMER1_EV	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_TIMER1_EV. Read value is 0.
7	AUX_TIMER0_EV	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_TIMER0_EV. Read value is 0.
6	AUX_TDC_DONE	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_TDC_DONE. Read value is 0.
5	AUX_ADC_DONE	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_ADC_DONE. Read value is 0.
4	AUX_COMPB	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_COMPB. Read value is 0.
3	AUX_COMPA	W	0h	Write 1 to clear EVTOAONFLAGS.AUX_COMPA. Read value is 0.
2	SWEV2	W	0h	Write 1 to clear EVTOAONFLAGS.SWEV2. Read value is 0.
1	SWEV1	W	0h	Write 1 to clear EVTOAONFLAGS.SWEV1. Read value is 0.
0	SWEV0	W	0h	Write 1 to clear EVTOAONFLAGS.SWEV0. Read value is 0.

20.8.3.12 EVTOMCUFLAGS Register (Offset = 30h) [Reset = 00000000h]

EVTOMCUFLAGS is shown in Table 20-87.

Return to the Summary Table.

Events to MCU Flags
This register contains a collection of event flags routed to MCU domain.
To clear an event flag, write to EVTOMCUFLAGSCLR or write 0 to event flag in this register. Follow procedure described in AUX_SYSIF:WUCLR to clear AUX_WU_EV event flag.

Table 20-87. EVTOMCUFLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_TIMER2_PULSE	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER2_PULSE occurs on EVSTAT3.AUX_TIMER2_PULSE.
14	AUX_TIMER2_EV3	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER2_EV3 occurs on EVSTAT3.AUX_TIMER2_EV3.
13	AUX_TIMER2_EV2	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER2_EV2 occurs on EVSTAT3.AUX_TIMER2_EV2.
12	AUX_TIMER2_EV1	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER2_EV1 occurs on EVSTAT3.AUX_TIMER2_EV1.
11	AUX_TIMER2_EV0	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER2_EV0 occurs on EVSTAT3.AUX_TIMER2_EV0.
10	AUX_ADC_IRQ	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_ADC_IRQ occurs on EVSTAT3.AUX_ADC_IRQ.
9	MCU_OBSMUX0	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.MCU_OBSMUX0 occurs on EVSTAT2.MCU_OBSMUX0.
8	AUX_ADC_FIFO_ALMOST_FULL	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_ADC_FIFO_ALMOST_FULL occurs on EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL.
7	AUX_ADC_DONE	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_ADC_DONE occurs on EVSTAT3.AUX_ADC_DONE.
6	AUX_SMPH_AUTOTAKE_DONE	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_SMPH_AUTOTAKE_DONE occurs on EVSTAT3.AUX_SMPH_AUTOTAKE_DONE.
5	AUX_TIMER1_EV	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER1_EV occurs on EVSTAT3.AUX_TIMER1_EV.
4	AUX_TIMER0_EV	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TIMER0_EV occurs on EVSTAT3.AUX_TIMER0_EV.
3	AUX_TDC_DONE	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_TDC_DONE occurs on EVSTAT3.AUX_TDC_DONE.
2	AUX_COMPB	R/W0C	0h	This event flag is set when edge selected by EVTOMCUPOL.AUX_COMPB occurs on EVSTAT2.AUX_COMPB.
1	AUX_COMPA	R/W0C	0h	This event flag is set when edge selected by EVTOMCUPOL.AUX_COMPA occurs on EVSTAT2.AUX_COMPA.

Table 20-87. EVTOMCUFLAGS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	AUX_WU_EV	R/W0C	0h	This event flag is set when level selected by EVTOMCUPOL.AUX_WU_EV occurs on reduction-OR of the AUX_SYSIF:WUFLAGS register.

20.8.3.13 EVTOMCUPOL Register (Offset = 34h) [Reset = 00000000h]

EVTOMCUPOL is shown in Table 20-88.

Return to the Summary Table.

Event To MCU Polarity
Event source polarity configuration for EVTOMCUFLAGS.

Table 20-88. EVTOMCUPOL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_TIMER2_PULSE	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER2_PULSE. 0h = High level 1h = Low level
14	AUX_TIMER2_EV3	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER2_EV3. 0h = High level 1h = Low level
13	AUX_TIMER2_EV2	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER2_EV2. 0h = High level 1h = Low level
12	AUX_TIMER2_EV1	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER2_EV1. 0h = High level 1h = Low level
11	AUX_TIMER2_EV0	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER2_EV0. 0h = High level 1h = Low level
10	AUX_ADC_IRQ	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_ADC_IRQ. 0h = High level 1h = Low level
9	MCU_OBSMUX0	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.MCU_OBSMUX0. 0h = High level 1h = Low level
8	AUX_ADC_FIFO_ALMOST_FULL	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL. 0h = High level 1h = Low level
7	AUX_ADC_DONE	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_ADC_DONE. 0h = High level 1h = Low level
6	AUX_SMPH_AUTOTAKE_DONE	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_SMPH_AUTOTAKE_DONE. 0h = High level 1h = Low level
5	AUX_TIMER1_EV	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER1_EV. 0h = High level 1h = Low level
4	AUX_TIMER0_EV	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TIMER0_EV. 0h = High level 1h = Low level

Table 20-88. EVTOMCUPOL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	AUX_TDC_DONE	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_TDC_DONE. 0h = High level 1h = Low level
2	AUX_COMPB	R/W	0h	Select the event source edge that sets EVTOMCUFLAGS.AUX_COMPB. 0h = Rising edge 1h = Falling edge
1	AUX_COMPA	R/W	0h	Select the event source edge that sets EVTOMCUFLAGS.AUX_COMPA. 0h = Rising edge 1h = Falling edge
0	AUX_WU_EV	R/W	0h	Select the event source level that sets EVTOMCUFLAGS.AUX_WU_EV. 0h = High level 1h = Low level

20.8.3.14 EVTOMCUFLAGSCLR Register (Offset = 38h) [Reset = 00000000h]

EVTOMCUFLAGSCLR is shown in Table 20-89.

Return to the Summary Table.

Events To MCU Flags Clear
Clear event flags in EVTOMCUFLAGS.
In order to clear a level sensitive event flag, the event must be deasserted.

Table 20-89. EVTOMCUFLAGSCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_TIMER2_PULSE	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER2_PULSE. Read value is 0.
14	AUX_TIMER2_EV3	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER2_EV3. Read value is 0.
13	AUX_TIMER2_EV2	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER2_EV2. Read value is 0.
12	AUX_TIMER2_EV1	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER2_EV1. Read value is 0.
11	AUX_TIMER2_EV0	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER2_EV0. Read value is 0.
10	AUX_ADC_IRQ	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_ADC_IRQ. Read value is 0.
9	MCU_OBSMUX0	W	0h	Write 1 to clear EVTOMCUFLAGS.MCU_OBSMUX0. Read value is 0.
8	AUX_ADC_FIFO_ALMOST_FULL	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL. Read value is 0.
7	AUX_ADC_DONE	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_ADC_DONE. Read value is 0.
6	AUX_SMPH_AUTOTAKE_DONE	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_SMPH_AUTOTAKE_DONE. Read value is 0.
5	AUX_TIMER1_EV	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER1_EV. Read value is 0.
4	AUX_TIMER0_EV	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TIMER0_EV. Read value is 0.
3	AUX_TDC_DONE	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_TDC_DONE. Read value is 0.
2	AUX_COMPB	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_COMPB. Read value is 0.
1	AUX_COMPA	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_COMPA. Read value is 0.
0	AUX_WU_EV	W	0h	Write 1 to clear EVTOMCUFLAGS.AUX_WU_EV. Read value is 0.

20.8.3.15 COMBEVTOMCUMASK Register (Offset = 3Ch) [Reset = 00000000h]

COMBEVTOMCUMASK is shown in Table 20-90.

Return to the Summary Table.

Combined Event To MCU Mask
Select event flags in EVTOMCUFLAGS that contribute to the AUX_COMB event to EVENT and system CPU. The AUX_COMB event is high as long as one or more of the included event flags are set.

Table 20-90. COMBEVTOMCUMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	AUX_TIMER2_PULSE	R/W	0h	EVTOMCUFLAGS.AUX_TIMER2_PULSE contribution to the AUX_COMB event. 0: Exclude. 1: Include.
14	AUX_TIMER2_EV3	R/W	0h	EVTOMCUFLAGS.AUX_TIMER2_EV3 contribution to the AUX_COMB event. 0: Exclude. 1: Include.
13	AUX_TIMER2_EV2	R/W	0h	EVTOMCUFLAGS.AUX_TIMER2_EV2 contribution to the AUX_COMB event. 0: Exclude. 1: Include.
12	AUX_TIMER2_EV1	R/W	0h	EVTOMCUFLAGS.AUX_TIMER2_EV1 contribution to the AUX_COMB event. 0: Exclude. 1: Include.
11	AUX_TIMER2_EV0	R/W	0h	EVTOMCUFLAGS.AUX_TIMER2_EV0 contribution to the AUX_COMB event. 0: Exclude. 1: Include.
10	AUX_ADC_IRQ	R/W	0h	EVTOMCUFLAGS.AUX_ADC_IRQ contribution to the AUX_COMB event. 0: Exclude. 1: Include.
9	MCU_OBSMUX0	R/W	0h	EVTOMCUFLAGS.MCU_OBSMUX0 contribution to the AUX_COMB event. 0: Exclude. 1: Include.
8	AUX_ADC_FIFO_ ALMOST_FULL	R/W	0h	EVTOMCUFLAGS.AUX_ADC_FIFO_ALMOST_FULL contribution to the AUX_COMB event. 0: Exclude. 1: Include.
7	AUX_ADC_DONE	R/W	0h	EVTOMCUFLAGS.AUX_ADC_DONE contribution to the AUX_COMB event. 0: Exclude. 1: Include.
6	AUX_SMPH_AUTOTAKE_ DONE	R/W	0h	EVTOMCUFLAGS.AUX_SMPH_AUTOTAKE_DONE contribution to the AUX_COMB event. 0: Exclude. 1: Include.
5	AUX_TIMER1_EV	R/W	0h	EVTOMCUFLAGS.AUX_TIMER1_EV contribution to the AUX_COMB event. 0: Exclude. 1: Include.
4	AUX_TIMER0_EV	R/W	0h	EVTOMCUFLAGS.AUX_TIMER0_EV contribution to the AUX_COMB event. 0: Exclude. 1: Include.

Table 20-90. COMBEVTOMCUMASK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	AUX_TDC_DONE	R/W	0h	EVTOMCUFLAGS.AUX_TDC_DONE contribution to the AUX_COMB event. 0: Exclude. 1: Include.
2	AUX_COMPB	R/W	0h	EVTOMCUFLAGS.AUX_COMPB contribution to the AUX_COMB event. 0: Exclude 1: Include.
1	AUX_COMPA	R/W	0h	EVTOMCUFLAGS.AUX_COMPA contribution to the AUX_COMB event. 0: Exclude. 1: Include.
0	AUX_WU_EV	R/W	0h	EVTOMCUFLAGS.AUX_WU_EV contribution to the AUX_COMB event. 0: Exclude. 1: Include.

20.8.3.16 EVOBSCFG Register (Offset = 40h) [Reset = 00000000h]

EVOBSCFG is shown in Table 20-91.

Return to the Summary Table.

Event Observation Configuration

Table 20-91. EVOBSCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved

Table 20-91. EVOBSCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	EVOBS_SEL	R/W	0h	Select which event from the asynchronous event bus that represents AUX_EV_OBS in AUX_AIODIOn. 0h = EVSTAT0.AUXIO0 1h = EVSTAT0.AUXIO1 2h = EVSTAT0.AUXIO2 3h = EVSTAT0.AUXIO3 4h = EVSTAT0.AUXIO4 5h = EVSTAT0.AUXIO5 6h = EVSTAT0.AUXIO6 7h = EVSTAT0.AUXIO7 8h = EVSTAT0.AUXIO8 9h = EVSTAT0.AUXIO9 Ah = EVSTAT0.AUXIO10 Bh = EVSTAT0.AUXIO11 Ch = EVSTAT0.AUXIO12 Dh = EVSTAT0.AUXIO13 Eh = EVSTAT0.AUXIO14 Fh = EVSTAT0.AUXIO15 10h = EVSTAT1.AUXIO16 11h = EVSTAT1.AUXIO17 12h = EVSTAT1.AUXIO18 13h = EVSTAT1.AUXIO19 14h = EVSTAT1.AUXIO20 15h = EVSTAT1.AUXIO21 16h = EVSTAT1.AUXIO22 17h = EVSTAT1.AUXIO23 18h = EVSTAT1.AUXIO24 19h = EVSTAT1.AUXIO25 1Ah = EVSTAT1.AUXIO26 1Bh = EVSTAT1.AUXIO27 1Ch = EVSTAT1.AUXIO28 1Dh = EVSTAT1.AUXIO29 1Eh = EVSTAT1.AUXIO30 1Fh = EVSTAT1.AUXIO31 20h = EVSTAT2.MANUAL_EV 21h = EVSTAT2.AON_RTC_CH2 22h = EVSTAT2.AON_RTC_CH2_DLY 23h = EVSTAT2.AON_RTC_4KHZ 24h = EVSTAT2.AON_BATMON_BAT_UPD 25h = EVSTAT2.AON_BATMON_TEMP_UPD 26h = EVSTAT2.SCLK_LF 27h = EVSTAT2.PWR_DWN 28h = EVSTAT2.MCU_ACTIVE 29h = EVSTAT2.VDDR_RECHARGE 2Ah = EVSTAT2.ACLK_REF 2Bh = EVSTAT2.MCU_EV 2Ch = EVSTAT2.MCU_OBSMUX0 2Dh = EVSTAT2.MCU_OBSMUX1 2Eh = EVSTAT2.AUX_COMPA 2Fh = EVSTAT2.AUX_COMPB 30h = EVSTAT3.AUX_TIMER2_EV0 31h = EVSTAT3.AUX_TIMER2_EV1 32h = EVSTAT3.AUX_TIMER2_EV2 33h = EVSTAT3.AUX_TIMER2_EV3 34h = EVSTAT3.AUX_TIMER2_PULSE 35h = EVSTAT3.AUX_TIMER1_EV 36h = EVSTAT3.AUX_TIMER0_EV 37h = EVSTAT3.AUX_TDC_DONE 38h = EVSTAT3.AUX_ISRC_RESET_N 39h = EVSTAT3.AUX_ADC_DONE 3Ah = EVSTAT3.AUX_ADC_IRQ 3Bh = EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Eh = EVSTAT3.AUX_DAC_HOLD_ACTIVE

Table 20-91. EVOBSCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Fh = EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY

20.8.3.17 PROGDLY Register (Offset = 44h) [Reset = 00000000h]

PROGDLY is shown in Table 20-92.

Return to the Summary Table.

Programmable Delay

Table 20-92. PROGDLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	VALUE decrements to 0 at a rate of 1 MHz. The event AUX_PROG_DLY_IDLE is high when VALUE is 0, otherwise it is low. Only use the programmable delay counter and the AUX_PROG_DLY_IDLE event when AUX_SYSIF:OPMODEACK.ACK equals A or LP. Decrementation of VALUE halts when either is true: - AUX_SCE:CTL.DBG_FREEZE_EN is set and system CPU is halted in debug mode. - AUX_SYSIF:TIMERHALT.PROGDLY is set.

20.8.3.18 MANUAL Register (Offset = 48h) [Reset = 00000000h]

MANUAL is shown in Table 20-93.

Return to the Summary Table.

Manual
Programmable event.

Table 20-93. MANUAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EV	R/W	0h	This bit field sets the value of EVSTAT2.MANUAL_EV.

20.8.3.19 EVSTAT0L Register (Offset = 4Ch) [Reset = 00000000h]

EVSTAT0L is shown in Table 20-94.

Return to the Summary Table.

Event Status 0 Low

Table 20-94. EVSTAT0L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT0 event 7 down to 0.

20.8.3.20 EVSTAT0H Register (Offset = 50h) [Reset = 00000000h]

EVSTAT0H is shown in Table 20-95.

Return to the Summary Table.

Event Status 0 High

Table 20-95. EVSTAT0H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT0 event 15 down to 8.

20.8.3.21 EVSTAT1L Register (Offset = 54h) [Reset = 00000000h]

EVSTAT1L is shown in Table 20-96.

Return to the Summary Table.

Event Status 1 Low

Table 20-96. EVSTAT1L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT1 event 7 down to 0.

20.8.3.22 EVSTAT1H Register (Offset = 58h) [Reset = 00000000h]

EVSTAT1H is shown in Table 20-97.

Return to the Summary Table.

Event Status 1 High

Table 20-97. EVSTAT1H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT1 event 15 down to 8.

20.8.3.23 EVSTAT2L Register (Offset = 5Ch) [Reset = 00000000h]

EVSTAT2L is shown in Table 20-98.

Return to the Summary Table.

Event Status 2 Low

Table 20-98. EVSTAT2L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT2 event 7 down to 0.

20.8.3.24 EVSTAT2H Register (Offset = 60h) [Reset = 00000000h]

EVSTAT2H is shown in Table 20-99.

Return to the Summary Table.

Event Status 2 High

Table 20-99. EVSTAT2H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT2 event 15 down to 8.

20.8.3.25 EVSTAT3L Register (Offset = 64h) [Reset = 00000000h]

EVSTAT3L is shown in Table 20-100.

Return to the Summary Table.

Event Status 3 Low

Table 20-100. EVSTAT3L Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT3 event 7 down to 0.

20.8.3.26 EVSTAT3H Register (Offset = 68h) [Reset = 00000000h]

EVSTAT3H is shown in Table 20-101.

Return to the Summary Table.

Event Status 3 High

Table 20-101. EVSTAT3H Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ALIAS_EV	R	0h	Alias of EVSTAT3 event 15 down to 8.

20.8.4 AUX_SMPH Registers

Table 20-102 lists the memory-mapped registers for the AUX_SMPH registers. All register offset addresses not listed in Table 20-102 should be considered as reserved locations and the register contents should not be modified.

Table 20-102. AUX_SMPH Registers

Offset	Acronym	Register Name	Section
0h	SMPH0	Semaphore 0	Section 20.8.4.1
4h	SMPH1	Semaphore 1	Section 20.8.4.2
8h	SMPH2	Semaphore 2	Section 20.8.4.3
Ch	SMPH3	Semaphore 3	Section 20.8.4.4
10h	SMPH4	Semaphore 4	Section 20.8.4.5
14h	SMPH5	Semaphore 5	Section 20.8.4.6
18h	SMPH6	Semaphore 6	Section 20.8.4.7
1Ch	SMPH7	Semaphore 7	Section 20.8.4.8
20h	AUTOTAKE	Auto Take	Section 20.8.4.9

Complex bit access types are encoded to fit into small table cells. Table 20-103 shows the codes that are used for access types in this section.

Table 20-103. AUX_SMPH Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.4.1 SMPH0 Register (Offset = 0h) [Reset = 00000001h]

SMPH0 is shown in Table 20-104.

Return to the Summary Table.

Semaphore 0

Table 20-104. SMPH0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.2 SMPH1 Register (Offset = 4h) [Reset = 00000001h]

SMPH1 is shown in Table 20-105.

Return to the Summary Table.

Semaphore 1

Table 20-105. SMPH1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.3 SMPH2 Register (Offset = 8h) [Reset = 00000001h]

SMPH2 is shown in Table 20-106.

Return to the Summary Table.

Semaphore 2

Table 20-106. SMPH2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.4 SMPH3 Register (Offset = Ch) [Reset = 00000001h]

SMPH3 is shown in Table 20-107.

Return to the Summary Table.

Semaphore 3

Table 20-107. SMPH3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.5 SMPH4 Register (Offset = 10h) [Reset = 00000001h]

SMPH4 is shown in Table 20-108.

Return to the Summary Table.

Semaphore 4

Table 20-108. SMPH4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.6 SMPH5 Register (Offset = 14h) [Reset = 00000001h]

SMPH5 is shown in Table 20-109.

Return to the Summary Table.

Semaphore 5

Table 20-109. SMPH5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.7 SMPH6 Register (Offset = 18h) [Reset = 00000001h]

SMPH6 is shown in Table 20-110.

Return to the Summary Table.

Semaphore 6

Table 20-110. SMPH6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.8 SMPH7 Register (Offset = 1Ch) [Reset = 00000001h]

SMPH7 is shown in Table 20-111.

Return to the Summary Table.

Semaphore 7

Table 20-111. SMPH7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Request or release of semaphore. Request by read: 0: Semaphore not available. 1: Semaphore granted. Release by write: 0: Do not use. 1: Release semaphore.

20.8.4.9 AUTOTAKE Register (Offset = 20h) [Reset = 00000000h]

AUTOTAKE is shown in Table 20-112.

Return to the Summary Table.

Auto Take
Sticky Request for Single Semaphore.

Table 20-112. AUTOTAKE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SMPH_ID	R/W	0h	Write the semaphore ID,0x0-0x7, to SMPH_ID to request this semaphore until it is granted. When semaphore SMPH_ID is granted, event AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE becomes 1. The event becomes 0 when software releases the semaphore or writes a new value to SMPH_ID. To avoid corrupted semaphores: - Usage of this functionality must be restricted to one CPU core. - Software must wait until AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE is 1 before it writes a new value to SMPH_ID.

20.8.5 AUX_TDC Registers

Table 20-113 lists the memory-mapped registers for the AUX_TDC registers. All register offset addresses not listed in Table 20-113 should be considered as reserved locations and the register contents should not be modified.

Table 20-113. AUX_TDC Registers

Offset	Acronym	Register Name	Section
0h	CTL	Control	Section 20.8.5.1
4h	STAT	Status	Section 20.8.5.2
8h	RESULT	Result	Section 20.8.5.3
Ch	SATCFG	Saturation Configuration	Section 20.8.5.4
10h	TRIGSRC	Trigger Source	Section 20.8.5.5
14h	TRIGCNT	Trigger Counter	Section 20.8.5.6
18h	TRIGCNTLOAD	Trigger Counter Load	Section 20.8.5.7
1Ch	TRIGCNTCFG	Trigger Counter Configuration	Section 20.8.5.8
20h	PRECTL	Prescaler Control	Section 20.8.5.9
24h	PRECNTR	Prescaler Counter	Section 20.8.5.10

Complex bit access types are encoded to fit into small table cells. Table 20-114 shows the codes that are used for access types in this section.

Table 20-114. AUX_TDC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.5.1 CTL Register (Offset = 0h) [Reset = 00000000h]

CTL is shown in Table 20-115.

Return to the Summary Table.

Control

Table 20-115. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	CMD	W	0h	TDC commands. 0h = Clear STAT.SAT, STAT.DONE, and RESULT.VALUE. This is not needed as prerequisite for a measurement. Reliable clear is only guaranteed from IDLE state. 1h = Synchronous counter start. The counter looks for the opposite edge of the selected start event before it starts to count when the selected edge occurs. This guarantees an edge-triggered start and is recommended for frequency measurements. 2h = Asynchronous counter start. The counter starts to count when the start event is high. To achieve precise edge-to-edge measurements you must ensure that the start event is low for at least 420 ns after you write this command. 3h = Force TDC state machine back to IDLE state. Never write this command while AUX_TDC:STAT.STATE equals CLR_CNT or WAIT_CLR_CNT_DONE.

20.8.5.2 STAT Register (Offset = 4h) [Reset = 00000006h]

STAT is shown in Table 20-116.

Return to the Summary Table.

Status

Table 20-116. STAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	SAT	R	0h	TDC measurement saturation flag. 0: Conversion has not saturated. 1: Conversion stopped due to saturation. This field is cleared when a new measurement is started or when CLR_RESULT is written to CTL.CMD.
6	DONE	R	0h	TDC measurement complete flag. 0: TDC measurement has not yet completed. 1: TDC measurement has completed. This field clears when a new TDC measurement starts or when you write CLR_RESULT to CTL.CMD.
5-0	STATE	R	6h	TDC state machine status. 0h = Current state is TDC_STATE_WAIT_START. The fast-counter circuit looks for the start condition. The state machine waits for the fast-counter to increment. 4h = Current state is TDC_STATE_WAIT_STARTSTOPCNTEN. The fast-counter circuit looks for the start condition. The state machine waits for the fast-counter to increment. 6h = Current state is TDC_STATE_IDLE. This is the default state after reset and abortion. State will change when you write CTL.CMD to either RUN_SYNC_START or RUN. 7h = Current state is TDC_STATE_CLRCNT. The fast-counter circuit is reset. 8h = Current state is TDC_STATE_WAIT_STOP. The state machine waits for the fast-counter circuit to stop. Ch = Current state is TDC_STATE_WAIT_STOPCNTDOWN. The fast-counter circuit looks for the stop condition. It will ignore a number of stop events configured in TRIGCNTLOAD.CNT. Eh = Current state is TDC_STATE_GETRESULTS. The state machine copies the counter value from the fast-counter circuit. Fh = Current state is TDC_STATE_POR. This is the reset state. 16h = Current state is TDC_STATE_WAIT_CLRCNT_DONE. The state machine waits for fast-counter circuit to finish reset. 1Eh = Current state is TDC_WAIT_STARTFALL. The fast-counter circuit waits for a falling edge on the start event. 2Eh = Current state is TDC_FORCESTOP. You wrote ABORT to CTL.CMD to abort the TDC measurement.

20.8.5.3 RESULT Register (Offset = 8h) [Reset = 00000002h]

RESULT is shown in Table 20-117.

Return to the Summary Table.

Result
Result of last TDC conversion.

Table 20-117. RESULT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-0	VALUE	R	2h	TDC conversion result. The result of the TDC conversion is given in number of clock edges of the clock source selected in DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL. Both rising and falling edges are counted. If TDC counter saturates, VALUE is slightly higher than SATCFG.LIMIT, as it takes a non-zero time to stop the measurement. Hence, the maximum value of this field becomes slightly higher than 2^{24} if you configure SATCFG.LIMIT to R24.

20.8.5.4 SATCFG Register (Offset = Ch) [Reset = 0000000Fh]

SATCFG is shown in Table 20-118.

Return to the Summary Table.

Saturation Configuration

Table 20-118. SATCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	LIMIT	R/W	Fh	Saturation limit. The flag STAT.SAT is set when the TDC counter saturates. Values not enumerated are not supported 3h = Result bit 12: TDC conversion saturates and stops when RESULT.VALUE[12] is set. 4h = Result bit 13: TDC conversion saturates and stops when RESULT.VALUE[13] is set. 5h = Result bit 14: TDC conversion saturates and stops when RESULT.VALUE[14] is set. 6h = Result bit 15: TDC conversion saturates and stops when RESULT.VALUE[15] is set. 7h = Result bit 16: TDC conversion saturates and stops when RESULT.VALUE[16] is set. 8h = Result bit 17: TDC conversion saturates and stops when RESULT.VALUE[17] is set. 9h = Result bit 18: TDC conversion saturates and stops when RESULT.VALUE[18] is set. Ah = Result bit 19: TDC conversion saturates and stops when RESULT.VALUE[19] is set. Bh = Result bit 20: TDC conversion saturates and stops when RESULT.VALUE[20] is set. Ch = Result bit 21: TDC conversion saturates and stops when RESULT.VALUE[21] is set. Dh = Result bit 22: TDC conversion saturates and stops when RESULT.VALUE[22] is set. Eh = Result bit 23: TDC conversion saturates and stops when RESULT.VALUE[23] is set. Fh = Result bit 24: TDC conversion saturates and stops when RESULT.VALUE[24] is set.

20.8.5.5 TRIGSRC Register (Offset = 10h) [Reset = 00000000h]

TRIGSRC is shown in Table 20-119.

Return to the Summary Table.

Trigger Source
Select source and polarity for TDC start and stop events. See the Technical Reference Manual for event timing requirements.

Table 20-119. TRIGSRC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	STOP_POL	R/W	0h	Polarity of stop source. Change only while STAT.STATE is IDLE. 0h = TDC conversion stops when high level is detected. 1h = TDC conversion stops when low level is detected.

Table 20-119. TRIGSRC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	STOP_SRC	R/W	0h	Select stop source from the asynchronous AUX event bus. Change only while STAT.STATE is IDLE. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE

Table 20-119. TRIGSRC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Eh = Select TDC Prescaler event which is generated by configuration of PRECTL. 3Fh = No event.
7	RESERVED	R	0h	Reserved
6	START_POL	R/W	0h	Polarity of start source. Change only while STAT.STATE is IDLE. 0h = TDC conversion starts when high level is detected. 1h = TDC conversion starts when low level is detected.

Table 20-119. TRIGSRC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	START_SRC	R/W	0h	Select start source from the asynchronous AUX event bus. Change only while STAT.STATE is IDLE. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE

Table 20-119. TRIGSRC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Eh = Select TDC Prescaler event which is generated by configuration of PRECTL. 3Fh = No event.

20.8.5.6 TRIGCNT Register (Offset = 14h) [Reset = 00000000h]

TRIGCNT is shown in Table 20-120.

Return to the Summary Table.

Trigger Counter
Stop-counter control and status.

Table 20-120. TRIGCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CNT	R/W	0h	Number of stop events to ignore when AUX_TDC:TRIGCNTCFG.EN is 1. Read CNT to get the remaining number of stop events to ignore during a TDC measurement. Write CNT to update the remaining number of stop events to ignore during a TDC measurement. The TDC measurement ignores updates of CNT if there are no more stop events left to ignore. When AUX_TDC:TRIGCNTCFG.EN is 1, TRIGCNTLOAD.CNT is loaded into CNT at the start of the measurement.

20.8.5.7 TRIGCNTLOAD Register (Offset = 18h) [Reset = 00000000h]

TRIGCNTLOAD is shown in Table 20-121.

Return to the Summary Table.

Trigger Counter Load
Stop-counter load.

Table 20-121. TRIGCNTLOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CNT	R/W	0h	Number of stop events to ignore when AUX_TDC:TRIGCNTCFG.EN is 1. To measure frequency of an event source: - Set start event equal to stop event. - Set CNT to number of periods to measure. Both 0 and 1 values measures a single event source period. To measure pulse width of an event source: - Set start event source equal to stop event source. - Select different polarity for start and stop event. - Set CNT to 0. To measure time from the start event to the Nth stop event when N > 1: - Select different start and stop event source. - Set CNT to (N-1). See the Technical Reference Manual for event timing requirements. When AUX_TDC:TRIGCNTCFG.EN is 1, CNT is loaded into TRIGCNT.CNT at the start of the measurement.

20.8.5.8 TRIGCNTCFG Register (Offset = 1Ch) [Reset = 00000000h]

TRIGCNTCFG is shown in Table 20-122.

Return to the Summary Table.

Trigger Counter Configuration
Stop-counter configuration.

Table 20-122. TRIGCNTCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Enable stop-counter. 0: Disable stop-counter. 1: Enable stop-counter. Change only while STAT.STATE is IDLE.

20.8.5.9 PRECTL Register (Offset = 20h) [Reset = 0000003Fh]

PRECTL is shown in Table 20-123.

Return to the Summary Table.

Prescaler Control
The prescaler can be used to count events that are faster than the AUX bus rate.
It can be used to:
- count pulses on a specified event from the asynchronous event bus.
- prescale a specified event from the asynchronous event bus.
To use the prescaler output as an event source in TDC measurements you must set both TRIGSRC.START_SRC and TRIGSRC.STOP_SRC to AUX_TDC_PRE.
It is recommended to use the prescaler when the signal frequency to measure exceeds 1/10th of the AUX bus rate.

Table 20-123. PRECTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	RESET_N	R/W	0h	Prescaler reset. 0: Reset prescaler. 1: Release reset of prescaler. AUX_TDC_PRE event becomes 0 when you reset the prescaler.
6	RATIO	R/W	0h	Prescaler ratio. This controls how often the AUX_TDC_PRE event is generated by the prescaler. 0h = Prescaler divides input by 16. AUX_TDC_PRE event has a rising edge for every 16 rising edges of the input. AUX_TDC_PRE event toggles on every 8th rising edge of the input. 1h = Prescaler divides input by 64. AUX_TDC_PRE event has a rising edge for every 64 rising edges of the input. AUX_TDC_PRE event toggles on every 32nd rising edge of the input.

Table 20-123. PRECTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	SRC	R/W	3Fh	Prescaler event source. Select an event from the asynchronous AUX event bus to connect to the prescaler input. Configure only while RESET_N is 0. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY

Table 20-123. PRECTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.

20.8.5.10 PRECNTR Register (Offset = 24h) [Reset = 00000000h]

PRECNTR is shown in Table 20-124.

Return to the Summary Table.

Prescaler Counter

Table 20-124. PRECNTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CNT	R/W	0h	Prescaler counter value. Write a value to CNT to capture the value of the 16-bit prescaler counter into CNT. Read CNT to get the captured value. The read value gets 1 LSB uncertainty if the event source level rises when you release the reset. The read value gets 1 LSB uncertainty if the event source level rises when you capture the prescaler counter. Please note the following: - The prescaler counter is reset to 2 by PRECTL.RESET_N. - The captured value is 2 when the number of rising edges on prescaler input is less than 3. Otherwise, captured value equals number of event pulses - 1.

20.8.6 AUX_TIMER01 Registers

Table 20-125 lists the memory-mapped registers for the AUX_TIMER01 registers. All register offset addresses not listed in Table 20-125 should be considered as reserved locations and the register contents should not be modified.

Table 20-125. AUX_TIMER01 Registers

Offset	Acronym	Register Name	Section
0h	T0CFG	Timer 0 Configuration	Section 20.8.6.1
4h	T0CTL	Timer 0 Control	Section 20.8.6.2
8h	T0TARGET	Timer 0 Target	Section 20.8.6.3
Ch	T0CNTR	Timer 0 Counter	Section 20.8.6.4
10h	T1CFG	Timer 1 Configuration	Section 20.8.6.5
14h	T1CTL	Timer 1 Control	Section 20.8.6.6
18h	T1TARGET	Timer 1 Target	Section 20.8.6.7
1Ch	T1CNTR	Timer 1 Counter	Section 20.8.6.8

Complex bit access types are encoded to fit into small table cells. Table 20-126 shows the codes that are used for access types in this section.

Table 20-126. AUX_TIMER01 Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.6.1 T0CFG Register (Offset = 0h) [Reset = 00000000h]

T0CFG is shown in Table 20-127.

Return to the Summary Table.

Timer 0 Configuration

Table 20-127. T0CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	TICK_SRC_POL	R/W	0h	Tick source polarity for Timer 0. 0h = Count on rising edges of TICK_SRC. 1h = Count on falling edges of TICK_SRC.

Table 20-127. T0CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	TICK_SRC	R/W	0h	Select Timer 0 tick source from the synchronous event bus. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = No event. 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Eh = AUX_EVCTL:EVSTAT3.AUX_DAC_HOLD_ACTIVE 3Fh = AUX_EVCTL:EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY

Table 20-127. T0CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-4	PRE	R/W	0h	Prescaler division ratio is 2^{PRE}: 0x0: Divide by 1. 0x1: Divide by 2. 0x2: Divide by 4. ... 0xF: Divide by 32,768.
3-2	RESERVED	R	0h	Reserved
1	MODE	R/W	0h	Timer 0 mode. Configure source for Timer 0 prescaler. 0h = Use clock as source for prescaler. Note that AUX_SYSIF:PEROPRATE.TIMER01_OP_RATE sets the clock frequency. 1h = Use event set by TICK_SRC as source for prescaler.
0	RELOAD	R/W	0h	Timer 0 reload mode. 0h = Manual mode. Timer 0 stops and T0CTL.EN becomes 0 when the counter value becomes equal to or greater than T0TARGET.VALUE. 1h = Continuous mode. Timer 0 restarts when the counter value becomes equal to or greater than (T0TARGET.VALUE - 1).

20.8.6.2 T0CTL Register (Offset = 4h) [Reset = 00000000h]

T0CTL is shown in Table 20-128.

Return to the Summary Table.

Timer 0 Control

Table 20-128. T0CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Timer 0 enable. 0: Disable Timer 0. 1: Enable Timer 0. The counter restarts from 0 when you enable Timer 0.

20.8.6.3 T0TARGET Register (Offset = 8h) [Reset = 00000000h]

T0TARGET is shown in Table 20-129.

Return to the Summary Table.

Timer 0 Target

Table 20-129. T0TARGET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Timer 0 target value. Manual Reload Mode: - Timer 0 increments until the counter value becomes equal to or greater than VALUE. - AUX_TIMER0_EV pulses high for 1 peripheral clock period when the counter value is equal to or greater than VALUE. Note: When VALUE is 0, Timer 0 counts to 1. AUX_TIMER0_EV pulses high for 1 peripheral clock period. Continuous Reload Mode: - Timer 0 increments until the counter value becomes equal to or greater than (VALUE - 1), then restarts from 0. - AUX_TIMER0_EV pulses high for 1 peripheral clock period when the counter value is 0, except for when you enable the timer. Note: When VALUE is less than 2, Timer 0 counter value remains 0. AUX_TIMER0_EV goes high and remains high 1 peripheral clock period after you enable the timer. It is allowed to update the VALUE while the timer runs.

20.8.6.4 T0CNTR Register (Offset = Ch) [Reset = 00000000h]

T0CNTR is shown in Table 20-130.

Return to the Summary Table.

Timer 0 Counter

Table 20-130. T0CNTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Timer 0 counter value.

20.8.6.5 T1CFG Register (Offset = 10h) [Reset = 00000000h]

T1CFG is shown in Table 20-131.

Return to the Summary Table.

Timer 1 Configuration

Table 20-131. T1CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	TICK_SRC_POL	R/W	0h	Tick source polarity for Timer 1. 0h = Count on rising edges of TICK_SRC. 1h = Count on falling edges of TICK_SRC.

Table 20-131. T1CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	TICK_SRC	R/W	0h	Select Timer 1 tick source from the synchronous event bus. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = No event. 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Eh = AUX_EVCTL:EVSTAT3.AUX_DAC_HOLD_ACTIVE 3Fh = AUX_EVCTL:EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY

Table 20-131. T1CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-4	PRE	R/W	0h	Prescaler division ratio is 2^{PRE}: 0x0: Divide by 1. 0x1: Divide by 2. 0x2: Divide by 4. ... 0xF: Divide by 32,768.
3-2	RESERVED	R	0h	Reserved
1	MODE	R/W	0h	Timer 1 mode. Configure source for Timer 1 prescaler. 0h = Use clock as source for prescaler. Note that AUX_SYSIF:PEROPRATE.TIMER01_OP_RATE sets the clock frequency. 1h = Use event set by TICK_SRC as source for prescaler.
0	RELOAD	R/W	0h	Timer 1 reload mode. 0h = Manual mode. Timer 1 stops and T1CTL.EN becomes 0 when the counter value becomes equal to or greater than T1TARGET.VALUE. 1h = Continuous mode. Timer 1 restarts when the counter value becomes equal to or greater than (T1TARGET.VALUE - 1).

20.8.6.6 T1CTL Register (Offset = 14h) [Reset = 00000000h]

T1CTL is shown in Table 20-132.

Return to the Summary Table.

Timer 1 Control

Table 20-132. T1CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Timer 1 enable. 0: Disable Timer 1. 1: Enable Timer 1. The counter restarts from 0 when you enable Timer 1.

20.8.6.7 T1TARGET Register (Offset = 18h) [Reset = 00000000h]

T1TARGET is shown in Table 20-133.

Return to the Summary Table.

Timer 1 Target
Timer 1 counter target value

Table 20-133. T1TARGET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Timer 1 target value. Manual Reload Mode: - Timer 1 increments until the counter value becomes equal to or greater than VALUE. - AUX_TIMER1_EV pulses high for 1 peripheral clock period when the counter value is equal to or greater than VALUE. Note: When VALUE is 0, Timer 1 counts to 1. AUX_TIMER1_EV pulses high for 1 peripheral clock period. Continuous Reload Mode: - Timer 1 increments until the counter value becomes equal to or greater than (VALUE - 1), then restarts from 0. - AUX_TIMER1_EV pulses high for 1 peripheral clock period when the counter value is 0, except for when you enable the timer. Note: When VALUE is less than 2, Timer 1 counter value remains 0. AUX_TIMER1_EV goes high and remains high 1 peripheral clock period after you enable the timer. It is allowed to update the VALUE while the timer runs.

20.8.6.8 T1CNTR Register (Offset = 1Ch) [Reset = 00000000h]

T1CNTR is shown in Table 20-134.

Return to the Summary Table.

Timer 1 Counter

Table 20-134. T1CNTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Timer 1 counter value.

20.8.7 AUX_TIMER2 Registers

Table 20-135 lists the memory-mapped registers for the AUX_TIMER2 registers. All register offset addresses not listed in Table 20-135 should be considered as reserved locations and the register contents should not be modified.

Table 20-135. AUX_TIMER2 Registers

Offset	Acronym	Register Name	Section
0h	CTL	Timer Control	Section 20.8.7.1
4h	TARGET	Target	Section 20.8.7.2
8h	SHDWTARGET	Shadow Target	Section 20.8.7.3
Ch	CNTR	Counter	Section 20.8.7.4
10h	PRECFG	Clock Prescaler Configuration	Section 20.8.7.5
14h	EVCTL	Event Control	Section 20.8.7.6
18h	PULSETRIG	Pulse Trigger	Section 20.8.7.7
80h	CH0EVCFG	Channel 0 Event Configuration	Section 20.8.7.8
84h	CH0CCFG	Channel 0 Capture Configuration	Section 20.8.7.9
88h	CH0PCC	Channel 0 Pipeline Capture Compare	Section 20.8.7.10
8Ch	CH0CC	Channel 0 Capture Compare	Section 20.8.7.11
90h	CH1EVCFG	Channel 1 Event Configuration	Section 20.8.7.12
94h	CH1CCFG	Channel 1 Capture Configuration	Section 20.8.7.13
98h	CH1PCC	Channel 1 Pipeline Capture Compare	Section 20.8.7.14
9Ch	CH1CC	Channel 1 Capture Compare	Section 20.8.7.15
A0h	CH2EVCFG	Channel 2 Event Configuration	Section 20.8.7.16
A4h	CH2CCFG	Channel 2 Capture Configuration	Section 20.8.7.17
A8h	CH2PCC	Channel 2 Pipeline Capture Compare	Section 20.8.7.18
ACh	CH2CC	Channel 2 Capture Compare	Section 20.8.7.19
B0h	CH3EVCFG	Channel 3 Event Configuration	Section 20.8.7.20
B4h	CH3CCFG	Channel 3 Capture Configuration	Section 20.8.7.21
B8h	CH3PCC	Channel 3 Pipeline Capture Compare	Section 20.8.7.22
BCh	CH3CC	Channel 3 Capture Compare	Section 20.8.7.23

Complex bit access types are encoded to fit into small table cells. Table 20-136 shows the codes that are used for access types in this section.

Table 20-136. AUX_TIMER2 Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.7.1 CTL Register (Offset = 0h) [Reset = 00000000h]

CTL is shown in Table 20-137.

Return to the Summary Table.

Timer Control

Table 20-137. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	CH3_RESET	R/W	0h	Channel 3 reset. 0: No effect. 1: Reset CH3CC, CH3PCC, CH3EVCFG, and CH3CCFG. Read returns 0.
5	CH2_RESET	R/W	0h	Channel 2 reset. 0: No effect. 1: Reset CH2CC, CH2PCC, CH2EVCFG, and CH2CCFG. Read returns 0.
4	CH1_RESET	R/W	0h	Channel 1 reset. 0: No effect. 1: Reset CH1CC, CH1PCC, CH1EVCFG, and CH1CCFG. Read returns 0.
3	CH0_RESET	R/W	0h	Channel 0 reset. 0: No effect. 1: Reset CH0CC, CH0PCC, CH0EVCFG, and CH0CCFG. Read returns 0.
2	TARGET_EN	R/W	0h	Select counter target value. You must select TARGET to use shadow target functionality. 0h = 65535 1h = TARGET.VALUE
1-0	MODE	R/W	0h	Timer mode control. The timer restarts from 0 when you set MODE to UP_ONCE, UP_PER, or UPDWN_PER. When you write MODE all internally queued updates to [CHnCC.*] and TARGET clear. 0h = Disable timer. Updates to counter, channels, and events stop. 1h = Count up once. The timer increments from 0 to target value, then stops and sets MODE to DIS. 2h = Count up periodically. The timer increments from 0 to target value, repeatedly. Period = (target value + 1) * timer clock period 3h = Count up and down periodically. The timer counts from 0 to target value and back to 0, repeatedly. Period = (target value * 2) * timer clock period

20.8.7.2 TARGET Register (Offset = 4h) [Reset = 00000000h]

TARGET is shown in Table 20-138.

Return to the Summary Table.

Target
User defined counter target.

Table 20-138. TARGET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	16 bit user defined counter target value, which is used when selected by CTL.TARGET_EN.

20.8.7.3 SHDWTARGET Register (Offset = 8h) [Reset = 00000000h]

SHDWTARGET is shown in Table 20-139.

Return to the Summary Table.

Shadow Target

Table 20-139. SHDWTARGET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Target value for next counter period. The timer copies VALUE to TARGET.VALUE when CNTR.VALUE becomes 0. The copy does not happen when you restart the timer. This is useful to avoid period jitter in PWM applications with time-varying period, sometimes referenced as phase corrected PWM.

20.8.7.4 CNTR Register (Offset = Ch) [Reset = 00000000h]

CNTR is shown in Table 20-140.

Return to the Summary Table.

Counter

Table 20-140. CNTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	16 bit current counter value.

20.8.7.5 PRECFG Register (Offset = 10h) [Reset = 00000000h]

PRECFG is shown in Table 20-141.

Return to the Summary Table.

Clock Prescaler Configuration

Table 20-141. PRECFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CLKDIV	R/W	0h	Clock division. CLKDIV determines the timer clock frequency for counter, synchronization, and timer event updates. The timer clock frequency is the clock selected by AUX_SYSIF:TIMER2CLKCTL.SRC divided by (CLKDIV + 1). This inverse is the timer clock period. 0x00: Divide by 1. 0x01: Divide by 2. ... 0xFF: Divide by 256.

20.8.7.6 EVCTL Register (Offset = 14h) [Reset = 00000000h]

EVCTL is shown in Table 20-142.

Return to the Summary Table.

Event Control
Set and clear individual events manually. Manual update of an event takes priority over automatic channel updates to the same event. You cannot set and clear an event at the same time, such requests will be neglected. An event can be automatically cleared, set, toggled, or pulsed by each channel, listed in decreasing order of priority. The action with highest priority happens when multiple channels want to update an event at the same time.
The four events connect to the asynchronous AUX event bus:
- Event 0 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0.
- Event 1 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1.
- Event 2 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2.
- Event 3 connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3.

Table 20-142. EVCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV3_SET	W	0h	Set event 3. Write 1 to set event 3.
6	EV3_CLR	W	0h	Clear event 3. Write 1 to clear event 3.
5	EV2_SET	W	0h	Set event 2. Write 1 to set event 2.
4	EV2_CLR	W	0h	Clear event 2. Write 1 to clear event 2.
3	EV1_SET	W	0h	Set event 1. Write 1 to set event 1.
2	EV1_CLR	W	0h	Clear event 1. Write 1 to clear event 1.
1	EV0_SET	W	0h	Set event 0. Write 1 to set event 0.
0	EV0_CLR	W	0h	Clear event 0. Write 1 to clear event 0.

20.8.7.7 PULSETRIG Register (Offset = 18h) [Reset = 00000000h]

PULSETRIG is shown in Table 20-143.

Return to the Summary Table.

Pulse Trigger

Table 20-143. PULSETRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	TRIG	W	0h	Pulse trigger. Write 1 to generate a pulse to AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE. Pulse width equals the duty cycle of AUX_SYSIF:TIMER2CLKCTL.SRC.

20.8.7.8 CH0EVCFG Register (Offset = 80h) [Reset = 00000000h]

CH0EVCFG is shown in Table 20-144.

Return to the Summary Table.

Channel 0 Event Configuration
This register configures channel function and enables event outputs.
Each channel has an edge-detection circuit with memory. The circuit is:
- enabled while CCACT selects a capture function and CTL.MODE is different from DIS.
- flushed while CCACT selects a capture function and you change CTL.MODE from DIS to another mode.
The flush action uses two AUX_SYSIF:TIMER2CLKCTL.SRC clock periods. It prevents capture events caused by expired signal values stored in the edge-detection circuit.

Table 20-144. CH0EVCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV3_GEN	R/W	0h	Event 3 enable. 0: Channel 0 does not control event 3. 1: Channel 0 controls event 3. When 0 < CCACT < 8, EV3_GEN becomes zero after a capture or compare event.
6	EV2_GEN	R/W	0h	Event 2 enable. 0: Channel 0 does not control event 2. 1: Channel 0 controls event 2. When 0 < CCACT < 8, EV2_GEN becomes zero after a capture or compare event.
5	EV1_GEN	R/W	0h	Event 1 enable. 0: Channel 0 does not control event 1. 1: Channel 0 controls event 1. When 0 < CCACT < 8, EV1_GEN becomes zero after a capture or compare event.
4	EV0_GEN	R/W	0h	Event 0 enable. 0: Channel 0 does not control event 0. 1: Channel 0 controls event 0. When 0 < CCACT < 8, EV0_GEN becomes zero after a capture or compare event.

Table 20-144. CH0EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CCACT	R/W	0h	Capture-Compare action. Capture-Compare action defines 15 different channel functions that utilize capture, compare, and zero events. 0h = Disable channel. 1h = Set on capture, and then disable channel. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH0CC.VALUE. - Disable channel. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Set CCACT to SET_ON_CAPT with no event enable. - Configure CH0CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you set CCACT to SET_ON_CAPT_DIS. Event enable is optional. These steps prevent capture events caused by expired signal values in edge-detection circuit. 2h = Clear on zero, toggle on compare, and then disable channel. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are set when CH0CC.VALUE = 0 and CNTR.VALUE = 0. 3h = Set on zero, toggle on compare, and then disable channel. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are cleared when CH0CC.VALUE = 0 and CNTR.VALUE = 0. 4h = Clear on compare, and then disable channel. Channel function sequence: - Clear enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. 5h = Set on compare, and then disable channel. Channel function sequence: - Set enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. 6h = Toggle on compare, and then disable channel. Channel function sequence: - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. 7h = Pulse on compare, and then disable channel. Channel function sequence: - Pulse enabled events when CH0CC.VALUE = CNTR.VALUE. - Disable channel. The event is high for two timer clock periods. 8h = Period and pulse width measurement. Continuously capture period and pulse width of the signal selected by CH0CCFG.CAPT_SRC relative to the signal edge given by CH0CCFG.EDGE. Set enabled events when CH0CC.VALUE contains signal period and CH0PCC.VALUE contains signal pulse width. Notes: - Make sure that you configure CH0CCFG.CAPT_SRC and CCACT when CTL.MODE equals DIS, then set CTL.MODE to UP_ONCE or UP_PER. - The counter restarts in the selected timer mode when CH0CC.VALUE contains the signal period. - If more than one channel uses this function, the channels will perform this function one at a time. The channel with lowest number has priority and performs the function first. Next measurement starts

Table 20-144. CH0EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				when current measurement completes successfully or times out. A timeout occurs when counter equals target. - If you want to observe a timeout event configure another channel to SET_ON_CAPT. Signal property requirements: - Signal Period >= 2 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal Period <= 65535 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal low and high phase >= (1 + PRECFG.CLKDIV) * timer clock period. 9h = Set on capture repeatedly. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH0CC.VALUE. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Select this function with no event enable. - Configure CH0CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you enable events. These steps prevent capture events caused by expired signal values in edge-detection circuit. Ah = Clear on zero, toggle on compare repeatedly. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. Set CTL.MODE to UPDWN_PER for center-aligned PWM generation. Duty cycle is given by: When CH0CC.VALUE <= TARGET.VALUE: Duty cycle = 1 - (CH0CC.VALUE / TARGET.VALUE). When CH0CC.VALUE > TARGET.VALUE: Duty cycle = 0. Enabled events are set when CH0CC.VALUE = 0 and CNTR.VALUE = 0. Bh = Set on zero, toggle on compare repeatedly. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. Set CTL.MODE to UP_PER for edge-aligned PWM generation. Duty cycle is given by: When CH0CC.VALUE <= TARGET.VALUE: Duty cycle = CH0CC.VALUE / (TARGET.VALUE + 1). When CH0CC.VALUE > TARGET.VALUE: Duty cycle = 1. Enabled events are cleared when CH0CC.VALUE = 0 and CNTR.VALUE = 0. Ch = Clear on compare repeatedly. Channel function sequence: - Clear enabled events when CH0CC.VALUE = CNTR.VALUE. Dh = Set on compare repeatedly. Channel function sequence: - Set enabled events when CH0CC.VALUE = CNTR.VALUE. Eh = Toggle on compare repeatedly. Channel function sequence: - Toggle enabled events when CH0CC.VALUE = CNTR.VALUE. Fh = Pulse on compare repeatedly. Channel function sequence: - Pulse enabled events when CH0CC.VALUE = CNTR.VALUE. The event is high for two timer clock periods.

20.8.7.9 CH0CCFG Register (Offset = 84h) [Reset = 00000000h]

CH0CCFG is shown in Table 20-145.

Return to the Summary Table.

Channel 0 Capture Configuration

Table 20-145. CH0CCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved

Table 20-145. CH0CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-1	CAPT_SRC	R/W	0h	Select capture signal source from the asynchronous AUX event bus. The selected signal enters the edge-detection circuit. False capture events can occur when: - the edge-detection circuit contains expired signal samples and the circuit is enabled without flush as described in CH0EVCFG - this register is reconfigured while CTL.MODE is different from DIS. You can avoid false capture events. When wanted channel function is: - SET_ON_CAPT_DIS, see description for SET_ON_CAPT_DIS in CH0EVCFG.CCACT. - SET_ON_CAPT, see description for SET_ON_CAPT in CH0EVCFG.CCACT. - PER_PULSE_WIDTH_MEAS, see description for PER_PULSE_WIDTH_MEAS in CH0EVCFG.CCACT. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0

Table 20-145. CH0CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.
0	EDGE	R/W	0h	Edge configuration. Channel captures counter value at selected edge on signal source selected by CAPT_SRC. See CH0EVCFG.CCACT. 0h = Capture CNTR.VALUE at falling edge of CAPT_SRC. 1h = Capture CNTR.VALUE at rising edge of CAPT_SRC.

20.8.7.10 CH0PCC Register (Offset = 88h) [Reset = 00000000h]

CH0PCC is shown in Table 20-146.

Return to the Summary Table.

Channel 0 Pipeline Capture Compare

Table 20-146. CH0PCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Pipeline Capture Compare value. 16-bit user defined pipeline compare value or channel-updated capture value. Compare mode: An update of VALUE will be transferred to CH0CC.VALUE when the next CNTR.VALUE is zero and CTL.MODE is different from DIS. This is useful for PWM generation and prevents jitter on the edges of the generated signal. Capture mode: When CH0EVCFG.CCACT equals PER_PULSE_WIDTH_MEAS then VALUE contains the width of the low or high phase of the selected signal. This is specified by CH0CCFG.EDGE and CH0CCFG.CAPT_SRC.

20.8.7.11 CH0CC Register (Offset = 8Ch) [Reset = 00000000h]

CH0CC is shown in Table 20-147.

Return to the Summary Table.

Channel 0 Capture Compare

Table 20-147. CH0CC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Capture Compare value. 16-bit user defined compare value or channel-updated capture value. Compare mode: VALUE is compared against CNTR.VALUE and an event is generated as specified by CH0EVCFG.CCACT when these are equal. Capture mode: The current counter value is stored in VALUE when a capture event occurs. CH0EVCFG.CCACT determines if VALUE is a signal period or a regular capture value.

20.8.7.12 CH1EVCFG Register (Offset = 90h) [Reset = 00000000h]

CH1EVCFG is shown in Table 20-148.

Return to the Summary Table.

Channel 1 Event Configuration
This register configures channel function and enables event outputs.
Each channel has an edge-detection circuit with memory. The circuit is:
- enabled while CCACT selects a capture function and CTL.MODE is different from DIS.
- flushed while CCACT selects a capture function and you change CTL.MODE from DIS to another mode.
The flush action uses two AUX_SYSIF:TIMER2CLKCTL.SRC clock periods. It prevents capture events caused by expired signal values stored in the edge-detection circuit.

Table 20-148. CH1EVCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV3_GEN	R/W	0h	Event 3 enable. 0: Channel 1 does not control event 3. 1: Channel 1 controls event 3. When 0 < CCACT < 8, EV3_GEN becomes zero after a capture or compare event.
6	EV2_GEN	R/W	0h	Event 2 enable. 0: Channel 1 does not control event 2. 1: Channel 1 controls event 2. When 0 < CCACT < 8, EV2_GEN becomes zero after a capture or compare event.
5	EV1_GEN	R/W	0h	Event 1 enable. 0: Channel 1 does not control event 1. 1: Channel 1 controls event 1. When 0 < CCACT < 8, EV1_GEN becomes zero after a capture or compare event.
4	EV0_GEN	R/W	0h	Event 0 enable. 0: Channel 1 does not control event 0. 1: Channel 1 controls event 0. When 0 < CCACT < 8, EV0_GEN becomes zero after a capture or compare event.

Table 20-148. CH1EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CCACT	R/W	0h	Capture-Compare action. Capture-Compare action defines 15 different channel functions that utilize capture, compare, and zero events. 0h = Disable channel. 1h = Set on capture, and then disable channel. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH1CC.VALUE. - Disable channel. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Set CCACT to SET_ON_CAPT with no event enable. - Configure CH1CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you set CCACT to SET_ON_CAPT_DIS. Event enable is optional. These steps prevent capture events caused by expired signal values in edge-detection circuit. 2h = Clear on zero, toggle on compare, and then disable channel. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are set when CH1CC.VALUE = 0 and CNTR.VALUE = 0. 3h = Set on zero, toggle on compare, and then disable channel. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are cleared when CH1CC.VALUE = 0 and CNTR.VALUE = 0. 4h = Clear on compare, and then disable channel. Channel function sequence: - Clear enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. 5h = Set on compare, and then disable channel. Channel function sequence: - Set enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. 6h = Toggle on compare, and then disable channel. Channel function sequence: - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. 7h = Pulse on compare, and then disable channel. Channel function sequence: - Pulse enabled events when CH1CC.VALUE = CNTR.VALUE. - Disable channel. The event is high for two timer clock periods. 8h = Period and pulse width measurement. Continuously capture period and pulse width of the signal selected by CH1CCFG.CAPT_SRC relative to the signal edge given by CH1CCFG.EDGE. Set enabled events when CH1CC.VALUE contains signal period and CH1PCC.VALUE contains signal pulse width. Notes: - Make sure that you configure CH1CCFG.CAPT_SRC and CCACT when CTL.MODE equals DIS, then set CTL.MODE to UP_ONCE or UP_PER. - The counter restarts in the selected timer mode when CH1CC.VALUE contains the signal period. - If more than one channel uses this function, the channels will perform this function one at a time. The channel with lowest number has priority and performs the function first. Next measurement starts

Table 20-148. CH1EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				when current measurement completes successfully or times out. A timeout occurs when counter equals target. - If you want to observe a timeout event configure another channel to SET_ON_CAPT. Signal property requirements: - Signal Period >= 2 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal Period <= 65535 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal low and high phase >= (1 + PRECFG.CLKDIV) * timer clock period. 9h = Set on capture repeatedly. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH1CC.VALUE. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Select this function with no event enable. - Configure CH1CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you enable events. These steps prevent capture events caused by expired signal values in edge-detection circuit. Ah = Clear on zero, toggle on compare repeatedly. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. Set CTL.MODE to UPDWN_PER for center-aligned PWM generation. Duty cycle is given by: When CH1CC.VALUE <= TARGET.VALUE: Duty cycle = 1 - (CH1CC.VALUE / TARGET.VALUE). When CH1CC.VALUE > TARGET.VALUE: Duty cycle = 0. Enabled events are set when CH1CC.VALUE = 0 and CNTR.VALUE = 0. Bh = Set on zero, toggle on compare repeatedly. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. Set CTL.MODE to UP_PER for edge-aligned PWM generation. Duty cycle is given by: When CH1CC.VALUE <= TARGET.VALUE: Duty cycle = CH1CC.VALUE / (TARGET.VALUE + 1). When CH1CC.VALUE > TARGET.VALUE: Duty cycle = 1. Enabled events are cleared when CH1CC.VALUE = 0 and CNTR.VALUE = 0. Ch = Clear on compare repeatedly. Channel function sequence: - Clear enabled events when CH1CC.VALUE = CNTR.VALUE. Dh = Set on compare repeatedly. Channel function sequence: - Set enabled events when CH1CC.VALUE = CNTR.VALUE. Eh = Toggle on compare repeatedly. Channel function sequence: - Toggle enabled events when CH1CC.VALUE = CNTR.VALUE. Fh = Pulse on compare repeatedly. Channel function sequence: - Pulse enabled events when CH1CC.VALUE = CNTR.VALUE. The event is high for two timer clock periods.

20.8.7.13 CH1CCFG Register (Offset = 94h) [Reset = 00000000h]

CH1CCFG is shown in Table 20-149.

Return to the Summary Table.

Channel 1 Capture Configuration

Table 20-149. CH1CCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved

Table 20-149. CH1CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-1	CAPT_SRC	R/W	0h	Select capture signal source from the asynchronous AUX event bus. The selected signal enters the edge-detection circuit. False capture events can occur when: - the edge-detection circuit contains expired signal samples and the circuit is enabled without flush as described in CH1EVCFG - this register is reconfigured while CTL.MODE is different from DIS. You can avoid false capture events. When wanted channel function is: - SET_ON_CAPT_DIS, see description for SET_ON_CAPT_DIS in CH1EVCFG.CCACT. - SET_ON_CAPT, see description for SET_ON_CAPT in CH1EVCFG.CCACT. - PER_PULSE_WIDTH_MEAS, see description for PER_PULSE_WIDTH_MEAS in CH1EVCFG.CCACT. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0

Table 20-149. CH1CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.
0	EDGE	R/W	0h	Edge configuration. Channel captures counter value at selected edge on signal source selected by CAPT_SRC. See CH1EVCFG.CCACT. 0h = Capture CNTR.VALUE at falling edge of CAPT_SRC. 1h = Capture CNTR.VALUE at rising edge of CAPT_SRC.

20.8.7.14 CH1PCC Register (Offset = 98h) [Reset = 00000000h]

CH1PCC is shown in Table 20-150.

Return to the Summary Table.

Channel 1 Pipeline Capture Compare

Table 20-150. CH1PCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Pipeline Capture Compare value. 16-bit user defined pipeline compare value or channel-updated capture value. Compare mode: An update of VALUE will be transferred to CH1CC.VALUE when the next CNTR.VALUE is zero and CTL.MODE is different from DIS. This is useful for PWM generation and prevents jitter on the edges of the generated signal. Capture mode: When CH1EVCFG.CCACT equals PER_PULSE_WIDTH_MEAS then VALUE contains the width of the low or high phase of the selected signal. This is specified by CH1CCFG.EDGE and CH1CCFG.CAPT_SRC.

20.8.7.15 CH1CC Register (Offset = 9Ch) [Reset = 00000000h]

CH1CC is shown in Table 20-151.

Return to the Summary Table.

Channel 1 Capture Compare

Table 20-151. CH1CC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Capture Compare value. 16-bit user defined compare value or channel-updated capture value. Compare mode: VALUE is compared against CNTR.VALUE and an event is generated as specified by CH1EVCFG.CCACT when these are equal. Capture mode: The current counter value is stored in VALUE when a capture event occurs. CH1EVCFG.CCACT determines if VALUE is a signal period or a regular capture value.

20.8.7.16 CH2EVCFG Register (Offset = A0h) [Reset = 00000000h]

CH2EVCFG is shown in Table 20-152.

Return to the Summary Table.

Channel 2 Event Configuration
This register configures channel function and enables event outputs.
Each channel has an edge-detection circuit with memory. The circuit is:
- enabled while CCACT selects a capture function and CTL.MODE is different from DIS.
- flushed while CCACT selects a capture function and you change CTL.MODE from DIS to another mode.
The flush action uses two AUX_SYSIF:TIMER2CLKCTL.SRC clock periods. It prevents capture events caused by expired signal values stored in the edge-detection circuit.

Table 20-152. CH2EVCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV3_GEN	R/W	0h	Event 3 enable. 0: Channel 2 does not control event 3. 1: Channel 2 controls event 3. When 0 < CCACT < 8, EV3_GEN becomes zero after a capture or compare event.
6	EV2_GEN	R/W	0h	Event 2 enable. 0: Channel 2 does not control event 2. 1: Channel 2 controls event 2. When 0 < CCACT < 8, EV2_GEN becomes zero after a capture or compare event.
5	EV1_GEN	R/W	0h	Event 1 enable. 0: Channel 2 does not control event 1. 1: Channel 2 controls event 1. When 0 < CCACT < 8, EV1_GEN becomes zero after a capture or compare event.
4	EV0_GEN	R/W	0h	Event 0 enable. 0: Channel 2 does not control event 0. 1: Channel 2 controls event 0. When 0 < CCACT < 8, EV0_GEN becomes zero after a capture or compare event.

Table 20-152. CH2EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CCACT	R/W	0h	Capture-Compare action. Capture-Compare action defines 15 different channel functions that utilize capture, compare, and zero events. 0h = Disable channel. 1h = Set on capture, and then disable channel. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH2CC.VALUE. - Disable channel. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Set to SET_ON_CAPT with no event enable. - Configure CH2CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you set to SET_ON_CAPT_DIS. Event enable is optional. These steps prevent capture events caused by expired signal values in edge-detection circuit. 2h = Clear on zero, toggle on compare, and then disable channel. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are set when CH2CC.VALUE = 0 and CNTR.VALUE = 0. 3h = Set on zero, toggle on compare, and then disable channel. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are cleared when CH2CC.VALUE = 0 and CNTR.VALUE = 0. 4h = Clear on compare, and then disable channel. Channel function sequence: - Clear enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. 5h = Set on compare, and then disable channel. Channel function sequence: - Set enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. 6h = Toggle on compare, and then disable channel. Channel function sequence: - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. 7h = Pulse on compare, and then disable channel. Channel function sequence: - Pulse enabled events when CH2CC.VALUE = CNTR.VALUE. - Disable channel. The event is high for two timer clock periods. 8h = Period and pulse width measurement. Continuously capture period and pulse width of the signal selected by CH2CCFG.CAPT_SRC relative to the signal edge given by CH2CCFG.EDGE. Set enabled events when CH2CC.VALUE contains signal period and CH2PCC.VALUE contains signal pulse width. Notes: - Make sure that you configure CH2CCFG.CAPT_SRC and CCACT when CTL.MODE equals DIS, then set CTL.MODE to UP_ONCE or UP_PER. - The counter restarts in the selected timer mode when CH2CC.VALUE contains the signal period. - If more than one channel uses this function, the channels will perform this function one at a time. The channel with lowest number has priority and performs the function first. Next measurement starts

Table 20-152. CH2EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				when current measurement completes successfully or times out. A timeout occurs when counter equals target. - If you want to observe a timeout event configure another channel to SET_ON_CAPT. Signal property requirements: - Signal Period >= 2 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal Period <= 65535 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal low and high phase >= (1 + PRECFG.CLKDIV) * timer clock period. 9h = Set on capture repeatedly. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH2CC.VALUE. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Select this function with no event enable. - Configure CH2CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you enable events. These steps prevent capture events caused by expired signal values in edge-detection circuit. Ah = Clear on zero, toggle on compare repeatedly. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. Set CTL.MODE to UPDWN_PER for center-aligned PWM generation. Duty cycle is given by: When CH2CC.VALUE <= TARGET.VALUE: Duty cycle = 1 - (CH2CC.VALUE / TARGET.VALUE). When CH2CC.VALUE > TARGET.VALUE: Duty cycle = 0. Enabled events are set when CH2CC.VALUE = 0 and CNTR.VALUE = 0. Bh = Set on zero, toggle on compare repeatedly. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. Set CTL.MODE to UP_PER for edge-aligned PWM generation. Duty cycle is given by: When CH2CC.VALUE <= TARGET.VALUE: Duty cycle = CH2CC.VALUE / (TARGET.VALUE + 1). When CH2CC.VALUE > TARGET.VALUE: Duty cycle = 1. Enabled events are cleared when CH2CC.VALUE = 0 and CNTR.VALUE = 0. Ch = Clear on compare repeatedly. Channel function sequence: - Clear enabled events when CH2CC.VALUE = CNTR.VALUE. Dh = Set on compare repeatedly. Channel function sequence: - Set enabled events when CH2CC.VALUE = CNTR.VALUE. Eh = Toggle on compare repeatedly. Channel function sequence: - Toggle enabled events when CH2CC.VALUE = CNTR.VALUE. Fh = Pulse on compare repeatedly. Channel function sequence: - Pulse enabled events when CH2CC.VALUE = CNTR.VALUE. The event is high for two timer clock periods.

20.8.7.17 CH2CCFG Register (Offset = A4h) [Reset = 00000000h]

CH2CCFG is shown in Table 20-153.

Return to the Summary Table.

Channel 2 Capture Configuration

Table 20-153. CH2CCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved

Table 20-153. CH2CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-1	CAPT_SRC	R/W	0h	Select capture signal source from the asynchronous AUX event bus. The selected signal enters the edge-detection circuit. False capture events can occur when: - the edge-detection circuit contains expired signal samples and the circuit is enabled without flush as described in CH2EVCFG - this register is reconfigured while CTL.MODE is different from DIS. You can avoid false capture events. When wanted channel function is: - SET_ON_CAPT_DIS, see description for SET_ON_CAPT_DIS in CH2EVCFG.CCACT. - SET_ON_CAPT, see description for SET_ON_CAPT in CH2EVCFG.CCACT. - PER_PULSE_WIDTH_MEAS, see description for PER_PULSE_WIDTH_MEAS in CH2EVCFG.CCACT. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0

Table 20-153. CH2CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.
0	EDGE	R/W	0h	Edge configuration. Channel captures counter value at selected edge on signal source selected by CAPT_SRC. See CH2EVCFG.CCACT. 0h = Capture CNTR.VALUE at falling edge of CAPT_SRC. 1h = Capture CNTR.VALUE at rising edge of CAPT_SRC.

20.8.7.18 CH2PCC Register (Offset = A8h) [Reset = 00000000h]

CH2PCC is shown in Table 20-154.

Return to the Summary Table.

Channel 2 Pipeline Capture Compare

Table 20-154. CH2PCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Pipeline Capture Compare value. 16-bit user defined pipeline compare value or channel-updated capture value. Compare mode: An update of VALUE will be transferred to CH2CC.VALUE when the next CNTR.VALUE is zero and CTL.MODE is different from DIS. This is useful for PWM generation and prevents jitter on the edges of the generated signal. Capture mode: When CH2EVCFG.CCACT equals PER_PULSE_WIDTH_MEAS then VALUE contains the width of the low or high phase of the selected signal. This is specified by CH2CCFG.EDGE and CH2CCFG.CAPT_SRC.

20.8.7.19 CH2CC Register (Offset = ACh) [Reset = 00000000h]

CH2CC is shown in Table 20-155.

Return to the Summary Table.

Channel 2 Capture Compare

Table 20-155. CH2CC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Capture Compare value. 16-bit user defined compare value or channel-updated capture value. Compare mode: VALUE is compared against CNTR.VALUE and an event is generated as specified by CH2EVCFG.CCACT when these are equal. Capture mode: The current counter value is stored in VALUE when a capture event occurs. CH2EVCFG.CCACT determines if VALUE is a signal period or a regular capture value.

20.8.7.20 CH3EVCFG Register (Offset = B0h) [Reset = 00000000h]

CH3EVCFG is shown in Table 20-156.

Return to the Summary Table.

Channel 3 Event Configuration
This register configures channel function and enables event outputs.
Each channel has an edge-detection circuit with memory. The circuit is:
- enabled while CCACT selects a capture function and CTL.MODE is different from DIS.
- flushed while CCACT selects a capture function and you change CTL.MODE from DIS to another mode.
The flush action uses two AUX_SYSIF:TIMER2CLKCTL.SRC clock periods. It prevents capture events caused by expired signal values stored in the edge-detection circuit.

Table 20-156. CH3EVCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EV3_GEN	R/W	0h	Event 3 enable. 0: Channel 3 does not control event 3. 1: Channel 3 controls event 3. When 0 < CCACT < 8, EV3_GEN becomes zero after a capture or compare event.
6	EV2_GEN	R/W	0h	Event 2 enable. 0: Channel 3 does not control event 2. 1: Channel 3 controls event 2. When 0 < CCACT < 8, EV2_GEN becomes zero after a capture or compare event.
5	EV1_GEN	R/W	0h	Event 1 enable. 0: Channel 3 does not control event 1. 1: Channel 3 controls event 1. When 0 < CCACT < 8, EV1_GEN becomes zero after a capture or compare event.
4	EV0_GEN	R/W	0h	Event 0 enable. 0: Channel 3 does not control event 0. 1: Channel 3 controls event 0. When 0 < CCACT < 8, EV0_GEN becomes zero after a capture or compare event.

Table 20-156. CH3EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CCACT	R/W	0h	Capture-Compare action. Capture-Compare action defines 15 different channel functions that utilize capture, compare, and zero events. 0h = Disable channel. 1h = Set on capture, and then disable channel. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH3CC.VALUE. - Disable channel. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Set CCACT to SET_ON_CAPT with no event enable. - Configure CH3CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you set CCACT to SET_ON_CAPT_DIS. Event enable is optional. These steps prevent capture events caused by expired signal values in edge-detection circuit. 2h = Clear on zero, toggle on compare, and then disable channel. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are set when CH3CC.VALUE = 0 and CNTR.VALUE = 0. 3h = Set on zero, toggle on compare, and then disable channel. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. Enabled events are cleared when CH3CC.VALUE = 0 and CNTR.VALUE = 0. 4h = Clear on compare, and then disable channel. Channel function sequence: - Clear enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. 5h = Set on compare, and then disable channel. Channel function sequence: - Set enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. 6h = Toggle on compare, and then disable channel. Channel function sequence: - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. 7h = Pulse on compare, and then disable channel. Channel function sequence: - Pulse enabled events when CH3CC.VALUE = CNTR.VALUE. - Disable channel. The event is high for two timer clock periods. 8h = Period and pulse width measurement. Continuously capture period and pulse width of the signal selected by CH3CCFG.CAPT_SRC relative to the signal edge given by CH3CCFG.EDGE. Set enabled events when CH3CC.VALUE contains signal period and CH3PCC.VALUE contains signal pulse width. Notes: - Make sure that you configure CH3CCFG.CAPT_SRC and CCACT when CTL.MODE equals DIS, then set CTL.MODE to UP_ONCE or UP_PER. - The counter restarts in the selected timer mode when CH3CC.VALUE contains the signal period. - If more than one channel uses this function, the channels will perform this function one at a time. The channel with lowest number has priority and performs the function first. Next measurement starts

Table 20-156. CH3EVCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				when current measurement completes successfully or times out. A timeout occurs when counter equals target. - If you want to observe a timeout event configure another channel to SET_ON_CAPT. Signal property requirements: - Signal Period >= 2 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal Period <= 65535 * (1 + PRECFG.CLKDIV) * timer clock period. - Signal low and high phase >= (1 + PRECFG.CLKDIV) * timer clock period. 9h = Set on capture repeatedly. Channel function sequence: - Set enabled events on capture event and copy CNTR.VALUE to CH3CC.VALUE. Primary use scenario is to select this function before you start the timer. Follow these steps if you need to select this function while CTL.MODE is different from DIS: - Select this function with no event enable. - Configure CH3CCFG (optional). - Wait for three timer clock periods as defined in PRECFG before you enable events. These steps prevent capture events caused by expired signal values in edge-detection circuit. Ah = Clear on zero, toggle on compare repeatedly. Channel function sequence: - Clear enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. Set CTL.MODE to UPDWN_PER for center-aligned PWM generation. Duty cycle is given by: When CH3CC.VALUE <= TARGET.VALUE: Duty cycle = 1 - (CH3CC.VALUE / TARGET.VALUE). When CH3CC.VALUE > TARGET.VALUE: Duty cycle = 0. Enabled events are set when CH3CC.VALUE = 0 and CNTR.VALUE = 0. Bh = Set on zero, toggle on compare repeatedly. Channel function sequence: - Set enabled events when CNTR.VALUE = 0. - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. Set CTL.MODE to UP_PER for edge-aligned PWM generation. Duty cycle is given by: When CH3CC.VALUE <= TARGET.VALUE: Duty cycle = CH3CC.VALUE / (TARGET.VALUE + 1). When CH3CC.VALUE > TARGET.VALUE: Duty cycle = 1. Enabled events are cleared when CH3CC.VALUE = 0 and CNTR.VALUE = 0. Ch = Clear on compare repeatedly. Channel function sequence: - Clear enabled events when CH3CC.VALUE = CNTR.VALUE. Dh = Set on compare repeatedly. Channel function sequence: - Set enabled events when CH3CC.VALUE = CNTR.VALUE. Eh = Toggle on compare repeatedly. Channel function sequence: - Toggle enabled events when CH3CC.VALUE = CNTR.VALUE. Fh = Pulse on compare repeatedly. Channel function sequence: - Pulse enabled events when CH3CC.VALUE = CNTR.VALUE. The event is high for two timer clock periods.

20.8.7.21 CH3CCFG Register (Offset = B4h) [Reset = 00000000h]

CH3CCFG is shown in Table 20-157.

Return to the Summary Table.

Channel 3 Capture Configuration

Table 20-157. CH3CCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved

Table 20-157. CH3CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-1	CAPT_SRC	R/W	0h	Select capture signal source from the asynchronous AUX event bus. The selected signal enters the edge-detection circuit. False capture events can occur when: - the edge-detection circuit contains expired signal samples and the circuit is enabled without flush as described in CH3EVCFG - this register is reconfigured while CTL.MODE is different from DIS. You can avoid false capture events. When wanted channel function: - SET_ON_CAPT_DIS, see description for SET_ON_CAPT_DIS in CH3EVCFG.CCACT. - SET_ON_CAPT, see description for SET_ON_CAPT in CH3EVCFG.CCACT. - PER_PULSE_WIDTH_MEAS, see description for PER_PULSE_WIDTH_MEAS in CH3EVCFG.CCACT. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1

Table 20-157. CH3CCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL 3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.
0	EDGE	R/W	0h	Edge configuration. Channel captures counter value at selected edge on signal source selected by CAPT_SRC. See CH3EVCFG.CCACT. 0h = Capture CNTR.VALUE at falling edge of CAPT_SRC. 1h = Capture CNTR.VALUE at rising edge of CAPT_SRC.

20.8.7.22 CH3PCC Register (Offset = B8h) [Reset = 00000000h]

CH3PCC is shown in Table 20-158.

Return to the Summary Table.

Channel 3 Pipeline Capture Compare

Table 20-158. CH3PCC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Pipeline Capture Compare value. 16-bit user defined pipeline compare value or channel-updated capture value. Compare mode: An update of VALUE will be transferred to CH3CC.VALUE when the next CNTR.VALUE is zero and CTL.MODE is different from DIS. This is useful for PWM generation and prevents jitter on the edges of the generated signal. Capture mode: When CH3EVCFG.CCACT equals PER_PULSE_WIDTH_MEAS then VALUE contains the width of the low or high phase of the selected signal. This is specified by CH3CCFG.EDGE and CH3CCFG.CAPT_SRC.

20.8.7.23 CH3CC Register (Offset = BCh) [Reset = 00000000h]

CH3CC is shown in Table 20-159.

Return to the Summary Table.

Channel 3 Capture Compare

Table 20-159. CH3CC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Capture Compare value. 16-bit user defined compare value or channel-updated capture value. Compare mode: VALUE is compared against CNTR.VALUE and an event is generated as specified by CH3EVCFG.CCACT when these are equal. Capture mode: The current counter value is stored in VALUE when a capture event occurs. CH3EVCFG.CCACT determines if VALUE is a signal period or a regular capture value.

20.8.8 AUX_ANAIF Registers

Table 20-160 lists the memory-mapped registers for the AUX_ANAIF registers. All register offset addresses not listed in Table 20-160 should be considered as reserved locations and the register contents should not be modified.

Table 20-160. AUX_ANAIF Registers

Offset	Acronym	Register Name	Section
10h	ADCCTL	ADC Control	Section 20.8.8.1
14h	ADCFIFOSTAT	ADC FIFO Status	Section 20.8.8.2
18h	ADCFIFO	ADC FIFO	Section 20.8.8.3
1Ch	ADCTRIG	ADC Trigger	Section 20.8.8.4
20h	ISRCCTL	Current Source Control	Section 20.8.8.5
30h	DACCTL	DAC Control	Section 20.8.8.6
34h	LPMBIASCTL	Low Power Mode Bias Control	Section 20.8.8.7
38h	DACSMPLCTL	DAC Sample Control	Section 20.8.8.8
3Ch	DACSMPLCFG0	DAC Sample Configuration 0	Section 20.8.8.9
40h	DACSMPLCFG1	DAC Sample Configuration 1	Section 20.8.8.10
44h	DACVALUE	DAC Value	Section 20.8.8.11
48h	DACSTAT	DAC Status	Section 20.8.8.12

Complex bit access types are encoded to fit into small table cells. Table 20-161 shows the codes that are used for access types in this section.

Table 20-161. AUX_ANAIF Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.8.1 ADCCTL Register (Offset = 10h) [Reset = 00003F00h]

ADCCTL is shown in Table 20-162.

Return to the Summary Table.

ADC Control
Configuration of ADI_4_AUX:ADC0.SMPL_MODE decides if the ADC trigger starts sampling or conversion.

Table 20-162. ADCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	START_POL	R/W	0h	Select active polarity for START_SRC event. 0h = Set ADC trigger on rising edge of event source. 1h = Set ADC trigger on falling edge of event source.

Table 20-162. ADCCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-8	START_SRC	R/W	3Fh	Select ADC trigger event source from the asynchronous AUX event bus. Set START_SRC to NO_EVENT if you want to trigger the ADC manually through ADCTRIG.START. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.
7-2	RESERVED	R	0h	Reserved

Table 20-162. ADCCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	CMD	R/W	0h	ADC interface command. Non-enumerated values are not supported. The written value is returned when read. 0h = Disable ADC interface. 1h = Enable ADC interface. 3h = Flush ADC FIFO. You must set CMD to EN or DIS after flush. System CPU must wait two clock cycles before it sets CMD to EN or DIS.

20.8.8.2 ADCFIFOSTAT Register (Offset = 14h) [Reset = 00000001h]

ADCFIFOSTAT is shown in Table 20-163.

Return to the Summary Table.

ADC FIFO Status
FIFO can hold up to four ADC samples.

Table 20-163. ADCFIFOSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	OVERFLOW	R	0h	FIFO overflow flag. 0: FIFO has not overflowed. 1: FIFO has overflowed, this flag is sticky until you flush the FIFO. When the flag is set, the ADC FIFO write pointer is static. It is not possible to add more samples to the ADC FIFO. Flush FIFO to clear the flag.
3	UNDERFLOW	R	0h	FIFO underflow flag. 0: FIFO has not underflowed. 1: FIFO has underflowed, this flag is sticky until you flush the FIFO. When the flag is set, the ADC FIFO read pointer is static. Read returns the previous sample that was read. Flush FIFO to clear the flag.
2	FULL	R	0h	FIFO full flag. 0: FIFO is not full, there is less than 4 samples in the FIFO. 1: FIFO is full, there are 4 samples in the FIFO. When the flag is set, it is not possible to add more samples to the ADC FIFO. An attempt to add samples sets the OVERFLOW flag.
1	ALMOST_FULL	R	0h	FIFO almost full flag. 0: There are less than 3 samples in the FIFO, or the FIFO is full. The FULL flag is also asserted in the latter case. 1: There are 3 samples in the FIFO, there is room for one more sample.
0	EMPTY	R	1h	FIFO empty flag. 0: FIFO contains one or more samples. 1: FIFO is empty. When the flag is set, read returns the previous sample that was read and sets the UNDERFLOW flag.

20.8.8.3 ADCFIFO Register (Offset = 18h) [Reset = 00000000h]

ADCFIFO is shown in Table 20-164.

Return to the Summary Table.

ADC FIFO

Table 20-164. ADCFIFO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11-0	DATA	R/W	0h	FIFO data. Read: Get oldest ADC sample from FIFO. Write: Write dummy sample to FIFO. This is useful for code development when you do not have real ADC samples.

20.8.8.4 ADCTRIG Register (Offset = 1Ch) [Reset = 00000000h]

ADCTRIG is shown in Table 20-165.

Return to the Summary Table.

ADC Trigger

Table 20-165. ADCTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	START	W	0h	Manual ADC trigger. Write any value to START to trigger ADC. To manually trigger the ADC, you must set ADCCTL.START_SRC to NO_EVENT to avoid conflict with event-driven ADC trigger.

20.8.8.5 ISRCCTL Register (Offset = 20h) [Reset = 00000001h]

ISRCCTL is shown in Table 20-166.

Return to the Summary Table.

Current Source Control

Table 20-166. ISRCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RESET_N	R/W	1h	ISRC reset control. 0: ISRC drives 0 uA. 1: ISRC drives current ADI_4_AUX:ISRC.TRIM to COMPA_IN.

20.8.8.6 DACCTL Register (Offset = 30h) [Reset = 00000000h]

DACCTL is shown in Table 20-167.

Return to the Summary Table.

DAC Control
This register controls the analog part of the DAC.

Table 20-167. DACCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	DAC_EN	R/W	0h	DAC module enable. 0: Disable DAC. 1: Enable DAC. The Sensor Controller must not use the DAC when AUX_SYSIF:OPMODEREQ.REQ equals PDA. The System CPU must not use the DAC when AUX_SYSIF:OPMODEREQ.REQ equals PDA in Standby mode. The System CPU must set AUX_SYSIF:PEROPRATE.ANAIF_DAC_OP_RATE to BUS_RATE to use the DAC in Active and Idle modes. Standby, Active, and Idle are power modes defined in TI's Power Manager.
4	DAC_BUFFER_EN	R/W	0h	DAC buffer enable. DAC buffer reduces the time required to produce the programmed voltage at the expense of increased current consumption. 0: Disable DAC buffer. 1: Enable DAC buffer. Enable buffer when DAC_VOUT_SEL equals COMPA_IN. Do not enable the buffer when AUX_SYSIF:OPMODEREQ.REQ equals PDA or PDLP.
3	DAC_PRECHARGE_EN	R/W	0h	DAC precharge enable. Only enable precharge when ADI_4_AUX:MUX2.DAC_VREF_SEL equals DCOUPL and VDDS is higher than 2.65 V. DAC output voltage range: 0: 0 V to 1.28 V. 1: 1.28 V to 2.56 V. Otherwise, see ADI_4_AUX:MUX2.DAC_VREF_SEL for DAC output voltage range. Enable precharge 1 us before you enable the DAC and the buffer.
2-0	DAC_VOUT_SEL	R/W	0h	DAC output connection. An analog node must only have one driver. Other drivers for the following analog nodes are configured in [ANATOP_MMAP::ADI_4_AUX:*]. 0h = Connect to nothing It is recommended to use NC as intermediate step when you change DAC_VOUT_SEL. 1h = Connect to COMPB_REF analog node. Required setting to use Comparator B. 2h = Connect to COMPA_REF analog node. It is not possible to drive external loads connected to COMPA_REF I/O mux with this setting. 4h = Connect to COMPA_IN analog node. Required setting to drive external load selected in ADI_4_AUX:MUX1.COMPA_IN.

20.8.8.7 LPMBIASCTL Register (Offset = 34h) [Reset = 00000000h]

LPMBIASCTL is shown in Table 20-168.

Return to the Summary Table.

Low Power Mode Bias Control
The low power mode bias module provides bias current to DAC and Comparator A when AUX_SYSIF:OPMODEREQ.REQ differers from A.

Table 20-168. LPMBIASCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Module enable. 0: Disable low power mode bias module. 1: Enable low power mode bias module. Set EN to 1 15 us before you enable the DAC or Comparator A.

20.8.8.8 DACSMPLCTL Register (Offset = 38h) [Reset = 00000000h]

DACSMPLCTL is shown in Table 20-169.

Return to the Summary Table.

DAC Sample Control
The DAC sample clock maintains the DAC voltage stored in the sample-and-hold capacitor. The DAC sample clock waveform consists of a setup phase followed by a hold phase. In the setup phase the sample-and-hold capacitor charges to the programmed voltage. The hold phase maintains the voltage with minimal power. DACSMPLCFG0 and DACSMPLCFG1 configure the DAC sample clock waveform.

Table 20-169. DACSMPLCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	DAC sample clock enable. 0: Disable sample clock. The sample clock stops low and DACSTAT becomes 0 when the current sample clock period completes. 1: Enable DAC sample clock. DACSTAT must be 0 before you enable sample clock.

20.8.8.9 DACSMPLCFG0 Register (Offset = 3Ch) [Reset = 00000000h]

DACSMPLCFG0 is shown in Table 20-170.

Return to the Summary Table.

DAC Sample Configuration 0

Table 20-170. DACSMPLCFG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	CLKDIV	R/W	0h	Clock division. AUX_SYSIF:PEROPRATE.ANAIF_DAC_OP_RATE divided by (CLKDIV + 1) determines the sample clock base frequency. 0: Divide by 1. 1: Divide by 2. ... 63: Divide by 64.

20.8.8.10 DACSMPLCFG1 Register (Offset = 40h) [Reset = 00000000h]

DACSMPLCFG1 is shown in Table 20-171.

Return to the Summary Table.

DAC Sample Configuration 1
The sample clock period equals (high time + low time) * base period. DACSMPLCFG0.CLKDIV determines the base period.
Timing requirements (DAC Buffer On / DAC Buffer Off):
- (high time + low time) * base period > (4 us / 1 us)
- (high time * base period) > (2 us / 0.5 us)
- (low time * base period) > (2 us / 0.5 us)
- (low time * base period + HOLD_INTERVAL * sample clock period) < 32 us
If AUX_SYSIF:OPMODEREQ.REQ equals PDLP, you must set:
- H_PER = L_PER = HOLD_INTERVAL = 0.

Table 20-171. DACSMPLCFG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14	H_PER	R/W	0h	High time. The sample clock period is high for this many base periods. 0: 2 periods 1: 4 periods
13-12	L_PER	R/W	0h	Low time. The sample clock period is low for this many base periods. 0: 1 period 1: 2 periods 2: 3 periods 3: 4 periods
11-8	SETUP_CNT	R/W	0h	Setup count. Number of active sample clock periods during the setup phase. 0: 1 sample clock period 1: 2 sample clock periods ... 15 : 16 sample clock periods
7-0	HOLD_INTERVAL	R/W	0h	Hold interval. Number of inactive sample clock periods between each active sample clock period during hold phase. The sample clock is low when inactive. The range is 0 to 255.

20.8.8.11 DACVALUE Register (Offset = 44h) [Reset = 00000000h]

DACVALUE is shown in Table 20-172.

Return to the Summary Table.

DAC Value

Table 20-172. DACVALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	VALUE	R/W	0h	DAC value. Digital data word for the DAC. Only change VALUE when DACCTL.DAC_EN is 0. Then wait 1 us before you enable the DAC.

20.8.8.12 DACSTAT Register (Offset = 48h) [Reset = 00000000h]

DACSTAT is shown in Table 20-173.

Return to the Summary Table.

DAC Status

Table 20-173. DACSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	SETUP_ACTIVE	R	0h	DAC setup phase status. 0: Sample clock is disabled or setup phase is complete. 1: Setup phase in progress.
0	HOLD_ACTIVE	R	0h	DAC hold phase status. 0: Sample clock is disabled or DAC is not in hold phase. 1: Hold phase in progress.

20.8.9 AUX_SYSIF Registers

Table 20-174 lists the memory-mapped registers for the AUX_SYSIF registers. All register offset addresses not listed in Table 20-174 should be considered as reserved locations and the register contents should not be modified.

Table 20-174. AUX_SYSIF Registers

Offset	Acronym	Register Name	Section
0h	OPMODEREQ	Operational Mode Request	Section 20.8.9.1
4h	OPMODEACK	Operational Mode Acknowledgement	Section 20.8.9.2
8h	PROGWU0CFG	Programmable Wakeup 0 Configuration	Section 20.8.9.3
Ch	PROGWU1CFG	Programmable Wakeup 1 Configuration	Section 20.8.9.4
10h	PROGWU2CFG	Programmable Wakeup 2 Configuration	Section 20.8.9.5
14h	PROGWU3CFG	Programmable Wakeup 3 Configuration	Section 20.8.9.6
18h	SWWUTRIG	Software Wakeup Triggers	Section 20.8.9.7
1Ch	WUFLAGS	Wakeup Flags	Section 20.8.9.8
20h	WUFLAGSCLR	Wakeup Flags Clear	Section 20.8.9.9
24h	WUGATE	Wakeup Gate	Section 20.8.9.10
28h	VECCFG0	Vector Configuration 0	Section 20.8.9.11
2Ch	VECCFG1	Vector Configuration 1	Section 20.8.9.12
30h	VECCFG2	Vector Configuration 2	Section 20.8.9.13
34h	VECCFG3	Vector Configuration 3	Section 20.8.9.14
38h	VECCFG4	Vector Configuration 4	Section 20.8.9.15
3Ch	VECCFG5	Vector Configuration 5	Section 20.8.9.16
40h	VECCFG6	Vector Configuration 6	Section 20.8.9.17
44h	VECCFG7	Vector Configuration 7	Section 20.8.9.18
48h	EVSYNCRATE	Event Synchronization Rate	Section 20.8.9.19
4Ch	PEROPRATE	Peripheral Operational Rate	Section 20.8.9.20
50h	ADCCLKCTL	ADC Clock Control	Section 20.8.9.21
54h	TDCCLKCTL	TDC Counter Clock Control	Section 20.8.9.22
58h	TDCREFCLKCTL	TDC Reference Clock Control	Section 20.8.9.23
5Ch	TIMER2CLKCTL	AUX_TIMER2 Clock Control	Section 20.8.9.24
60h	TIMER2CLKSTAT	AUX_TIMER2 Clock Status	Section 20.8.9.25
64h	TIMER2CLKSWITCH	AUX_TIMER2 Clock Switch	Section 20.8.9.26
68h	TIMER2DBGCTL	AUX_TIMER2 Debug Control	Section 20.8.9.27
70h	CLKSHIFTDET	Clock Shift Detection	Section 20.8.9.28
74h	RECHARGETRIG	VDDR Recharge Trigger	Section 20.8.9.29
78h	RECHARGEDET	VDDR Recharge Detection	Section 20.8.9.30
7Ch	RTCSUBSECINC0	Real Time Counter Sub Second Increment 0	Section 20.8.9.31
80h	RTCSUBSECINC1	Real Time Counter Sub Second Increment 1	Section 20.8.9.32
84h	RTCSUBSECINCCTL	Real Time Counter Sub Second Increment Control	Section 20.8.9.33
88h	RTCSEC	Real Time Counter Second	Section 20.8.9.34
8Ch	RTCSUBSEC	Real Time Counter Sub-Second	Section 20.8.9.35
90h	RTCEVCLR	AON_RTC Event Clear	Section 20.8.9.36
94h	BATMONBAT	AON_BATMON Battery Voltage Value	Section 20.8.9.37
9Ch	BATMONTEMP	AON_BATMON Temperature Value	Section 20.8.9.38
A0h	TIMERHALT	Timer Halt	Section 20.8.9.39
B0h	TIMER2BRIDGE	AUX_TIMER2 Bridge	Section 20.8.9.40

Table 20-174. AUX_SYSIF Registers (continued)

Offset	Acronym	Register Name	Section
B4h	SWPWRPROF	Software Power Profiler	Section 20.8.9.41

Complex bit access types are encoded to fit into small table cells. Table 20-175 shows the codes that are used for access types in this section.

Table 20-175. AUX_SYSIF Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
RH	R H	Read Set or cleared by hardware
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.9.1 OPMODEREQ Register (Offset = 0h) [Reset = 00000000h]

OPMODEREQ is shown in Table 20-176.

Return to the Summary Table.

Operational Mode Request
AUX can operate in three operational modes. Each mode is associated with:
- a SCE clock source or rate, given by AON_PMCTL:AUXSCECLK. This rate is termed SCE_RATE.
- a system power supply state request. AUX can request powerdown (uLDO) or active (GLDO or DCDC) system power supply state.
- a specific system response to an active AUX wakeup flag. The response is dependent on what operational mode is requested.
uLDO power supply state offers limited current supply. AUX_SCE cannot use certain peripherals and functions such as AUX_DDI0_OSC, AUX_TDC and AUX_ANAIF ADC interface in this power supply state.
Follow these rules:
- It is not allowed to change a request until it has been acknowledged through OPMODEACK.
- A change in mode request must happen stepwise along this sequence, the direction is irrelevant:
PDA - A - LP - PDLP.
Failure to follow these rules might result in unexpected behavior and must be avoided.

Table 20-176. OPMODEREQ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	REQ	R/W	0h	AUX operational mode request. 0h = Active operational mode, characterized by: - Active system power supply state (GLDO or DCDC) request. - AON_PMCTL:AUXSCECLK.SRC sets the SCE clock frequency (SCE_RATE). - An active wakeup flag does not change operational mode. 1h = Lowpower operational mode, characterized by: - Powerdown system power supply state (uLDO) request. - SCE clock frequency (SCE_RATE) equals SCLK_MF. - An active wakeup flag does not change operational mode. 2h = Powerdown operational mode with wakeup to active mode, characterized by: - Powerdown system power supply state (uLDO) request. - AON_PMCTL:AUXSCECLK.PD_SRC sets the SCE clock frequency (SCE_RATE). - An active wakeup flag overrides the operational mode externally to active (A) as long as the flag is set. 3h = Powerdown operational mode with wakeup to lowpower mode, characterized by: - Powerdown system power supply state (uLDO) request. - AON_PMCTL:AUXSCECLK.PD_SRC sets the SCE clock frequency (SCE_RATE). - An active wakeup flag overrides the operational mode externally to lowpower (LP) as long as the flag is set.

20.8.9.2 OPMODEACK Register (Offset = 4h) [Reset = 00000000h]

OPMODEACK is shown in Table 20-177.

Return to the Summary Table.

Operational Mode Acknowledgement
AUX_SCE program must assume that the current operational mode is the one acknowledged.

Table 20-177. OPMODEACK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	ACK	R	0h	AUX operational mode acknowledgement. 0h = Active operational mode is acknowledged. 1h = Lowpower operational mode is acknowledged. 2h = Powerdown operational mode with wakeup to active mode is acknowledged. 3h = Powerdown operational mode with wakeup to lowpower mode is acknowledged.

20.8.9.3 PROGWU0CFG Register (Offset = 8h) [Reset = 00000000h]

PROGWU0CFG is shown in Table 20-178.

Return to the Summary Table.

Programmable Wakeup 0 Configuration
Configure this register to enable a customized AUX wakeup flag. The wakeup flag will be captured by AON_PMCTL which responds according to the current operational mode. You can select WUFLAGS.PROG_WU0 to trigger execution of a programmable AUX_SCE vector by configuration of VECCFGn. You need to follow the procedure described in WUFLAGSCLR to clear this flag. You need to follow the procedure described in WUGATE to configure it.

Table 20-178. PROGWU0CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	POL	R/W	0h	Polarity of WU_SRC. The procedure used to clear the wakeup flag decides level or edge sensitivity, see WUFLAGSCLR.PROG_WU0. 0h = The wakeup flag is set when WU_SRC is high or goes high. 1h = The wakeup flag is set when WU_SRC is low or goes low.
6	EN	R/W	0h	Programmable wakeup flag enable. 0: Disable wakeup flag. 1: Enable wakeup flag.

Table 20-178. PROGWU0CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU_SRC	R/W	0h	Wakeup source from the asynchronous AUX event bus. Only change WU_SRC when EN is 0 or WUFLAGSCLR.PROG_WU0 is 1. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL

Table 20-178. PROGWU0CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.

20.8.9.4 PROGWU1CFG Register (Offset = Ch) [Reset = 00000000h]

PROGWU1CFG is shown in Table 20-179.

Return to the Summary Table.

Programmable Wakeup 1 Configuration
Configure this register to enable a customized AUX wakeup flag. The wakeup flag will be captured by AON_PMCTL which responds according to the current operational mode. You can select WUFLAGS.PROG_WU1 to trigger execution of a programmable AUX_SCE vector by configuration of VECCFGn. You need to follow the procedure described in WUFLAGSCLR to clear this flag. You need to follow the procedure described in WUGATE to configure it.

Table 20-179. PROGWU1CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	POL	R/W	0h	Polarity of WU_SRC. The procedure used to clear the wakeup flag decides level or edge sensitivity, see WUFLAGSCLR.PROG_WU1. 0h = The wakeup flag is set when WU_SRC is high or goes high. 1h = The wakeup flag is set when WU_SRC is low or goes low.
6	EN	R/W	0h	Programmable wakeup flag enable. 0: Disable wakeup flag. 1: Enable wakeup flag.

Table 20-179. PROGWU1CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU_SRC	R/W	0h	Wakeup source from the asynchronous AUX event bus. Only change WU_SRC when EN is 0 or WUFLAGSCLR.PROG_WU1 is 1. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL

Table 20-179. PROGWU1CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.

20.8.9.5 PROGWU2CFG Register (Offset = 10h) [Reset = 00000000h]

PROGWU2CFG is shown in Table 20-180.

Return to the Summary Table.

Programmable Wakeup 2 Configuration
Configure this register to enable a customized AUX wakeup flag. The wakeup flag will be captured by AON_PMCTL which responds according to the current operational mode. You can select WUFLAGS.PROG_WU2 to trigger execution of a programmable AUX_SCE vector by configuration of VECCFGn. You need to follow the procedure described in WUFLAGSCLR to clear this flag. You need to follow the procedure described in WUGATE to configure it.

Table 20-180. PROGWU2CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	POL	R/W	0h	Polarity of WU_SRC. The procedure used to clear the wakeup flag decides level or edge sensitivity, see WUFLAGSCLR.PROG_WU2. 0h = The wakeup flag is set when WU_SRC is high or goes high. 1h = The wakeup flag is set when WU_SRC is low or goes low.
6	EN	R/W	0h	Programmable wakeup flag enable. 0: Disable wakeup flag. 1: Enable wakeup flag.

Table 20-180. PROGWU2CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU_SRC	R/W	0h	Wakeup source from the asynchronous AUX event bus. Only change WU_SRC when EN is 0 or WUFLAGSCLR.PROG_WU2 is 1. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL

Table 20-180. PROGWU2CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.

20.8.9.6 PROGWU3CFG Register (Offset = 14h) [Reset = 00000000h]

PROGWU3CFG is shown in Table 20-181.

Return to the Summary Table.

Programmable Wakeup 3 Configuration
Configure this register to enable a customized AUX wakeup flag. The wakeup flag will be captured by AON_PMCTL which responds according to the current operational mode. You can select WUFLAGS.PROG_WU3 to trigger execution of a programmable AUX_SCE vector by configuration of VECCFGn. You need to follow the procedure described in WUFLAGSCLR to clear this flag. You need to follow the procedure described in WUGATE to configure it.

Table 20-181. PROGWU3CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	POL	R/W	0h	Polarity of WU_SRC. The procedure used to clear the wakeup flag decides level or edge sensitivity, see WUFLAGSCLR.PROG_WU3. 0h = The wakeup flag is set when WU_SRC is high or goes high. 1h = The wakeup flag is set when WU_SRC is low or goes low.
6	EN	R/W	0h	Programmable wakeup flag enable. 0: Disable wakeup flag. 1: Enable wakeup flag.

Table 20-181. PROGWU3CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-0	WU_SRC	R/W	0h	Wakeup source from the asynchronous AUX event bus. Only change WU_SRC when EN is 0 or WUFLAGSCLR.PROG_WU3 is 1. If you write a non-enumerated value the behavior is identical to NO_EVENT. The written value is returned when read. 0h = AUX_EVCTL:EVSTAT0.AUXIO0 1h = AUX_EVCTL:EVSTAT0.AUXIO1 2h = AUX_EVCTL:EVSTAT0.AUXIO2 3h = AUX_EVCTL:EVSTAT0.AUXIO3 4h = AUX_EVCTL:EVSTAT0.AUXIO4 5h = AUX_EVCTL:EVSTAT0.AUXIO5 6h = AUX_EVCTL:EVSTAT0.AUXIO6 7h = AUX_EVCTL:EVSTAT0.AUXIO7 8h = AUX_EVCTL:EVSTAT0.AUXIO8 9h = AUX_EVCTL:EVSTAT0.AUXIO9 Ah = AUX_EVCTL:EVSTAT0.AUXIO10 Bh = AUX_EVCTL:EVSTAT0.AUXIO11 Ch = AUX_EVCTL:EVSTAT0.AUXIO12 Dh = AUX_EVCTL:EVSTAT0.AUXIO13 Eh = AUX_EVCTL:EVSTAT0.AUXIO14 Fh = AUX_EVCTL:EVSTAT0.AUXIO15 10h = AUX_EVCTL:EVSTAT1.AUXIO16 11h = AUX_EVCTL:EVSTAT1.AUXIO17 12h = AUX_EVCTL:EVSTAT1.AUXIO18 13h = AUX_EVCTL:EVSTAT1.AUXIO19 14h = AUX_EVCTL:EVSTAT1.AUXIO20 15h = AUX_EVCTL:EVSTAT1.AUXIO21 16h = AUX_EVCTL:EVSTAT1.AUXIO22 17h = AUX_EVCTL:EVSTAT1.AUXIO23 18h = AUX_EVCTL:EVSTAT1.AUXIO24 19h = AUX_EVCTL:EVSTAT1.AUXIO25 1Ah = AUX_EVCTL:EVSTAT1.AUXIO26 1Bh = AUX_EVCTL:EVSTAT1.AUXIO27 1Ch = AUX_EVCTL:EVSTAT1.AUXIO28 1Dh = AUX_EVCTL:EVSTAT1.AUXIO29 1Eh = AUX_EVCTL:EVSTAT1.AUXIO30 1Fh = AUX_EVCTL:EVSTAT1.AUXIO31 20h = AUX_EVCTL:EVSTAT2.MANUAL_EV 21h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2 22h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY 23h = AUX_EVCTL:EVSTAT2.AON_RTC_4KHZ 24h = AUX_EVCTL:EVSTAT2.AON_BATMON_BAT_UPD 25h = AUX_EVCTL:EVSTAT2.AON_BATMON_TEMP_UPD 26h = AUX_EVCTL:EVSTAT2.SCLK_LF 27h = AUX_EVCTL:EVSTAT2.PWR_DWN 28h = AUX_EVCTL:EVSTAT2.MCU_ACTIVE 29h = AUX_EVCTL:EVSTAT2.VDDR_RECHARGE 2Ah = AUX_EVCTL:EVSTAT2.ACLK_REF 2Bh = AUX_EVCTL:EVSTAT2.MCU_EV 2Ch = AUX_EVCTL:EVSTAT2.MCU_OBSMUX0 2Dh = AUX_EVCTL:EVSTAT2.MCU_OBSMUX1 2Eh = AUX_EVCTL:EVSTAT2.AUX_COMPA 2Fh = AUX_EVCTL:EVSTAT2.AUX_COMPB 30h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 31h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 32h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 33h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 34h = AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 35h = AUX_EVCTL:EVSTAT3.AUX_TIMER1_EV 36h = AUX_EVCTL:EVSTAT3.AUX_TIMER0_EV 37h = AUX_EVCTL:EVSTAT3.AUX_TDC_DONE 38h = AUX_EVCTL:EVSTAT3.AUX_ISRC_RESET_N 39h = AUX_EVCTL:EVSTAT3.AUX_ADC_DONE 3Ah = AUX_EVCTL:EVSTAT3.AUX_ADC_IRQ 3Bh = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_ALMOST_FULL

Table 20-181. PROGWU3CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
				3Ch = AUX_EVCTL:EVSTAT3.AUX_ADC_FIFO_NOT_EMPTY 3Dh = AUX_EVCTL:EVSTAT3.AUX_SMPH_AUTOTAKE_DONE 3Fh = No event.

20.8.9.7 SWWUTRIG Register (Offset = 18h) [Reset = 00000000h]

SWWUTRIG is shown in Table 20-182.

Return to the Summary Table.

Software Wakeup Triggers
System CPU uses these wakeup flags to perform handshaking with AUX_SCE. The wakeup flags can change the operational mode of AUX and guarantees a non-zero SCE clock rate. AUX_SCE wakeup vectors are configured in VECCFGn.

Table 20-182. SWWUTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	SW_WU3	W	0h	Software wakeup 3 trigger. 0: No effect. 1: Set WUFLAGS.SW_WU3 and trigger AUX wakeup.
2	SW_WU2	W	0h	Software wakeup 2 trigger. 0: No effect. 1: Set WUFLAGS.SW_WU2 and trigger AUX wakeup.
1	SW_WU1	W	0h	Software wakeup 1 trigger. 0: No effect. 1: Set WUFLAGS.SW_WU1 and trigger AUX wakeup.
0	SW_WU0	W	0h	Software wakeup 0 trigger. 0: No effect. 1: Set WUFLAGS.SW_WU0 and trigger AUX wakeup.

20.8.9.8 WUFLAGS Register (Offset = 1Ch) [Reset = 00000000h]

WUFLAGS is shown in Table 20-183.

Return to the Summary Table.

Wakeup Flags
This register holds the eight AUX wakeup flags. Each flag can cause AUX operational mode to change as given in OPMODEREQ. To clear flag n you must set bit n in WUFLAGSCLR until flag n is read as 0. You must clear bit n in WUFLAGSCLR before flag n can be set again.

Table 20-183. WUFLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	SW_WU3	R	0h	Software wakeup 3 flag. 0: Software wakeup 3 not triggered. 1: Software wakeup 3 triggered.
6	SW_WU2	R	0h	Software wakeup 2 flag. 0: Software wakeup 2 not triggered. 1: Software wakeup 2 triggered.
5	SW_WU1	R	0h	Software wakeup 1 flag. 0: Software wakeup 1 not triggered. 1: Software wakeup 1 triggered.
4	SW_WU0	R	0h	Software wakeup 0 flag. 0: Software wakeup 0 not triggered. 1: Software wakeup 0 triggered.
3	PROG_WU3	R	0h	Programmable wakeup 3. 0: Programmable wakeup 3 not triggered. 1: Programmable wakeup 3 triggered.
2	PROG_WU2	R	0h	Programmable wakeup 2. 0: Programmable wakeup 2 not triggered. 1: Programmable wakeup 2 triggered.
1	PROG_WU1	R	0h	Programmable wakeup 1. 0: Programmable wakeup 1 not triggered. 1: Programmable wakeup 1 triggered.
0	PROG_WU0	R	0h	Programmable wakeup 0. 0: Programmable wakeup 0 not triggered. 1: Programmable wakeup 0 triggered.

20.8.9.9 WUFLAGSCLR Register (Offset = 20h) [Reset = 0000000Fh]

WUFLAGSCLR is shown in Table 20-184.

Return to the Summary Table.

Wakeup Flags Clear
This register clears AUX wakeup flags WUFLAGS.
To clear programmable wakeup flags you must disable the AUX wakeup output first. After the programmable wakeup flags are cleared you must re-enable the AUX wakeup output. Write WUGATE to disable or enable the AUX wakeup output. This procedure is not required when you want to clear a software-triggered wakeup.

Table 20-184. WUFLAGSCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	SW_WU3	R/W	0h	Clear software wakeup flag 3. 0: No effect. 1: Clear WUFLAGS.SW_WU3. Keep high until WUFLAGS.SW_WU3 is 0.
6	SW_WU2	R/W	0h	Clear software wakeup flag 2. 0: No effect. 1: Clear WUFLAGS.SW_WU2. Keep high until WUFLAGS.SW_WU2 is 0.
5	SW_WU1	R/W	0h	Clear software wakeup flag 1. 0: No effect. 1: Clear WUFLAGS.SW_WU1. Keep high until WUFLAGS.SW_WU1 is 0.
4	SW_WU0	R/W	0h	Clear software wakeup flag 0. 0: No effect. 1: Clear WUFLAGS.SW_WU0. Keep high until WUFLAGS.SW_WU0 is 0.
3	PROG_WU3	R/W	1h	Programmable wakeup flag 3. 0: No effect. 1: Clear WUFLAGS.PROG_WU3. Keep high until WUFLAGS.PROG_WU3 is 0. The wakeup flag becomes edge sensitive if you write PROG_WU3 to 0 when PROGWU3CFG.EN is 1. The wakeup flag becomes level sensitive if you write PROG_WU3 to 0 when PROGWU3CFG.EN is 0, then set PROGWU3CFG.EN.
2	PROG_WU2	R/W	1h	Programmable wakeup flag 2. 0: No effect. 1: Clear WUFLAGS.PROG_WU2. Keep high until WUFLAGS.PROG_WU2 is 0. The wakeup flag becomes edge sensitive if you write PROG_WU2 to 0 when PROGWU2CFG.EN is 1. The wakeup flag becomes level sensitive if you write PROG_WU2 to 0 when PROGWU2CFG.EN is 0, then set PROGWU2CFG.EN.
1	PROG_WU1	R/W	1h	Programmable wakeup flag 1. 0: No effect. 1: Clear WUFLAGS.PROG_WU1. Keep high until WUFLAGS.PROG_WU1 is 0. The wakeup flag becomes edge sensitive if you write PROG_WU1 to 0 when PROGWU1CFG.EN is 1. The wakeup flag becomes level sensitive if you write PROG_WU1 to 0 when PROGWU1CFG.EN is 0, then set PROGWU1CFG.EN.

Table 20-184. WUFLAGSCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PROG_WU0	R/W	1h	Programmable wakeup flag 0. 0: No effect. 1: Clear WUFLAGS.PROG_WU0. Keep high until WUFLAGS.PROG_WU0 is 0. The wakeup flag becomes edge sensitive if you write PROG_WU0 to 0 when PROGWU0CFG.EN is 1. The wakeup flag becomes level sensitive if you write PROG_WU0 to 0 when PROGWU0CFG.EN is 0, then set PROGWU0CFG.EN.

20.8.9.10 WUGATE Register (Offset = 24h) [Reset = 00000000h]

WUGATE is shown in Table 20-185.

Return to the Summary Table.

Wakeup Gate
You must disable the AUX wakeup output:
- Before you clear a programmable wakeup flag.
- Before you change the value of [PROGWUnCFG.EN] or [PROGWUnCFG.WU_SRC].
The AUX wakeup output must be re-enabled after clear operation or programmable wakeup configuration.

Table 20-185. WUGATE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Wakeup output enable. 0: Disable AUX wakeup output. 1: Enable AUX wakeup output.

20.8.9.11 VECCFG0 Register (Offset = 28h) [Reset = 00000000h]

VECCFG0 is shown in Table 20-186.

Return to the Summary Table.

Vector Configuration 0
AUX_SCE wakeup vector 0 configuration

Table 20-186. VECCFG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 0. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.12 VECCFG1 Register (Offset = 2Ch) [Reset = 00000000h]

VECCFG1 is shown in Table 20-187.

Return to the Summary Table.

Vector Configuration 1
AUX_SCE wakeup vector 1 configuration

Table 20-187. VECCFG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 1. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.13 VECCFG2 Register (Offset = 30h) [Reset = 00000000h]

VECCFG2 is shown in Table 20-188.

Return to the Summary Table.

Vector Configuration 2
AUX_SCE wakeup vector 2 configuration

Table 20-188. VECCFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 2. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.14 VECCFG3 Register (Offset = 34h) [Reset = 00000000h]

VECCFG3 is shown in Table 20-189.

Return to the Summary Table.

Vector Configuration 3
AUX_SCE wakeup vector 3 configuration

Table 20-189. VECCFG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 3. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.15 VECCFG4 Register (Offset = 38h) [Reset = 00000000h]

VECCFG4 is shown in Table 20-190.

Return to the Summary Table.

Vector Configuration 4
AUX_SCE wakeup vector 4 configuration

Table 20-190. VECCFG4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 4. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.16 VECCFG5 Register (Offset = 3Ch) [Reset = 00000000h]

VECCFG5 is shown in Table 20-191.

Return to the Summary Table.

Vector Configuration 5
AUX_SCE wakeup vector 5 configuration

Table 20-191. VECCFG5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 5. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.17 VECCFG6 Register (Offset = 40h) [Reset = 00000000h]

VECCFG6 is shown in Table 20-192.

Return to the Summary Table.

Vector Configuration 6
AUX_SCE wakeup vector 6 configuration

Table 20-192. VECCFG6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 6. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.18 VECCFG7 Register (Offset = 44h) [Reset = 00000000h]

VECCFG7 is shown in Table 20-193.

Return to the Summary Table.

Vector Configuration 7
AUX_SCE wakeup vector 7 configuration

Table 20-193. VECCFG7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-0	VEC_EV	R/W	0h	Select trigger event for vector 7. Non-enumerated values are treated as NONE. 0h = Vector is disabled. 1h = WUFLAGS.PROG_WU0 2h = WUFLAGS.PROG_WU1 3h = WUFLAGS.PROG_WU2 4h = WUFLAGS.PROG_WU3 5h = WUFLAGS.SW_WU0 6h = WUFLAGS.SW_WU1 7h = WUFLAGS.SW_WU2 8h = WUFLAGS.SW_WU3 9h = AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY

20.8.9.19 EVSYNCRATE Register (Offset = 48h) [Reset = 00000000h]

EVSYNCRATE is shown in Table 20-194.

Return to the Summary Table.

Event Synchronization Rate
Configure synchronization rate for certain events to the synchronous AUX event bus.
You must select SCE rate when AUX_SCE uses the event. You must select AUX bus rate when system CPU uses the event.
SCE rate equals rate configured in AON_PMCTL:AUXSCECLK. AUX bus rate equals SCE rate, or SCLK_HF divided by two when MCU domain is active.

Table 20-194. EVSYNCRATE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	AUX_COMPA_SYNC_RATE	R/W	0h	Select synchronization rate for AUX_EVCTL:EVSTAT2.AUX_COMPA event. 0h = SCE rate 1h = AUX bus rate
1	AUX_COMPB_SYNC_RATE	R/W	0h	Select synchronization rate for AUX_EVCTL:EVSTAT2.AUX_COMPB event. 0h = SCE rate 1h = AUX bus rate
0	AUX_TIMER2_SYNC_RATE	R/W	0h	Select synchronization rate for: - AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV0 - AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV1 - AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV2 - AUX_EVCTL:EVSTAT3.AUX_TIMER2_EV3 - AUX_EVCTL:EVSTAT3.AUX_TIMER2_PULSE 0h = SCE rate 1h = AUX bus rate

20.8.9.20 PEROPRATE Register (Offset = 4Ch) [Reset = 00000000h]

PEROPRATE is shown in Table 20-195.

Return to the Summary Table.

Peripheral Operational Rate
Some AUX peripherals are operated at either SCE or at AUX bus rate.
You must select SCE rate when AUX_SCE uses such peripheral or an event produced by it. You must select AUX bus rate when system CPU uses such peripheral.
SCE rate equals rate configured in AON_PMCTL:AUXSCECLK. AUX bus rate equals SCE rate, or SCLK_HF divided by 2 when MCU domain is active.

Table 20-195. PEROPRATE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	ANAIF_DAC_OP_RATE	R/W	0h	Select operational rate for AUX_ANAIF DAC sample clock state machine. 0h = SCE rate 1h = AUX bus rate
2	TIMER01_OP_RATE	R/W	0h	Select operational rate for AUX_TIMER01. 0h = SCE rate 1h = AUX bus rate
1	SPIM_OP_RATE	R/W	0h	Select operational rate for AUX_SPIM. 0h = SCE rate 1h = AUX bus rate
0	MAC_OP_RATE	R/W	0h	Select operational rate for AUX_MAC. 0h = SCE rate 1h = AUX bus rate

20.8.9.21 ADCCLKCTL Register (Offset = 50h) [Reset = 00000000h]

ADCCLKCTL is shown in Table 20-196.

Return to the Summary Table.

ADC Clock Control

Table 20-196. ADCCLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	ACK	R	0h	Clock acknowledgement. 0: ADC clock is disabled. 1: ADC clock is enabled.
0	REQ	R/W	0h	ADC clock request. 0: Disable ADC clock. 1: Enable ADC clock. Only modify REQ when equal to ACK.

20.8.9.22 TDCCLKCTL Register (Offset = 54h) [Reset = 00000000h]

TDCCLKCTL is shown in Table 20-197.

Return to the Summary Table.

TDC Counter Clock Control
Controls if the AUX_TDC counter clock source is enabled.
These are the recommended steps to configure and request the counter clock:
- Ensure that REQ=0 and ACK=0.
- Configure clock source in DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL.
- Read DDI_0_OSC:CTL0 to avoid a race condition between previous step and next step.
- Set REQ=1 to request the clock.
- If DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL=RCOSC_HF (24 or 48 MHz), wait until ACK=1.
- If DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL=XOSC_HF, wait until ACK=1 and DDI_0_OSC:STAT2.XOSC_HF_FREQGOOD=1.

After these steps ACK stays high until REQ=0. It is hence not recommended to reconfigure DDI_0_OSC:CTL0.ACLK_TDC_SRC_SEL when ACK=1. In this case, there will be no indication of when the new clock source selection is ready.
These are the recommended steps to stop the counter clock:
- Ensure that REQ=1 and ACK=1.
- Set REQ=0 to stop the clock.
- Wait until ACK=0.

Table 20-197. TDCCLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	ACK	R	0h	TDC counter clock acknowledgement. 0: TDC counter clock is disabled. 1: TDC counter clock is enabled.
0	REQ	R/W	0h	TDC counter clock request. 0: Disable TDC counter clock. 1: Enable TDC counter clock. Only modify REQ when equal to ACK.

20.8.9.23 TDCREFCLKCTL Register (Offset = 58h) [Reset = 00000000h]

TDCREFCLKCTL is shown in Table 20-198.

Return to the Summary Table.

TDC Reference Clock Control
Controls if the AUX_TDC reference clock source is enabled.
These are the recommended steps to configure and request the reference clock:
- Ensure that REQ=0 and ACK=0.
- Configure clock source in DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL.
- Read DDI_0_OSC:CTL0 to avoid a race condition between previous step and next step.
- Set REQ=1 to request the clock.
- Wait until ACK=1.
After these steps ACK stays high until REQ=0. It is hence not recommended to reconfigure DDI_0_OSC:CTL0.ACLK_REF_SRC_SEL when ACK=1. In this case, there will be no indication of when the new clock source selection is ready.
These are the recommended steps to stop the reference clock:
- Ensure that REQ=1 and ACK=1.
- Set REQ=0 to stop the clock.
- Wait until ACK=0.
It is not recommended to enable the TDC reference clock if DDI_0_OSC:CTL0.SCLK_LF_SRC_SEL=RCOSCHFDLF (0x0).

Table 20-198. TDCREFCLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	ACK	R	0h	TDC reference clock acknowledgement. 0: TDC reference clock is disabled. 1: TDC reference clock is enabled.
0	REQ	R/W	0h	TDC reference clock request. 0: Disable TDC reference clock. 1: Enable TDC reference clock. Only modify REQ when equal to ACK.

20.8.9.24 TIMER2CLKCTL Register (Offset = 5Ch) [Reset = 00000000h]

TIMER2CLKCTL is shown in Table 20-199.

Return to the Summary Table.

AUX_TIMER2 Clock Control
Access to AUX_TIMER2 is only possible when TIMER2CLKSTAT.STAT is different from NONE.

Table 20-199. TIMER2CLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	SRC	R/W	0h	Select clock source for AUX_TIMER2. Update is only accepted if SRC equals TIMER2CLKSTAT.STAT or TIMER2CLKSWITCH.RDY is 1. It is recommended to select NONE only when TIMER2BRIDGE.BUSY is 0. A non-enumerated value is ignored. 0h = no clock 1h = SCLK_LF 2h = SCLK_MF 4h = SCLK_HF / 2

20.8.9.25 TIMER2CLKSTAT Register (Offset = 60h) [Reset = 00000000h]

TIMER2CLKSTAT is shown in Table 20-200.

Return to the Summary Table.

AUX_TIMER2 Clock Status

Table 20-200. TIMER2CLKSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	STAT	R	0h	AUX_TIMER2 clock source status. 0h = No clock 1h = SCLK_LF 2h = SCLK_MF 4h = SCLK_HF / 2

20.8.9.26 TIMER2CLKSWITCH Register (Offset = 64h) [Reset = 00000001h]

TIMER2CLKSWITCH is shown in Table 20-201.

Return to the Summary Table.

AUX_TIMER2 Clock Switch

Table 20-201. TIMER2CLKSWITCH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RDY	R	1h	Status of clock switcher. 0: TIMER2CLKCTL.SRC is different from TIMER2CLKSTAT.STAT. 1: TIMER2CLKCTL.SRC equals TIMER2CLKSTAT.STAT. RDY connects to AUX_EVCTL:EVSTAT3.AUX_TIMER2_CLKSWITCH_RDY.

20.8.9.27 TIMER2DBGCTL Register (Offset = 68h) [Reset = 00000000h]

TIMER2DBGCTL is shown in Table 20-202.

Return to the Summary Table.

AUX_TIMER2 Debug Control

Table 20-202. TIMER2DBGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	DBG_FREEZE_EN	R/W	0h	Debug freeze enable. 0: AUX_TIMER2 does not halt when the system CPU halts in debug mode. 1: Halt AUX_TIMER2 when the system CPU halts in debug mode.

20.8.9.28 CLKSHIFTDET Register (Offset = 70h) [Reset = 00000001h]

CLKSHIFTDET is shown in Table 20-203.

Return to the Summary Table.

Clock Shift Detection
A transition in the MCU domain state causes a non-accumulative change to the SCE clock period when the AUX clock rate is derived from SCLK_MF or SCLK_LF:
- A single SCE clock cycle is 6 thru 8 SCLK_HF cycles longer when MCU domain enters active state.
- A single SCE clock cycle is 6 thru 8 SCLK_HF cycles shorter when MCU domain exits active state.
AUX_SCE detects if such events occurred to the SCE clock during the time period between a clear of STAT and a read of STAT.

Table 20-203. CLKSHIFTDET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W	1h	Clock shift detection. Write: 0: Restart clock shift detection. 1: Do not use. Read: 0: MCU domain did not enter or exit active state since you wrote 0 to STAT. 1: MCU domain entered or exited active state since you wrote 0 to STAT.

20.8.9.29 RECHARGETRIG Register (Offset = 74h) [Reset = 00000000h]

RECHARGETRIG is shown in Table 20-204.

Return to the Summary Table.

VDDR Recharge Trigger

Table 20-204. RECHARGETRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	TRIG	R/W	0h	Recharge trigger. 0: No effect. 1: Request VDDR recharge. Request VDDR recharge only when AUX_EVCTL:EVSTAT2.PWR_DWN is 1. Follow this sequence when OPMODEREQ.REQ is LP: - Set TRIG. - Wait until AUX_EVCTL:EVSTAT2.VDDR_RECHARGE is 1. - Clear TRIG. - Wait until AUX_EVCTL:EVSTAT2.VDDR_RECHARGE is 0. Follow this sequence when OPMODEREQ.REQ is PDA or PDLP: - Set TRIG. - Clear TRIG.

20.8.9.30 RECHARGEDET Register (Offset = 78h) [Reset = 00000000h]

RECHARGEDET is shown in Table 20-205.

Return to the Summary Table.

VDDR Recharge Detection
Some applications can be sensitive to power noise caused by recharge of VDDR. You can detect if VDDR recharge occurs.

Table 20-205. RECHARGEDET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	STAT	R	0h	VDDR recharge detector status. 0: No recharge of VDDR has occurred since EN was set. 1: Recharge of VDDR has occurred since EN was set.
0	EN	R/W	0h	VDDR recharge detector enable. 0: Disable recharge detection. STAT becomes zero. 1: Enable recharge detection.

20.8.9.31 RTCSUBSECINC0 Register (Offset = 7Ch) [Reset = 00000000h]

RTCSUBSECINC0 is shown in Table 20-206.

Return to the Summary Table.

Real Time Counter Sub Second Increment 0
INC15_0 will replace bits 15:0 in AON_RTC:SUBSECINC when RTCSUBSECINCCTL.UPD_REQ is set.
AUX_SCE is not allowed to access this register when system state is secure. Any access will suspend the AUX_SCE.

Table 20-206. RTCSUBSECINC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	INC15_0	R/W	0h	New value for bits 15:0 in AON_RTC:SUBSECINC.

20.8.9.32 RTCSUBSECINC1 Register (Offset = 80h) [Reset = 00000000h]

RTCSUBSECINC1 is shown in Table 20-207.

Return to the Summary Table.

Real Time Counter Sub Second Increment 1
INC23_16 will replace bits 23:16 in AON_RTC:SUBSECINC when RTCSUBSECINCCTL.UPD_REQ is set. AUX_SCE is not allowed to access this register when system state is secure. Any access will suspend the AUX_SCE.

Table 20-207. RTCSUBSECINC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	INC23_16	R/W	0h	New value for bits 23:16 in AON_RTC:SUBSECINC.

20.8.9.33 RTCSUBSECINCCTL Register (Offset = 84h) [Reset = 00000000h]

RTCSUBSECINCCTL is shown in Table 20-208.

Return to the Summary Table.

Real Time Counter Sub Second Increment Control
AUX_SCE is not allowed to access this register when system state is secure. Any access will suspend the AUX_SCE.

Table 20-208. RTCSUBSECINCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	UPD_ACK	R	0h	Update acknowledgement. 0: AON_RTC has not acknowledged UPD_REQ. 1: AON_RTC has acknowledged UPD_REQ.
0	UPD_REQ	R/W	0h	Request AON_RTC to update AON_RTC:SUBSECINC. 0: Clear request to update. 1: Set request to update. Only change UPD_REQ when it equals UPD_ACK. Clear UPD_REQ after UPD_ACK is 1.

20.8.9.34 RTCSEC Register (Offset = 88h) [Reset = 00000000h]

RTCSEC is shown in Table 20-209.

Return to the Summary Table.

Real Time Counter Second
System CPU must not access this register. Instead, system CPU must access AON_RTC:SEC.VALUE directly.

Table 20-209. RTCSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	SEC	R	0h	Bits 15:0 in AON_RTC:SEC.VALUE. Follow this procedure to get the correct value: - Do two dummy reads of SEC. - Then read SEC until two consecutive reads are equal.

20.8.9.35 RTCSUBSEC Register (Offset = 8Ch) [Reset = 00000000h]

RTCSUBSEC is shown in Table 20-210.

Return to the Summary Table.

Real Time Counter Sub-Second
System CPU must not access this register. Instead, system CPU must access AON_RTC:SUBSEC.VALUE directly.

Table 20-210. RTCSUBSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	SUBSEC	R	0h	Bits 31:16 in AON_RTC:SUBSEC.VALUE. Follow this procedure to get the correct value: - Do two dummy reads SUBSEC. - Then read SUBSEC until two consecutive reads are equal.

20.8.9.36 RTCEVCLR Register (Offset = 90h) [Reset = 00000000h]

RTCEVCLR is shown in Table 20-211.

Return to the Summary Table.

AON_RTC Event Clear
Request to clear events:
- AON_RTC:EVFLAGS.CH2.
- AON_RTC:EVFLAGS.CH2 delayed version.
- AUX_EVCTL:EVSTAT2.AON_RTC_CH2.
- AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY.

Table 20-211. RTCEVCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RTC_CH2_EV_CLR	R/W	0h	Clear events from AON_RTC channel 2. 0: No effect. 1: Clear events from AON_RTC channel 2. Keep RTC_CH2_EV_CLR high until AUX_EVCTL:EVSTAT2.AON_RTC_CH2 and AUX_EVCTL:EVSTAT2.AON_RTC_CH2_DLY are 0.

20.8.9.37 BATMONBAT Register (Offset = 94h) [Reset = 00000000h]

BATMONBAT is shown in Table 20-212.

Return to the Summary Table.

AON_BATMON Battery Voltage Value
Read access to AON_BATMON:BAT. System CPU must not access this register. Instead, system CPU must access AON_BATMON:BAT directly. AON_BATMON:BAT updates during VDDR recharge or active operational mode.

Table 20-212. BATMONBAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	INT	RH	0h	See AON_BATMON:BAT.INT. Follow this procedure to get the correct value: - Do two dummy reads of INT. - Then read INT until two consecutive reads are equal.
7-0	FRAC	R	0h	See AON_BATMON:BAT.FRAC. Follow this procedure to get the correct value: - Do two dummy reads of FRAC. - Then read FRAC until two consecutive reads are equal.

20.8.9.38 BATMONTEMP Register (Offset = 9Ch) [Reset = 00000000h]

BATMONTEMP is shown in Table 20-213.

Return to the Summary Table.

AON_BATMON Temperature Value
Read access to AON_BATMON:TEMP. System CPU must not access this register. Instead, system CPU must access AON_BATMON:TEMP directly. AON_BATMON:TEMP updates during VDDR recharge or active operational mode.

Table 20-213. BATMONTEMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-11	SIGN	R	0h	Sign extension of INT. Follow this procedure to get the correct value: - Do two dummy reads of SIGN. - Then read SIGN until two consecutive reads are equal.
10-2	INT	RH	0h	See AON_BATMON:TEMP.INT. Follow this procedure to get the correct value: - Do two dummy reads of INT. - Then read INT until two consecutive reads are equal.
1-0	FRAC	R	0h	See AON_BATMON:TEMP.FRAC. Follow this procedure to get the correct value: - Do two dummy reads of FRAC. - Then read FRAC until two consecutive reads are equal.

20.8.9.39 TIMERHALT Register (Offset = A0h) [Reset = 00000000h]

TIMERHALT is shown in Table 20-214.

Return to the Summary Table.

Timer Halt
Debug register

Table 20-214. TIMERHALT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	PROGDLY	RH/W	0h	Halt programmable delay. 0: AUX_EVCTL:PROGDLY.VALUE decrements as normal. 1: Halt AUX_EVCTL:PROGDLY.VALUE decrementation.
2	AUX_TIMER2	RH/W	0h	Halt AUX_TIMER2. 0: AUX_TIMER2 operates as normal. 1: Halt AUX_TIMER2 operation.
1	AUX_TIMER1	RH/W	0h	Halt AUX_TIMER01 Timer 1. 0: AUX_TIMER01 Timer 1 operates as normal. 1: Halt AUX_TIMER01 Timer 1 operation.
0	AUX_TIMER0	RH/W	0h	Halt AUX_TIMER01 Timer 0. 0: AUX_TIMER01 Timer 0 operates as normal. 1: Halt AUX_TIMER01 Timer 0 operation.

20.8.9.40 TIMER2BRIDGE Register (Offset = B0h) [Reset = 00000000h]

TIMER2BRIDGE is shown in Table 20-215.

Return to the Summary Table.

AUX_TIMER2 Bridge

Table 20-215. TIMER2BRIDGE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	BUSY	R	0h	Status of bus transactions to AUX_TIMER2. 0: No unfinished bus transactions. 1: A bus transaction is ongoing.

20.8.9.41 SWPWRPROF Register (Offset = B4h) [Reset = 00000000h]

SWPWRPROF is shown in Table 20-216.

Return to the Summary Table.

Software Power Profiler

Table 20-216. SWPWRPROF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2-0	STAT	R/W	0h	Software status bits that can be read by the power profiler.

20.8.10 AUX_SPIM Registers

Table 20-217 lists the memory-mapped registers for the AUX_SPIM registers. All register offset addresses not listed in Table 20-217 should be considered as reserved locations and the register contents should not be modified.

Table 20-217. AUX_SPIM Registers

Offset	Acronym	Register Name	Section
0h	SPIMCFG	SPI Master Configuration	Section 20.8.10.1
4h	MISOCFG	MISO Configuration	Section 20.8.10.2
8h	MOSICTL	MOSI Control	Section 20.8.10.3
Ch	TX8	Transmit 8 Bit	Section 20.8.10.4
10h	TX16	Transmit 16 Bit	Section 20.8.10.5
14h	RX8	Receive 8 Bit	Section 20.8.10.6
18h	RX16	Receive 16 Bit	Section 20.8.10.7
1Ch	SCLKIDLE	SCLK Idle	Section 20.8.10.8
20h	DATAIDLE	Data Idle	Section 20.8.10.9

Complex bit access types are encoded to fit into small table cells. Table 20-218 shows the codes that are used for access types in this section.

Table 20-218. AUX_SPIM Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.10.1 SPIMCFG Register (Offset = 0h) [Reset = 00000000h]

SPIMCFG is shown in Table 20-219.

Return to the Summary Table.

SPI Master Configuration
Write operation stalls until current transfer completes.

Table 20-219. SPIMCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-2	DIV	R/W	0h	SCLK divider. Peripheral clock frequency division gives the SCLK clock frequency. The division factor equals (2 * (DIV+1)): 0x00: Divide by 2. 0x01: Divide by 4. 0x02: Divide by 6. ... 0x3F: Divide by 128.
1	PHA	R/W	0h	Phase of the MOSI and MISO data signals. 0: Sample MISO at leading (odd) edges and shift MOSI at trailing (even) edges of SCLK. 1: Sample MISO at trailing (even) edges and shift MOSI at leading (odd) edges of SCLK.
0	POL	R/W	0h	Polarity of the SCLK signal. 0: SCLK is low when idle, first clock edge rises. 1: SCLK is high when idle, first clock edge falls.

20.8.10.2 MISOCFG Register (Offset = 4h) [Reset = 00000000h]

MISOCFG is shown in Table 20-220.

Return to the Summary Table.

MISO Configuration
Write operation stalls until current transfer completes.

Table 20-220. MISOCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-0	AUXIO	R/W	0h	AUXIO to MISO mux. Select the AUXIO pin that connects to MISO.

20.8.10.3 MOSICTL Register (Offset = 8h) [Reset = 00000000h]

MOSICTL is shown in Table 20-221.

Return to the Summary Table.

MOSI Control
Write operation stalls until current transfer completes.

Table 20-221. MOSICTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	VALUE	W	0h	MOSI level control. 0: Set MOSI low. 1: Set MOSI high.

20.8.10.4 TX8 Register (Offset = Ch) [Reset = 00000000h]

TX8 is shown in Table 20-222.

Return to the Summary Table.

Transmit 8 Bit
Write operation stalls until current transfer completes.

Table 20-222. TX8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	DATA	W	0h	8 bit data transfer. Write DATA to start transfer, MSB first. When transfer completes, MOSI stays at the value of LSB.

20.8.10.5 TX16 Register (Offset = 10h) [Reset = 00000000h]

TX16 is shown in Table 20-223.

Return to the Summary Table.

Transmit 16 Bit
Write operation stalls until current transfer completes.

Table 20-223. TX16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DATA	W	0h	16 bit data transfer. Write DATA to start transfer, MSB first. When transfer completes, MOSI stays at the value of LSB.

20.8.10.6 RX8 Register (Offset = 14h) [Reset = 00000000h]

RX8 is shown in Table 20-224.

Return to the Summary Table.

Receive 8 Bit
Read operation stalls until current transfer completes.

Table 20-224. RX8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	DATA	R	0h	Latest 8 bits received on MISO.

20.8.10.7 RX16 Register (Offset = 18h) [Reset = 00000000h]

RX16 is shown in Table 20-225.

Return to the Summary Table.

Receive 16 Bit
Read operation stalls until current transfer completes.

Table 20-225. RX16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DATA	R	0h	Latest 16 bits received on MISO.

20.8.10.8 SCLKIDLE Register (Offset = 1Ch) [Reset = 00000001h]

SCLKIDLE is shown in Table 20-226.

Return to the Summary Table.

SCLK Idle
Read operation stalls until SCLK is idle with no remaining clock edges.

Table 20-226. SCLKIDLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R	1h	Wait for SCLK idle. Read operation stalls until SCLK is idle with no remaining clock edges. Read then returns 1. AUX_SCE can use this to control CS deassertion.

20.8.10.9 DATAIDLE Register (Offset = 20h) [Reset = 00000001h]

DATAIDLE is shown in Table 20-227.

Return to the Summary Table.

Data Idle
Read operation stalls until current transfer completes.

Table 20-227. DATAIDLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R	1h	Wait for data idle. Read operation stalls until the SCLK period associated with LSB transmission completes. Read then returns 1. AUX_SCE can use this to control CS deassertion.

20.8.11 AUX_MAC Registers

Table 20-228 lists the memory-mapped registers for the AUX_MAC registers. All register offset addresses not listed in Table 20-228 should be considered as reserved locations and the register contents should not be modified.

Table 20-228. AUX_MAC Registers

Offset	Acronym	Register Name	Section
0h	OP0S	Signed Operand 0	Section 20.8.11.1
4h	OP0U	Unsigned Operand 0	Section 20.8.11.2
8h	OP1SMUL	Signed Operand 1 and Multiply	Section 20.8.11.3
Ch	OP1UMUL	Unsigned Operand 1 and Multiply	Section 20.8.11.4
10h	OP1SMAC	Signed Operand 1 and Multiply-Accumulate	Section 20.8.11.5
14h	OP1UMAC	Unsigned Operand 1 and Multiply-Accumulate	Section 20.8.11.6
18h	OP1SADD16	Signed Operand 1 and 16-bit Addition	Section 20.8.11.7
1Ch	OP1UADD16	Unsigned Operand 1 and 16-bit Addition	Section 20.8.11.8
20h	OP1SADD32	Signed Operand 1 and 32-bit Addition	Section 20.8.11.9
24h	OP1UADD32	Unsigned Operand 1 and 32-bit Addition	Section 20.8.11.10
28h	CLZ	Count Leading Zero	Section 20.8.11.11
2Ch	CLS	Count Leading Sign	Section 20.8.11.12
30h	ACCSHIFT	Accumulator Shift	Section 20.8.11.13
34h	ACCRESET	Accumulator Reset	Section 20.8.11.14
38h	ACC15_0	Accumulator Bits 15:0	Section 20.8.11.15
3Ch	ACC16_1	Accumulator Bits 16:1	Section 20.8.11.16
40h	ACC17_2	Accumulator Bits 17:2	Section 20.8.11.17
44h	ACC18_3	Accumulator Bits 18:3	Section 20.8.11.18
48h	ACC19_4	Accumulator Bits 19:4	Section 20.8.11.19
4Ch	ACC20_5	Accumulator Bits 20:5	Section 20.8.11.20
50h	ACC21_6	Accumulator Bits 21:6	Section 20.8.11.21
54h	ACC22_7	Accumulator Bits 22:7	Section 20.8.11.22
58h	ACC23_8	Accumulator Bits 23:8	Section 20.8.11.23
5Ch	ACC24_9	Accumulator Bits 24:9	Section 20.8.11.24
60h	ACC25_10	Accumulator Bits 25:10	Section 20.8.11.25
64h	ACC26_11	Accumulator Bits 26:11	Section 20.8.11.26
68h	ACC27_12	Accumulator Bits 27:12	Section 20.8.11.27
6Ch	ACC28_13	Accumulator Bits 28:13	Section 20.8.11.28
70h	ACC29_14	Accumulator Bits 29:14	Section 20.8.11.29
74h	ACC30_15	Accumulator Bits 30:15	Section 20.8.11.30
78h	ACC31_16	Accumulator Bits 31:16	Section 20.8.11.31
7Ch	ACC32_17	Accumulator Bits 32:17	Section 20.8.11.32
80h	ACC33_18	Accumulator Bits 33:18	Section 20.8.11.33
84h	ACC34_19	Accumulator Bits 34:19	Section 20.8.11.34
88h	ACC35_20	Accumulator Bits 35:20	Section 20.8.11.35
8Ch	ACC36_21	Accumulator Bits 36:21	Section 20.8.11.36
90h	ACC37_22	Accumulator Bits 37:22	Section 20.8.11.37
94h	ACC38_23	Accumulator Bits 38:23	Section 20.8.11.38
98h	ACC39_24	Accumulator Bits 39:24	Section 20.8.11.39
9Ch	ACC39_32	Accumulator Bits 39:32	Section 20.8.11.40

Complex bit access types are encoded to fit into small table cells. Table 20-229 shows the codes that are used for access types in this section.

Table 20-229. AUX_MAC Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.11.1 OP0S Register (Offset = 0h) [Reset = 00000000h]

OP0S is shown in Table 20-230.

Return to the Summary Table.

Signed Operand 0

Table 20-230. OP0S Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP0_VALUE	W	0h	Signed operand 0. Operand for multiply, multiply-and-accumulate, or 32-bit add operations.

20.8.11.2 OP0U Register (Offset = 4h) [Reset = 00000000h]

OP0U is shown in Table 20-231.

Return to the Summary Table.

Unsigned Operand 0

Table 20-231. OP0U Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP0_VALUE	W	0h	Unsigned operand 0. Operand for multiply, multiply-and-accumulate, or 32-bit add operations.

20.8.11.3 OP1SMUL Register (Offset = 8h) [Reset = 00000000h]

OP1SMUL is shown in Table 20-232.

Return to the Summary Table.

Signed Operand 1 and Multiply

Table 20-232. OP1SMUL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Signed operand 1 and multiplication trigger. Write OP1_VALUE to set signed operand 1 and trigger the following operation: When operand 0 was written to OP0S.OP0_VALUE: ACC = OP1_VALUE * OP0S.OP0_VALUE. When operand 0 was written to OP0U.OP0_VALUE: ACC = OP1_VALUE * OP0U.OP0_VALUE.

20.8.11.4 OP1UMUL Register (Offset = Ch) [Reset = 00000000h]

OP1UMUL is shown in Table 20-233.

Return to the Summary Table.

Unsigned Operand 1 and Multiply

Table 20-233. OP1UMUL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Unsigned operand 1 and multiplication trigger. Write OP1_VALUE to set unsigned operand 1 and trigger the following operation: When operand 0 was written to OP0S.OP0_VALUE: ACC = OP1_VALUE * OP0S.OP0_VALUE. When operand 0 was written to OP0U.OP0_VALUE: ACC = OP1_VALUE * OP0U.OP0_VALUE.

20.8.11.5 OP1SMAC Register (Offset = 10h) [Reset = 00000000h]

OP1SMAC is shown in Table 20-234.

Return to the Summary Table.

Signed Operand 1 and Multiply-Accumulate

Table 20-234. OP1SMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Signed operand 1 and multiply-accumulation trigger. Write OP1_VALUE to set signed operand 1 and trigger the following operation: When operand 0 was written to OP0S.OP0_VALUE: ACC = ACC + (OP1_VALUE * OP0S.OP0_VALUE). When operand 0 was written to OP0U.OP0_VALUE: ACC = ACC + (OP1_VALUE * OP0U.OP0_VALUE).

20.8.11.6 OP1UMAC Register (Offset = 14h) [Reset = 00000000h]

OP1UMAC is shown in Table 20-235.

Return to the Summary Table.

Unsigned Operand 1 and Multiply-Accumulate

Table 20-235. OP1UMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Unsigned operand 1 and multiply-accumulation trigger. Write OP1_VALUE to set unsigned operand 1 and trigger the following operation: When operand 0 was written to OP0S.OP0_VALUE: ACC = ACC + (OP1_VALUE * OP0S.OP0_VALUE). When operand 0 was written to OP0U.OP0_VALUE: ACC = ACC + (OP1_VALUE * OP0U.OP0_VALUE).

20.8.11.7 OP1SADD16 Register (Offset = 18h) [Reset = 00000000h]

OP1SADD16 is shown in Table 20-236.

Return to the Summary Table.

Signed Operand 1 and 16-bit Addition

Table 20-236. OP1SADD16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Signed operand 1 and 16-bit addition trigger. Write OP1_VALUE to set signed operand 1 and trigger the following operation: ACC = ACC + OP1_VALUE.

20.8.11.8 OP1UADD16 Register (Offset = 1Ch) [Reset = 00000000h]

OP1UADD16 is shown in Table 20-237.

Return to the Summary Table.

Unsigned Operand 1 and 16-bit Addition

Table 20-237. OP1UADD16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Unsigned operand 1 and 16-bit addition trigger. Write OP1_VALUE to set unsigned operand 1 and trigger the following operation: ACC = ACC + OP1_VALUE.

20.8.11.9 OP1SADD32 Register (Offset = 20h) [Reset = 00000000h]

OP1SADD32 is shown in Table 20-238.

Return to the Summary Table.

Signed Operand 1 and 32-bit Addition

Table 20-238. OP1SADD32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Upper half of signed 32-bit operand and addition trigger. Write OP1_VALUE to set upper half of signed 32-bit operand and trigger the following operation: When lower half of 32-bit operand was written to OP0S.OP0_VALUE: ACC = ACC + ((OP1_VALUE << 16) | OP0S.OP0_VALUE). When lower half of 32-bit operand was written to OP0U.OP0_VALUE: ACC = ACC + ((OP1_VALUE << 16) | OP0U.OP0_VALUE).

20.8.11.10 OP1UADD32 Register (Offset = 24h) [Reset = 00000000h]

OP1UADD32 is shown in Table 20-239.

Return to the Summary Table.

Unsigned Operand 1 and 32-bit Addition

Table 20-239. OP1UADD32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OP1_VALUE	W	0h	Upper half of unsigned 32-bit operand and addition trigger. Write OP1_VALUE to set upper half of unsigned 32-bit operand and trigger the following operation: When lower half of 32-bit operand was written to OP0S.OP0_VALUE: ACC = ACC + ((OP1_VALUE << 16) | OP0S.OP0_VALUE). When lower half of 32-bit operand was written to OP0U.OP0_VALUE: ACC = ACC + ((OP1_VALUE << 16) | OP0U.OP0_VALUE).

20.8.11.11 CLZ Register (Offset = 28h) [Reset = 00000028h]

CLZ is shown in Table 20-240.

Return to the Summary Table.

Count Leading Zero

Table 20-240. CLZ Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	VALUE	R	28h	Number of leading zero bits in the accumulator: 0x00: 0 leading zeros. 0x01: 1 leading zero. ... 0x28: 40 leading zeros (accumulator value is 0).

20.8.11.12 CLS Register (Offset = 2Ch) [Reset = 00000028h]

CLS is shown in Table 20-241.

Return to the Summary Table.

Count Leading Sign

Table 20-241. CLS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	VALUE	R	28h	Number of leading sign bits in the accumulator. When MSB of accumulator is 0, VALUE is number of leading zeros, MSB included. When MSB of accumulator is 1, VALUE is number of leading ones, MSB included. VALUE range is 1 thru 40.

20.8.11.13 ACCSHIFT Register (Offset = 30h) [Reset = 00000000h]

ACCSHIFT is shown in Table 20-242.

Return to the Summary Table.

Accumulator Shift
Only one shift operation can be triggered per register write.

Table 20-242. ACCSHIFT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	LSL1	W	0h	Logic shift left by 1 bit. Write 1 to shift the accumulator one bit to the left, 0 inserted at bit 0.
1	LSR1	W	0h	Logic shift right by 1 bit. Write 1 to shift the accumulator one bit to the right, 0 inserted at bit 39.
0	ASR1	W	0h	Arithmetic shift right by 1 bit. Write 1 to shift the accumulator one bit to the right, previous sign bit inserted at bit 39.

20.8.11.14 ACCRESET Register (Offset = 34h) [Reset = 00000000h]

ACCRESET is shown in Table 20-243.

Return to the Summary Table.

Accumulator Reset

Table 20-243. ACCRESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	TRG	W	0h	Write any value to this register to trigger a reset of all bits in the accumulator.

20.8.11.15 ACC15_0 Register (Offset = 38h) [Reset = 00000000h]

ACC15_0 is shown in Table 20-244.

Return to the Summary Table.

Accumulator Bits 15:0

Table 20-244. ACC15_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Value of the accumulator, bits 15:0. Write VALUE to initialize bits 15:0 of accumulator.

20.8.11.16 ACC16_1 Register (Offset = 3Ch) [Reset = 00000000h]

ACC16_1 is shown in Table 20-245.

Return to the Summary Table.

Accumulator Bits 16:1

Table 20-245. ACC16_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 16:1.

20.8.11.17 ACC17_2 Register (Offset = 40h) [Reset = 00000000h]

ACC17_2 is shown in Table 20-246.

Return to the Summary Table.

Accumulator Bits 17:2

Table 20-246. ACC17_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 17:2.

20.8.11.18 ACC18_3 Register (Offset = 44h) [Reset = 00000000h]

ACC18_3 is shown in Table 20-247.

Return to the Summary Table.

Accumulator Bits 18:3

Table 20-247. ACC18_3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 18:3.

20.8.11.19 ACC19_4 Register (Offset = 48h) [Reset = 00000000h]

ACC19_4 is shown in Table 20-248.

Return to the Summary Table.

Accumulator Bits 19:4

Table 20-248. ACC19_4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 19:4.

20.8.11.20 ACC20_5 Register (Offset = 4Ch) [Reset = 00000000h]

ACC20_5 is shown in Table 20-249.

Return to the Summary Table.

Accumulator Bits 20:5

Table 20-249. ACC20_5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 20:5.

20.8.11.21 ACC21_6 Register (Offset = 50h) [Reset = 00000000h]

ACC21_6 is shown in Table 20-250.

Return to the Summary Table.

Accumulator Bits 21:6

Table 20-250. ACC21_6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 21:6.

20.8.11.22 ACC22_7 Register (Offset = 54h) [Reset = 00000000h]

ACC22_7 is shown in Table 20-251.

Return to the Summary Table.

Accumulator Bits 22:7

Table 20-251. ACC22_7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 22:7.

20.8.11.23 ACC23_8 Register (Offset = 58h) [Reset = 00000000h]

ACC23_8 is shown in Table 20-252.

Return to the Summary Table.

Accumulator Bits 23:8

Table 20-252. ACC23_8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 23:8.

20.8.11.24 ACC24_9 Register (Offset = 5Ch) [Reset = 00000000h]

ACC24_9 is shown in Table 20-253.

Return to the Summary Table.

Accumulator Bits 24:9

Table 20-253. ACC24_9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 24:9.

20.8.11.25 ACC25_10 Register (Offset = 60h) [Reset = 00000000h]

ACC25_10 is shown in Table 20-254.

Return to the Summary Table.

Accumulator Bits 25:10

Table 20-254. ACC25_10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 25:10.

20.8.11.26 ACC26_11 Register (Offset = 64h) [Reset = 00000000h]

ACC26_11 is shown in Table 20-255.

Return to the Summary Table.

Accumulator Bits 26:11

Table 20-255. ACC26_11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 26:11.

20.8.11.27 ACC27_12 Register (Offset = 68h) [Reset = 00000000h]

ACC27_12 is shown in Table 20-256.

Return to the Summary Table.

Accumulator Bits 27:12

Table 20-256. ACC27_12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 27:12.

20.8.11.28 ACC28_13 Register (Offset = 6Ch) [Reset = 00000000h]

ACC28_13 is shown in Table 20-257.

Return to the Summary Table.

Accumulator Bits 28:13

Table 20-257. ACC28_13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 28:13.

20.8.11.29 ACC29_14 Register (Offset = 70h) [Reset = 00000000h]

ACC29_14 is shown in Table 20-258.

Return to the Summary Table.

Accumulator Bits 29:14

Table 20-258. ACC29_14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 29:14.

20.8.11.30 ACC30_15 Register (Offset = 74h) [Reset = 00000000h]

ACC30_15 is shown in Table 20-259.

Return to the Summary Table.

Accumulator Bits 30:15

Table 20-259. ACC30_15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 30:15.

20.8.11.31 ACC31_16 Register (Offset = 78h) [Reset = 00000000h]

ACC31_16 is shown in Table 20-260.

Return to the Summary Table.

Accumulator Bits 31:16

Table 20-260. ACC31_16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Value of the accumulator, bits 31:16. Write VALUE to initialize bits 31:16 of accumulator.

20.8.11.32 ACC32_17 Register (Offset = 7Ch) [Reset = 00000000h]

ACC32_17 is shown in Table 20-261.

Return to the Summary Table.

Accumulator Bits 32:17

Table 20-261. ACC32_17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 32:17.

20.8.11.33 ACC33_18 Register (Offset = 80h) [Reset = 00000000h]

ACC33_18 is shown in Table 20-262.

Return to the Summary Table.

Accumulator Bits 33:18

Table 20-262. ACC33_18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 33:18.

20.8.11.34 ACC34_19 Register (Offset = 84h) [Reset = 00000000h]

ACC34_19 is shown in Table 20-263.

Return to the Summary Table.

Accumulator Bits 34:19

Table 20-263. ACC34_19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 34:19.

20.8.11.35 ACC35_20 Register (Offset = 88h) [Reset = 00000000h]

ACC35_20 is shown in Table 20-264.

Return to the Summary Table.

Accumulator Bits 35:20

Table 20-264. ACC35_20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 35:20.

20.8.11.36 ACC36_21 Register (Offset = 8Ch) [Reset = 00000000h]

ACC36_21 is shown in Table 20-265.

Return to the Summary Table.

Accumulator Bits 36:21

Table 20-265. ACC36_21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 36:21.

20.8.11.37 ACC37_22 Register (Offset = 90h) [Reset = 00000000h]

ACC37_22 is shown in Table 20-266.

Return to the Summary Table.

Accumulator Bits 37:22

Table 20-266. ACC37_22 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 37:22.

20.8.11.38 ACC38_23 Register (Offset = 94h) [Reset = 00000000h]

ACC38_23 is shown in Table 20-267.

Return to the Summary Table.

Accumulator Bits 38:23

Table 20-267. ACC38_23 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 38:23.

20.8.11.39 ACC39_24 Register (Offset = 98h) [Reset = 00000000h]

ACC39_24 is shown in Table 20-268.

Return to the Summary Table.

Accumulator Bits 39:24

Table 20-268. ACC39_24 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	Value of the accumulator, bits 39:24.

20.8.11.40 ACC39_32 Register (Offset = 9Ch) [Reset = 00000000h]

ACC39_32 is shown in Table 20-269.

Return to the Summary Table.

Accumulator Bits 39:32

Table 20-269. ACC39_32 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	VALUE	R/W	0h	Value of the accumulator, bits 39:32. Write VALUE to initialize bits 39:32 of accumulator.

20.8.12 AUX_SCE Registers

Table 20-270 lists the memory-mapped registers for the AUX_SCE registers. All register offset addresses not listed in Table 20-270 should be considered as reserved locations and the register contents should not be modified.

Table 20-270. AUX_SCE Registers

Offset	Acronym	Register Name	Section
0h	CTL	Internal	Section 20.8.12.1
4h	FETCHSTAT	Internal	Section 20.8.12.2
8h	CPUSTAT	Internal	Section 20.8.12.3
Ch	WUSTAT	Internal	Section 20.8.12.4
10h	REG1_0	Internal	Section 20.8.12.5
14h	REG3_2	Internal	Section 20.8.12.6
18h	REG5_4	Internal	Section 20.8.12.7
1Ch	REG7_6	Internal	Section 20.8.12.8
20h	LOOPADDR	Internal	Section 20.8.12.9
24h	LOOPCNT	Internal	Section 20.8.12.10
28h	NONSECDDIACC0	Non-Secure DDI Access 0	Section 20.8.12.11
2Ch	NONSECDDIACC1	Non-Secure DDI Access 1	Section 20.8.12.12
30h	NONSECDDIACC2	Non-Secure DDI Access 2	Section 20.8.12.13
34h	NONSECDDIACC3	Non-Secure DDI Access 3	Section 20.8.12.14

Complex bit access types are encoded to fit into small table cells. Table 20-271 shows the codes that are used for access types in this section.

Table 20-271. AUX_SCE Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.8.12.1 CTL Register (Offset = 0h) [Reset = 00000000h]

CTL is shown in Table 20-272.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-272. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	FORCE_EV_LOW	R/W	0h	Internal. Only to be used through TI provided API.
23-16	FORCE_EV_HIGH	R/W	0h	Internal. Only to be used through TI provided API.
15-8	RESET_VECTOR	R/W	0h	Internal. Only to be used through TI provided API.
7	RESERVED	R	0h	Reserved
6	DBG_FREEZE_EN	R/W	0h	Internal. Only to be used through TI provided API.
5	FORCE_WU_LOW	R/W	0h	Internal. Only to be used through TI provided API.
4	FORCE_WU_HIGH	R/W	0h	Internal. Only to be used through TI provided API.
3	RESTART	R/W	0h	Internal. Only to be used through TI provided API.
2	SINGLE_STEP	R/W	0h	Internal. Only to be used through TI provided API.
1	SUSPEND	R/W	0h	Internal. Only to be used through TI provided API.
0	CLK_EN	R/W	0h	Internal. Only to be used through TI provided API.

20.8.12.2 FETCHSTAT Register (Offset = 4h) [Reset = 00000000h]

FETCHSTAT is shown in Table 20-273.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-273. FETCHSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	OPCODE	R	0h	Internal. Only to be used through TI provided API.
15-0	PC	R	0h	Internal. Only to be used through TI provided API.

20.8.12.3 CPUSTAT Register (Offset = 8h) [Reset = 00000000h]

CPUSTAT is shown in Table 20-274.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-274. CPUSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	BUS_ERROR	R	0h	Internal. Only to be used through TI provided API.
10	SLEEP	R	0h	Internal. Only to be used through TI provided API.
9	WEV	R	0h	Internal. Only to be used through TI provided API.
8	HALTED	R	0h	Internal. Only to be used through TI provided API.
7-4	RESERVED	R	0h	Reserved
3	V_FLAG	R	0h	Internal. Only to be used through TI provided API.
2	C_FLAG	R	0h	Internal. Only to be used through TI provided API.
1	N_FLAG	R	0h	Internal. Only to be used through TI provided API.
0	Z_FLAG	R	0h	Internal. Only to be used through TI provided API.

20.8.12.4 WUSTAT Register (Offset = Ch) [Reset = 00000000h]

WUSTAT is shown in Table 20-275.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-275. WUSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	EXC_VECTOR	R	0h	Internal. Only to be used through TI provided API.
15-9	RESERVED	R	0h	Reserved
8	WU_SIGNAL	R	0h	Internal. Only to be used through TI provided API.
7-0	EV_SIGNALS	R	0h	Internal. Only to be used through TI provided API.

20.8.12.5 REG1_0 Register (Offset = 10h) [Reset = 00000000h]

REG1_0 is shown in Table 20-276.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-276. REG1_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	REG1	R	0h	Internal. Only to be used through TI provided API.
15-0	REG0	R	0h	Internal. Only to be used through TI provided API.

20.8.12.6 REG3_2 Register (Offset = 14h) [Reset = 00000000h]

REG3_2 is shown in Table 20-277.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-277. REG3_2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	REG3	R	0h	Internal. Only to be used through TI provided API.
15-0	REG2	R	0h	Internal. Only to be used through TI provided API.

20.8.12.7 REG5_4 Register (Offset = 18h) [Reset = 00000000h]

REG5_4 is shown in Table 20-278.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-278. REG5_4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	REG5	R	0h	Internal. Only to be used through TI provided API.
15-0	REG4	R	0h	Internal. Only to be used through TI provided API.

20.8.12.8 REG7_6 Register (Offset = 1Ch) [Reset = 00000000h]

REG7_6 is shown in Table 20-279.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-279. REG7_6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	REG7	R	0h	Internal. Only to be used through TI provided API.
15-0	REG6	R	0h	Internal. Only to be used through TI provided API.

20.8.12.9 LOOPADDR Register (Offset = 20h) [Reset = 00000000h]

LOOPADDR is shown in Table 20-280.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-280. LOOPADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	STOP	R	0h	Internal. Only to be used through TI provided API.
15-0	START	R	0h	Internal. Only to be used through TI provided API.

20.8.12.10 LOOPCNT Register (Offset = 24h) [Reset = 00000000h]

LOOPCNT is shown in Table 20-281.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 20-281. LOOPCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	ITER_LEFT	R	0h	Internal. Only to be used through TI provided API.

20.8.12.11 NONSECDDIACC0 Register (Offset = 28h) [Reset = 00000000h]

NONSECDDIACC0 is shown in Table 20-282.

Return to the Summary Table.

Non-Secure DDI Access 0
When system is in secure state, AUX_SCE is allowed to update a predefined DDI half-word using SET or CLR access. Configuration will determine if AUX_SCE can read the same half-word. An access to a non-allowed register will suspend the AUX_SCE when system state is secure.
If ADDR field in two or more NONSECDDIACC registers are equal, the MASK and RD_EN from the highest numbered register will be used.
Examples:
Half-word with address of 0 corresponds to DDI_0_OSC:CTL0 bit range [15:0].
Half-word with address of 1 corresponds to DDI_0_OSC:CTL0 bit range [31:16].
…
Half-word with address of 34 corresponds to DDI_0_OSC:STAT2 bit range [15:0].
Half-word with address of 35 corresponds to DDI_0_OSC:STAT2 bit range [31:16].

Table 20-282. NONSECDDIACC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22	RD_EN	R/W	0h	Read Enable 0: AUX_SCE is not allowed to read DDI half-word given by ADDR. 1: AUX_SCE is allowed to read DDI half-word given by ADDR.
21-16	ADDR	R/W	0h	Address AUX_SCE is allowed to update this DDI half-word using SET or CLR access.
15-0	WR_MASK	R/W	0h	Mask AUX_SCE is allowed to update bits in half-word given by ADDR according to this bit mask.

20.8.12.12 NONSECDDIACC1 Register (Offset = 2Ch) [Reset = 00000000h]

NONSECDDIACC1 is shown in Table 20-283.

Return to the Summary Table.

Non-Secure DDI Access 1
When system is in secure state, AUX_SCE is allowed to update a predefined DDI half-word using SET or CLR access. Configuration will determine if AUX_SCE can read the same half-word. An access to a non-allowed register will suspend the AUX_SCE when system state is secure.
If ADDR field in two or more NONSECDDIACC registers are equal, the MASK and RD_EN from the highest numbered register will be used.
Examples:
Half-word with address of 0 corresponds to DDI_0_OSC:CTL0 bit range [15:0].
Half-word with address of 1 corresponds to DDI_0_OSC:CTL0 bit range [31:16].
…
Half-word with address of 34 corresponds to DDI_0_OSC:STAT2 bit range [15:0].
Half-word with address of 35 corresponds to DDI_0_OSC:STAT2 bit range [31:16].

Table 20-283. NONSECDDIACC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22	RD_EN	R/W	0h	Read Enable 0: AUX_SCE is not allowed to read DDI half-word given by ADDR. 1: AUX_SCE is allowed to read DDI half-word given by ADDR.
21-16	ADDR	R/W	0h	Address AUX_SCE is allowed to update this DDI half-word using SET or CLR access.
15-0	WR_MASK	R/W	0h	Mask AUX_SCE is allowed to update bits in half-word given by ADDR according to this bit mask.

20.8.12.13 NONSECDDIACC2 Register (Offset = 30h) [Reset = 00000000h]

NONSECDDIACC2 is shown in Table 20-284.

Return to the Summary Table.

Non-Secure DDI Access 2
When system is in secure state, AUX_SCE is allowed to update a predefined DDI half-word using SET or CLR access. Configuration will determine if AUX_SCE can read the same half-word. An access to a non-allowed register will suspend the AUX_SCE when system state is secure.
If ADDR field in two or more NONSECDDIACC registers are equal, the MASK and RD_EN from the highest numbered register will be used.
Examples:
Half-word with address of 0 corresponds to DDI_0_OSC:CTL0 bit range [15:0].
Half-word with address of 1 corresponds to DDI_0_OSC:CTL0 bit range [31:16].
…
Half-word with address of 34 corresponds to DDI_0_OSC:STAT2 bit range [15:0].
Half-word with address of 35 corresponds to DDI_0_OSC:STAT2 bit range [31:16].

Table 20-284. NONSECDDIACC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22	RD_EN	R/W	0h	Read Enable 0: AUX_SCE is not allowed to read DDI half-word given by ADDR. 1: AUX_SCE is allowed to read DDI half-word given by ADDR.
21-16	ADDR	R/W	0h	Address AUX_SCE is allowed to update this DDI half-word using SET or CLR access.
15-0	WR_MASK	R/W	0h	Mask AUX_SCE is allowed to update bits in half-word given by ADDR according to this bit mask.

20.8.12.14 NONSECDDIACC3 Register (Offset = 34h) [Reset = 00000000h]

NONSECDDIACC3 is shown in Table 20-285.

Return to the Summary Table.

Non-Secure DDI Access 3
When system is in secure state, AUX_SCE is allowed to update a predefined DDI half-word using SET or CLR access. Configuration will determine if AUX_SCE can read the same half-word. An access to a non-allowed register will suspend the AUX_SCE when system state is secure.
If ADDR field in two or more NONSECDDIACC registers are equal, the MASK and RD_EN from the highest numbered register will be used.
Examples:
Half-word with address of 0 corresponds to DDI_0_OSC:CTL0 bit range [15:0].
Half-word with address of 1 corresponds to DDI_0_OSC:CTL0 bit range [31:16].
…
Half-word with address of 34 corresponds to DDI_0_OSC:STAT2 bit range [15:0].
Half-word with address of 35 corresponds to DDI_0_OSC:STAT2 bit range [31:16].

Table 20-285. NONSECDDIACC3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22	RD_EN	R/W	0h	Read Enable 0: AUX_SCE is not allowed to read DDI half-word given by ADDR. 1: AUX_SCE is allowed to read DDI half-word given by ADDR.
21-16	ADDR	R/W	0h	Address AUX_SCE is allowed to update this DDI half-word using SET or CLR access.
15-0	WR_MASK	R/W	0h	Mask AUX_SCE is allowed to update bits in half-word given by ADDR according to this bit mask.

This page intentionally left blank.

Chapter 21

Battery Monitor and Temperature Sensor (BATMON)

This chapter describes the battery monitor and temperature sensor in the CC13x4x10 and CC26x4x10 device platform.

21.1 Introduction 1822
21.2 Functional Description 1822
21.3 AON_BATMON Registers 1823

21.1 Introduction

BATMON is a module automatically enabled at boot that monitors both the VDDS supply voltage and the temperature through an on-chip temperature sensor. When enabled, the battery and temperature monitor module is operational in all operation modes except the deepest power modes, Standby and Shutdown. When the device is in Standby, the measurements of the battery monitor module (both temperature and battery voltage) will be limited to recharge cycles. At least two measurements will be performed in each recharge cycle.

The battery monitor provides voltage and temperature information to several modules, including the Flash and the Radio. This is done to ensure correct operation and lowest power consumption. Therefore, it is not recommended to modify any settings in the battery monitor or turn it off.

21.2 Functional Description

The battery monitor is a 7-bit SAR-like ADC running at 125 kHz that performs alternate measurements of the supply voltage and the temperature sensor. When the battery monitor has settled on its first measurement, it stops working in SAR mode and starts linear tracking of voltage and temperature. A small digital core transforms these measurements to voltage and temperature in °C, which are read directly from the AON_BATMON:BAT and AON_BATMON:TEMP registers.

When a change in supply voltage or temperature is detected, the battery monitor will solely track the voltage until it has settled on a new constant level. The resolution of the ADC and the 125 kHz clock speed will limit the battery monitors capability of measuring voltage spikes. Due to the battery monitor not only alternating between temperature and battery voltage, but also between checking if there has been a positive or negative change since last read, there can be a delay of 6 clock cycles between a voltage dip and the time when the ADC notices that the temperature or voltage has changed. Due to the prioritization of voltage tracking upon detection of voltage changes, temperature changes might be detected with more delays if the voltage is also changing at the same time. This is important to keep in mind because the battery monitor is designed to measure the battery voltage; it is not designed to measure voltage spurs due to short periods of higher current consumption.

The module includes an event register, AON_BATMON:EVENT, that includes six events:

- TEMP_UPDATE: indicates that the temperature has changed
- BATT_UPDATE: indicates that the voltage has changed
- TEMP_BELOW_LL: indicates that the temperature is below the lower limit value that is set in the AON_BATMON:TEMPLL register
- TEMP_OVER_UL: indicates that the temperature is over the upper limit value that is set in the AON_BATMON:TEMPUL register
- BATT_BELOW_LL: indicates that the voltage is below the lower limit value that is set in the AON_BATMON:BATTLL register
- BATT_OVER_UL: indicates that the voltage is over the upper limit value that is set in the AON_BATMON:BATTUL register

These events must be cleared by writing to the AON_BATMON:EVENT register. The events will be asserted again if the conditions for the events are met (assertion of the new events takes precedence over the clearing of the events). In addition to the individual events listed previously, the battery monitor module has a combined event that is connected to the CPU as an interrupt line. The mask register, AON_BATMON:EVENTMASK, can be used to select which of the events in AON_BATMON:EVENT contribute to the combined event. These events are connected to the AON event fabric. For details, see Chapter 5.

21.3 AON_BATMON Registers

Table 21-1 lists the memory-mapped registers for the AON_BATMON registers. All register offset addresses not listed in Table 21-1 should be considered as reserved locations and the register contents should not be modified.

Table 21-1. AON_BATMON Registers

Offset	Acronym	Register Name	Section
0h	CTL	Internal	Section 21.3.1
4h	MEASCFG	Internal	Section 21.3.2
Ch	TEMPP0	Internal	Section 21.3.3
10h	TEMPP1	Internal	Section 21.3.4
14h	TEMPP2	Internal	Section 21.3.5
18h	BATMONP0	Internal	Section 21.3.6
1Ch	BATMONP1	Internal	Section 21.3.7
20h	IOSTRP0	Internal	Section 21.3.8
24h	FLASHPUMPP0	Internal	Section 21.3.9
28h	BAT	Last Measured Battery Voltage	Section 21.3.10
2Ch	BATUPD	Battery Update	Section 21.3.11
30h	TEMP	Temperature	Section 21.3.12
34h	TEMPUPD	Temperature Update	Section 21.3.13
48h	EVENTMASK	Event Mask	Section 21.3.14
4Ch	EVENT	Event	Section 21.3.15
50h	BATTUL	Battery Upper Limit.	Section 21.3.16
54h	BATTLL	Battery Lower Limit	Section 21.3.17
58h	TEMPUL	Temperature Upper Limit	Section 21.3.18
5Ch	TEMPLL	Temperature Lower Limit	Section 21.3.19

Complex bit access types are encoded to fit into small table cells. Table 21-2 shows the codes that are used for access types in this section.

Table 21-2. AON_BATMON Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

21.3.1 CTL Register (Offset = 0h) [Reset = 00000000h]

CTL is shown in Table 21-3.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-3. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	CALC_EN	R/W	0h	Internal. Only to be used through TI provided API.
0	MEAS_EN	R/W	0h	Internal. Only to be used through TI provided API.

21.3.2 MEASCFG Register (Offset = 4h) [Reset = 00000000h]

MEASCFG is shown in Table 21-4.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-4. MEASCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1-0	PER	R/W	0h	Internal. Only to be used through TI provided API.

21.3.3 TEMPP0 Register (Offset = Ch) [Reset = 00000000h]

TEMPP0 is shown in Table 21-5.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-5. TEMPP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.4 TEMPP1 Register (Offset = 10h) [Reset = 00000000h]

TEMPP1 is shown in Table 21-6.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-6. TEMPP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.5 TEMPP2 Register (Offset = 14h) [Reset = 00000000h]

TEMPP2 is shown in Table 21-7.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-7. TEMPP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.6 BATMONP0 Register (Offset = 18h) [Reset = 00000000h]

BATMONP0 is shown in Table 21-8.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-8. BATMONP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.7 BATMONP1 Register (Offset = 1Ch) [Reset = 00000000h]

BATMONP1 is shown in Table 21-9.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-9. BATMONP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.8 IOSTRP0 Register (Offset = 20h) [Reset = 00000028h]

IOSTRP0 is shown in Table 21-10.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-10. IOSTRP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-4	CFG2	R/W	2h	Internal. Only to be used through TI provided API.
3-0	CFG1	R/W	8h	Internal. Only to be used through TI provided API.

21.3.9 FLASHPUMPP0 Register (Offset = 24h) [Reset = 00000000h]

FLASHPUMPP0 is shown in Table 21-11.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 21-11. FLASHPUMPP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	DIS_NOISE_FILTER	R/W	0h	Internal. Only to be used through TI provided API.
8	FALLB	R/W	0h	Internal. Only to be used through TI provided API.
7-6	HIGHLIM	R/W	0h	Internal. Only to be used through TI provided API.
5	LOWLIM	R/W	0h	Internal. Only to be used through TI provided API.
4	OVR	R/W	0h	Internal. Only to be used through TI provided API.
3-0	CFG	R/W	0h	Internal. Only to be used through TI provided API.

21.3.10 BAT Register (Offset = 28h) [Reset = 00000000h]

BAT is shown in Table 21-12.

Return to the Summary Table.

Last Measured Battery Voltage
Total Battery Voltage = INT + FRAC
It is a sum of the integer and fraction parts.
This register has to be read when BATUPD.STAT = 1

Table 21-12. BAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	INT	R	0h	Integer part: 0x0: Battery voltage = 0V + fractional part ... 0x3: Battery voltage = 3V + fractional part 0x4: Battery voltage = 4V + fractional part
7-0	FRAC	R	0h	Fractional part, standard binary fractional encoding. 0x00: .0V ... 0x20: 1/8 = .125V 0x40: 1/4 = .25V 0x80: 1/2 = .5V ... 0xA0: 1/2 + 1/8 = .625V ... 0xFF: 1/2 + 1/4 + 1/8 + .. + 1/256 = 0.99V

21.3.11 BATUPD Register (Offset = 2Ch) [Reset = 00000000h]

BATUPD is shown in Table 21-13.

Return to the Summary Table.

Battery Update
Indicates BAT Updates

Table 21-13. BATUPD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W1C	0h	0: No update since last clear 1: New battery voltage is present. Write 1 to clear the status.

21.3.12 TEMP Register (Offset = 30h) [Reset = 00000000h]

TEMP is shown in Table 21-14.

Return to the Summary Table.

Temperature
Last Measured Temperature in Degrees Celsius
This register has to be read when TEMPUPD.STAT = 1.

Table 21-14. TEMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-8	INT	R	0h	Integer part of temperature value (signed) Total value = INT + FRAC 2's complement encoding 0x100: Min value 0x1D8: -40C 0x1FF: -1C 0x00: 0C 0x1B: 27C 0x55: 85C 0xFF: Max value
7-6	FRAC	R	0h	Fractional part of temperature value. Total value = INT + FRAC The encoding is an extension of the 2's complement encoding. 00: 0.0C 01: 0.25C 10: 0.5C 11: 0.75C For example: 000000001,00 = (1+0,00) = 1,00 000000000,11 = (0+0,75) = 0,75 000000000,10 = (0+0,50) = 0,50 000000000,01 = (0+0,25) = 0,25 000000000,00 = (0+0,00) = 0,00 111111111,11 = (-1+0,75) = -0,25 111111111,10 = (-1+0,50) = -0,50 111111111,01 = (-1+0,25) = -0,75 111111111,00 = (-1+0,00) = -1,00 111111110,11 = (-2+0,75) = -1,25
5-0	RESERVED	R	0h	Reserved

21.3.13 TEMPUPD Register (Offset = 34h) [Reset = 00000000h]

TEMPUPD is shown in Table 21-15.

Return to the Summary Table.

Temperature Update
Indicates TEMP Updates

Table 21-15. TEMPUPD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	STAT	R/W1C	0h	0: No update since last clear 1: New temperature is present. Write 1 to clear the status.

21.3.14 EVENTMASK Register (Offset = 48h) [Reset = 00000000h]

EVENTMASK is shown in Table 21-16.

Return to the Summary Table.

Event Mask

Table 21-16. EVENTMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	TEMP_UPDATE_MASK	R/W	0h	1: EVENT.TEMP_UPDATE contributes to combined event from BATMON 0: EVENT.TEMP_UPDATE does not contribute to combined event from BATMON
4	BATT_UPDATE_MASK	R/W	0h	1: EVENT.BATT_UPDATE contributes to combined event from BATMON 0: EVENT.BATT_UPDATE does not contribute to combined event from BATMON
3	TEMP_BELOW_LL_MASK	R/W	0h	1: EVENT.TEMP_BELOW_LL contributes to combined event from BATMON 0: EVENT.TEMP_BELOW_LL does not contribute to combined event from BATMON
2	TEMP_OVER_UL_MASK	R/W	0h	1: EVENT.TEMP_OVER_UL contributes to combined event from BATMON 0: EVENT.TEMP_OVER_UL does not contribute to combined event from BATMON
1	BATT_BELOW_LL_MASK	R/W	0h	1: EVENT.BATT_BELOW_LL contributes to combined event from BATMON 0: EVENT.BATT_BELOW_LL does not contribute to combined event from BATMON
0	BATT_OVER_UL_MASK	R/W	0h	1: EVENT.BATT_OVER_UL contributes to combined event from BATMON 0: EVENT.BATT_OVER_UL does not contribute to combined event from BATMON

21.3.15 EVENT Register (Offset = 4Ch) [Reset = 00000000h]

EVENT is shown in Table 21-17.

Return to the Summary Table.

Event

Table 21-17. EVENT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	TEMP_UPDATE	R/W1C	0h	Alias to TEMPUPD.STAT
4	BATT_UPDATE	R/W1C	0h	Alias to BATUPD.STAT
3	TEMP_BELOW_LL	R/W1C	0h	Read: 1: Temperature level is below the lower limit set by TEMPLL. 0: Temperature level is not below the lower limit set by TEMPLL. Write: 1: Clears the flag 0: No change in the flag
2	TEMP_OVER_UL	R/W1C	0h	Read: 1: Temperature level is above the upper limit set by TEMPUL. 0: Temperature level is not above the upper limit set by TEMPUL. Write: 1: Clears the flag 0: No change in the flag
1	BATT_BELOW_LL	R/W1C	0h	Read: 1: Battery level is below the lower limit set by BATTLL. 0: Battery level is not below the lower limit set by BATTLL. Write: 1: Clears the flag 0: No change in the flag
0	BATT_OVER_UL	R/W1C	0h	Read: 1: Battery level is above the upper limit set by BATTUL. 0: Battery level is not above the upper limit set by BATTUL. Write: 1: Clears the flag 0: No change in the flag

21.3.16 BATTUL Register (Offset = 50h) [Reset = 000007FFh]

BATTUL is shown in Table 21-18.

Return to the Summary Table.

Battery Upper Limit.
Total battery voltage = INT + FRAC
It is a sum of integer and fractional parts

Table 21-18. BATTUL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	INT	R/W	7h	Integer part: Total battery voltage = INT + FRAC (integer and fractional part) 0x0: Battery voltage = 0V + fractional part ... 0x3: Battery voltage = 3V + fractional part 0x4: Battery voltage = 4V + fractional part
7-0	FRAC	R/W	FFh	Fractional part, standard binary fractional encoding. 0x00: .0V ... 0x20: 1/8 = .125V 0x40: 1/4 = .25V 0x80: 1/2 = .5V ... 0xA0: 1/2 + 1/8 = .625V ... 0xFF: 1/2 + 1/4 + 1/8 + .. + 1/256 = 0.99V

21.3.17 BATTLL Register (Offset = 54h) [Reset = 00000000h]

BATTLL is shown in Table 21-19.

Return to the Summary Table.

Battery Lower Limit
Total battery voltage = INT + FRAC
It is a sum of integer and fractional parts

Table 21-19. BATTLL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	INT	R/W	0h	Integer part: Total battery voltage = INT + FRAC (integer and fractional part) 0x0: Battery voltage = 0V + fractional part ... 0x3: Battery voltage = 3V + fractional part 0x4: Battery voltage = 4V + fractional part
7-0	FRAC	R/W	0h	Fractional part, standard binary fractional encoding. 0x00: .0V ... 0x20: 1/8 = .125V 0x40: 1/4 = .25V 0x80: 1/2 = .5V ... 0xA0: 1/2 + 1/8 = .625V ... 0xFF: 1/2 + 1/4 + 1/8 + .. + 1/256 = 0.99V

21.3.18 TEMPUL Register (Offset = 58h) [Reset = 0000FFC0h]

TEMPUL is shown in Table 21-20.

Return to the Summary Table.

Temperature Upper Limit

Table 21-20. TEMPUL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-8	INT	R/W	FFh	Integer part (signed) of temperature upper limit. Total value = INTEGER + FRACTIONAL 2's complement encoding 0x100: Min value 0x1D8: -40C 0x1FF: -1C 0x00: 0C 0x1B: 27C 0x55: 85C 0xFF: Max value
7-6	FRAC	R/W	3h	Fractional part of temperature upper limit. Total value = INTEGER + FRACTIONAL The encoding is an extension of the 2's complement encoding. 00: 0.0C 01: 0.25C 10: 0.5C 11: 0.75C For example: 000000001,00 = (1+0,00) = 1,00 000000000,11 = (0+0,75) = 0,75 000000000,10 = (0+0,50) = 0,50 000000000,01 = (0+0,25) = 0,25 000000000,00 = (0+0,00) = 0,00 111111111,11 = (-1+0,75) = -0,25 111111111,10 = (-1+0,50) = -0,50 111111111,01 = (-1+0,25) = -0,75 111111111,00 = (-1+0,00) = -1,00 111111110,11 = (-2+0,75) = -1,25
5-0	RESERVED	R	0h	Reserved

21.3.19 TEMPLL Register (Offset = 5Ch) [Reset = 00010000h]

TEMPLL is shown in Table 21-21.

Return to the Summary Table.

Temperature Lower Limit

Table 21-21. TEMPLL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-8	INT	R/W	100h	Integer part (signed) of temperature lower limit. Total value = INTEGER + FRACTIONAL 2's complement encoding 0x100: Min value 0x1D8: -40C 0x1FF: -1C 0x00: 0C 0x1B: 27C 0x55: 85C 0xFF: Max value
7-6	FRAC	R/W	0h	Fractional part of temperature lower limit. Total value = INTEGER + FRACTIONAL The encoding is an extension of the 2's complement encoding. 00: 0.0C 01: 0.25C 10: 0.5C 11: 0.75C For example: 000000001,00 = (1+0,00) = 1,00 000000000,11 = (0+0,75) = 0,75 000000000,10 = (0+0,50) = 0,50 000000000,01 = (0+0,25) = 0,25 000000000,00 = (0+0,00) = 0,00 111111111,11 = (-1+0,75) = -0,25 111111111,10 = (-1+0,50) = -0,50 111111111,01 = (-1+0,25) = -0,75 111111111,00 = (-1+0,00) = -1,00 111111110,11 = (-2+0,75) = -1,25
5-0	RESERVED	R	0h	Reserved

Chapter 22

Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the Universal Asynchronous Receiver/Transmitter (UART).

22.1 Introduction 1844
22.2 Block Diagram 1845
22.3 Signal Description 1845
22.4 Functional Description 1845
22.5 Interface to µDMA 1850
22.6 Initialization and Configuration 1851
22.7 UART Registers 1852

22.1 Introduction

The controller of the CC13x4x10 and CC26x4x10 device platform includes a UART with the following features:

- Programmable baud rate generator allowing speeds up to 3 Mbps
- Separate 32 × 8 transmit (TX) and 32 × 12 receive (RX) first-in first-out (FIFO) buffers to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of ⅛, ¼, ½, ¾, and ⅞
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no parity bit generation and detection
  - 1 or 2 stop-bit generation
- Support for modem control functions CTS and RTS
- Independent masking of the TX FIFO, RX FIFO, RX timeout, modem status, and error conditions
- Standard FIFO-level and end-of-transmission interrupts
- Efficient transfers using micro direct memory access controller (µDMA):
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request is asserted when there is space in the FIFO; burst request is asserted at programmed FIFO level
- Programmable hardware flow control

22.2 Block Diagram

Figure 22-1 shows the UART module block diagram.

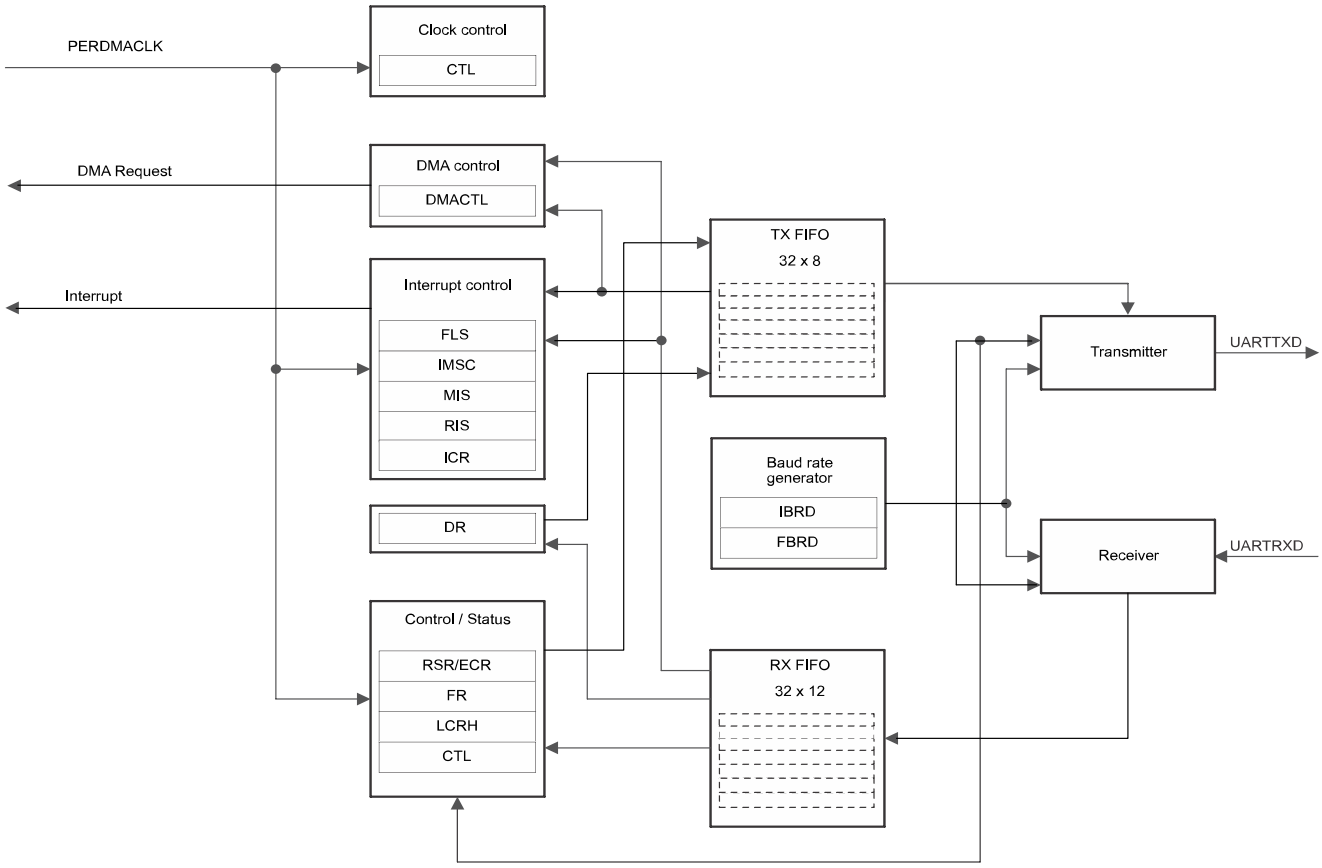


Figure 22-1. UART Module Block Diagram

22.3 Signal Description

Table 22-1 lists the external signals of the UART module and describes the function of each signal. The UART signals are set in the IOCFGn registers. For more information on configuring GPIOs, see Chapter 15.

Table 22-1. Signals for UART

Pin Name	Pin Number	Pin Type(1)	Description
UARTRxD	Assigned through GPIO configuration	I	UART module 0 receive
UARTTxD		O	UART module 0 transmit

(1) I = Input; O = Output; I/O = Bidirectional

22.4 Functional Description

Each CC13x4x10 and CC26x4x10 UART performs the functions of parallel-to-serial and serial-to-parallel conversions. The CC13x4x10 and CC26x4x10 UART is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and receive through the TXE and RXE bits of the UART Control Register (UART:CTL). Transmit and receive are both enabled out of reset. Before any control registers are programmed,

the UART must be disabled by clearing the UART:CTL.UARTEN register bit. If the UART is disabled during a transmit or receive operation, the current transaction completes before the UART stops.

22.4.1 Transmit and Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the TX FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits, according to the programmed configuration in the control registers. For details, see Figure 22-2.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse is detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data written to the RX FIFO.

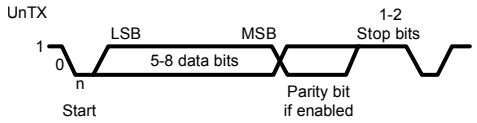


Figure 22-2. UART Character Frame

22.4.2 Baud Rate Generation

The baud rate divisor (BRD) is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud rate generator to determine the bit period. Having a fractional baudrate divider allows the UART to generate all standard baud rates.

The 16-bit integer is loaded through the UART Integer Baud Rate Divisor register (UART:IBRD), and the 6-bit fractional part is loaded with the UART Fractional Baud Rate Divisor register (UART:FBRD). Equation 4 shows the relationship of the BRD and the system clock.

$$BRD = BRDI + BRDF = PERDMACLK / (ClkDiv \times Baud\ Rate) \quad (4)$$

where:

- BRDI is the integer part of the BRD
- BRDF is the fractional part, separated by a decimal place
- PERDMACLK is the system clock connected to the UART
- ClkDiv is 16

The 6-bit fractional number that is loaded into the UART:FBRD.DIVFRAC bit field can be calculated by taking the fractional part of the baud rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors, as shown by Equation 5.

$$UART:FBRD.DIVFRAC = integer\ (BRDF \times 64 + 0.5) \quad (5)$$

Along with the UART Line Control, High Byte register (UART:LCRH), the UART:IBRD and the UART:FBRD registers form an internal 30-bit register. This internal register is updated only when a write operation to the UART:LCRH register is performed, so a write to the UART:LCRH register must follow any changes to the BRD for the changes to take effect.

The four possible sequences to update the baud rate registers are as follows:

- UART:IBRD write, UART:FBRD write, and UART:LCRH write
- UART:FBRD write, UART;IBRD write, and UART:LCRH write
- UART:IBRD write and UART:LCRH write
- UART:FBRD write and UART:LCRH write

22.4.3 Data Transmission

Data received is stored in the RX FIFO and has an extra four bits per character for status information. Data is written into the TX FIFO for transmission. If the UART is enabled, a data frame starts transmitting with the parameters indicated in the UART:LCRH register. Data continues to transmit until no data is left in the TX FIFO.

The UART:FR.BUSY bit is asserted as soon as the TXFIFO has data available to be transmitted (that is, the TXFIFO was empty, and data was written to it). The bit will remain asserted while data is being transmitted, and will not de-assert until the TX FIFO is empty, and the last character has transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the UARTRXD signal is continuously 1), and the data input goes low (a start bit was received), the receive counter begins running and data is sampled.

The start bit is valid and recognized if the UARTRXD signal is still low on the eighth cycle of the baud rate clock otherwise the start bit is ignored. After a valid start bit is detected, successive data bits are sampled on every sixteenth cycle of the baud rate clock. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the UART:LCRH register.

A valid stop bit is confirmed if the UARTRXD signal is high; otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO with any error bits associated with that word.

22.4.4 Modem Handshake Support

This section describes how to configure and use the modem flow control signals for UART0 when connected as a data terminal equipment (DTE), or as a data communications equipment (DCE). A modem is a DCE, and a computing device that connects to a modem is the DTE.

22.4.4.1 Signaling

The status signals provided by UART0 differ based on whether the UART is used as a DTE or a DCE. When used as a DTE, the modem flow control signals are defined as:

- **UART0CTS** is Clear To Send
- **UART0RTS** is Request To Send

When used as a DCE, the modem flow control signals are defined as:

- **UART0CTS** is Request To Send
- **UART0RTS** is Clear To Send

22.4.4.2 Flow Control

Either hardware or software can accomplish flow control. The following sections describe the different methods.

22.4.4.2.1 Hardware Flow Control (RTS and CTS)

Hardware flow control between two devices is accomplished by connecting the UART0RTS output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the UART0CTS input.

The UART0CTS input controls the transmitter. The transmitter can transmit data only when the UART0CTS input is asserted. The UART0RTS output signal indicates the state of the receive FIFO. UART0CTS remains asserted until the preprogrammed trigger level is reached, indicating that the RX FIFO has no space to store additional characters.

The UART:CTL.CTSEN and UART:CTL.RTSEN specify the flow control mode as shown in Table 22-2.

Table 22-2. Flow Control Mode

CTSEN	RTSEN	Description
1	1	RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

22.4.4.2.2 Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts can be generated for the U1CTS signal using bit 3 of the UART:IMSC register. The raw and masked interrupt status can be checked using the UART:RIS and UART:MIS registers. These interrupts can be cleared using the UART:ICR register.

22.4.5 FIFO Operation

The UART has two 32-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed through the UART Data Register (UART:DR). Read operations of the UART:DR register return a 12-bit value consisting of 8 data bits and 4 error flags, while write operations place 8-bit data in the TX FIFO.

Out of reset, both FIFOs are disabled and act as 1-byte-deep holding registers. The FIFOs are enabled by setting the UART:LCRH FEN register bit.

FIFO status can be monitored through the UART Flag Register (UART:FR) and the UART Receive Status Register (UART:RSR). Hardware monitors empty, full, and overrun conditions. The UART:FR register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the UART:RSR register shows overrun status through the OE bit. If the FIFOs are disabled, the empty and full flags are set according to the status of the 1-byte deep holding registers.

The trigger points at which the FIFOs generate interrupts are controlled through the UART Interrupt FIFO Level Select Register (UART:IFLS). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include ⅛, ¼, ½, ¾, and ⅞. For example, if the ¼ option is selected for the receive FIFO, the UART generates a receive interrupt after 8 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the ½ mark.

22.4.6 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun error
- Break error
- Parity error
- Framing error
- Receive time-out
- Transmit (when the condition defined in the UART:IFLS.TXSEL register bit is met)
- Receive (when the condition defined in the UART:IFLS.RXSEL register bit is met)
- End of transmission (when no data on TX line and TX FIFO underflow)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine (ISR) by reading the UART Masked Interrupt Status Register (UART:MIS).

The interrupt events that can trigger a controller-level interrupt are defined in the UART Interrupt Mask Register (UART:IMSC) by setting the corresponding bits. If interrupts are not used, the raw interrupt status is always visible through the UART Raw Interrupt Status Register (UART:RIS).

Interrupts can be cleared (for the UART:MIS and UART:RIS registers) by setting the corresponding bit in the UART Interrupt Clear Register (UART:ICR).

The receive time-out interrupt is asserted when the RX FIFO is not empty, and no further data is received over a 32-bit period. The receive time-out interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when the corresponding bit in the UART:ICR register is set.

The UART module provides the possibility of setting and clearing masks for every individual interrupt source using the UART Interrupt Mask Set/Clear Register (UART:IMSC). The five events that can cause combined interrupts to CPU are:

- RX: The receive interrupt changes state when one of the following events occurs:
  - If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level. When this happens, the receive interrupt is asserted high. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt.
  - If the FIFOs are disabled (have a depth of one location) and data is received, thereby filling the location, the receive interrupt is asserted high. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt.
- TX: The transmit interrupt changes state when one of the following events occurs:
  - If the FIFOs are enabled and the transmit FIFO is equal to or lower than the programmed trigger level, then the transmit interrupt is asserted high. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt.
  - If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the transmit interrupt is asserted high. The interrupt is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt.
- RX time-out: The receive time-out interrupt is asserted when the receive FIFO is not empty, and no more data is received during a 32-bit period. The receive time-out interrupt is cleared either when the FIFO becomes empty through reading all the data or by reading the holding register, or when 1 is written to the corresponding bit of the Interrupt Clear Register (UART:ICR).
- Modem status: The modem status interrupt is asserted if the modem status signal UART0CTS changes. It can be cleared using the corresponding clear bit in the UART:ICR register.
- Error: The error interrupt is asserted when an error occurs in the reception of data by the UART. The interrupt can be caused by a number of different error conditions:
  - Framing
  - Parity
  - Break
  - Overrun

  The cause of the interrupt can be determined by reading the UART:RIS register or the UART:MIS register. The interrupt can be cleared by writing to the relevant bits of the UART:ICR register.

In addition to the five events produced by the UART module, two additional events are ORed to the interrupt line:

- RX DMA done: Indicates that the receiver DMA has completed its task. This is a level interrupt provided by the DMA module, and is cleared using the dma_done clear register (UDMA:REQDONE) in DMA module.
- TX DMA done: Indicates that the transmit DMA has completed its task. This is a level interrupt provided by the DMA module, and is cleared using the dma_done clear register (UDMA:REQDONE) in DMA module.

22.4.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the UART:CTL.LBE register bit. In loopback mode, data transmitted on the UARTTXD output is received on the UARTRXD input. The LBE bit must be set before the UART is enabled.

22.5 Interface to µDMA

The CC13x4x10 and CC26x4x10 devices provide an interface to connect to a µDMA controller. Figure 22-3 shows the interface between the µDMA and UART. This interface contains four DMA requests as outputs (RX Single, RX Burst, TX Single, and TX Burst). The DMA interface also has two DMA request clears as inputs (for clearing TX and RX DMA requests). Each DMA request signal remains asserted until the relevant DMA clear signal is asserted.

The burst transfer and single transfer request signals are not mutually exclusive, and both can be asserted at the same time. For example, when there is more data than the trigger level in the receive FIFO, the burst transfer request and the single transfer request are asserted.

The single and burst requests cannot be masked separately by the UART module and if corresponding DMA (RX or TX) is enabled, both of these requests are sent to the DMA. The DMA configuration selects either single or burst request as the trigger. All request signals are deasserted if the UART is disabled or if the relevant DMA enable bit (TXDMAE or RXDMAE) in the DMA Control Register (UART:DMACTL) is cleared.

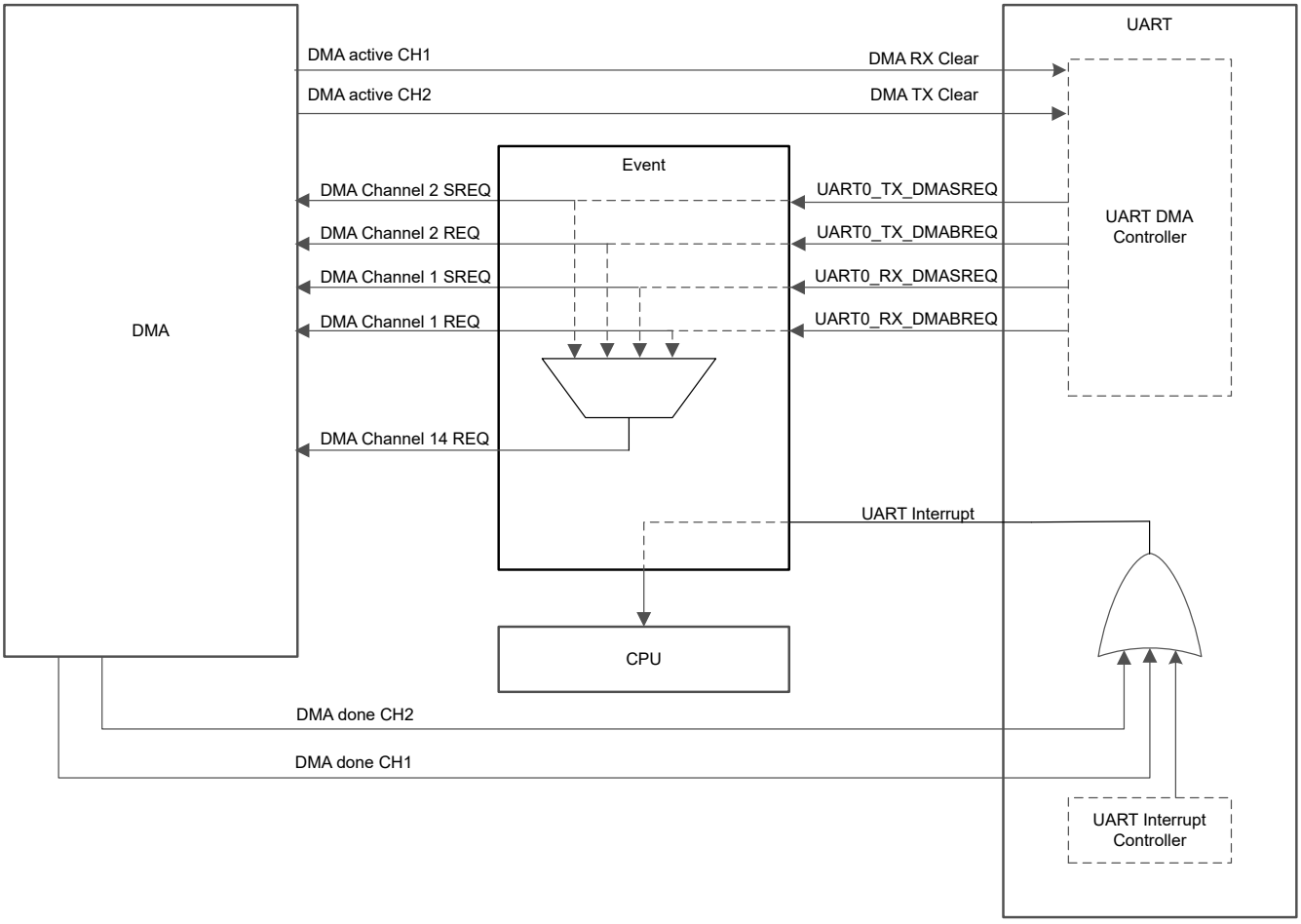


Figure 22-3. µDMA Example

22.6 Initialization and Configuration

The UART module provides four I/O signals to be routed to the DIOs. The following signals are selected through the IOCFGn registers in the IOC module.

- Inputs: RXD, CTS
- Outputs: TXD, RTS

CTS and RTS lines are active low.

Note

IOC must be configured before enabling the UART to avoid unwanted transitions on the input being processed as UART signals. When IOC is configured as UART-specific I/Os (RXD, CTS, TXD, or RTS), IOC sets static output driver enable to the DIO (output driver enable = 1 for output TXD and RTS and output driver enable = 0 for inputs RXD and CTS).

The following describes the necessary steps to enable and initialize the UART.

TI recommends using the UART2 driver in the SimpleLink™ CC13xx and CC26xx software development kit (SDK) when using the UART.

1. Enable the serial power domain and enable the UART module in the PRCM module by writing to the PRCM:UARTCLKGR register, the PRCM:UARTCLKGS register, and the PRCM:UARTCLKGDS register, and loading the setting to the clock controller by writing to the PRCM:CLKLOADCTL register
2. Configure the IOC module to map UART signals to the correct GPIO pins. For more information on pin connections, see Chapter 15.

This section discusses the steps required to use a UART module. For this example, the UART clock is assumed to be 24 MHz, and the desired UART configuration is the following:

- Baud rate: 115200
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the BRD because the UART:IBRD and UART:FBRD registers must be written before the UART:LCRH register. The BRD can be calculated using the equation described in Section 22.4.2.

$$BRD = 24\,000\,000 / (16 \times 115\,200) = 13.0208 \quad (6)$$

The result of Equation 6 indicates that the UART:IBRD.DIVINT field must be set to 13 decimal or 0xD. Equation 7 calculates the value to be loaded into the UART:FBRD register.

$$UART:FBRD.DIVFRAC = integer\ (0.0208 \times 64 + 0.5) = 1 \quad (7)$$

With the BRD values available, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UART:CTL.UARTEN bit.
2. Write the integer portion of the BRD to the UART:IBRD register.
3. Write the fractional portion of the BRD to the UART:FBRD register.
4. Write the desired serial parameters to the UART:LCRH register (in this case, a value of 0x00000060).
5. Enable the UART by setting the UART:CTL.UARTEN bit.

22.7 UART Registers

Table 22-3 lists the memory-mapped registers for the UART registers. All register offset addresses not listed in Table 22-3 should be considered as reserved locations and the register contents should not be modified.

Table 22-3. UART Registers

Offset	Acronym	Register Name	Section
0h	DR	Data	Section 22.7.1
4h	RSR	Status	Section 22.7.2
4h	ECR	Error Clear	Section 22.7.3
18h	FR	Flag	Section 22.7.4
24h	IBRD	Integer Baud-Rate Divisor	Section 22.7.5
28h	FBRD	Fractional Baud-Rate Divisor	Section 22.7.6
2Ch	LCRH	Line Control	Section 22.7.7
30h	CTL	Control	Section 22.7.8
34h	IFLS	Interrupt FIFO Level Select	Section 22.7.9
38h	IMSC	Interrupt Mask Set/Clear	Section 22.7.10
3Ch	RIS	Raw Interrupt Status	Section 22.7.11
40h	MIS	Masked Interrupt Status	Section 22.7.12
44h	ICR	Interrupt Clear	Section 22.7.13
48h	DMACTL	DMA Control	Section 22.7.14

Complex bit access types are encoded to fit into small table cells. Table 22-4 shows the codes that are used for access types in this section.

Table 22-4. UART Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

22.7.1 DR Register (Offset = 0h) [Reset = 00000000h]

DR is shown in Table 22-5.

Return to the Summary Table.

Data
For words to be transmitted:
- if the FIFOs are enabled (LCRH.FEN = 1), data written to this location is pushed onto the transmit FIFO
- if the FIFOs are not enabled (LCRH.FEN = 0), data is stored in the transmitter holding register (the bottom word of the transmit FIFO).
The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit.
The resultant word is then transmitted.
For received words:
- if the FIFOs are enabled (LCRH.FEN = 1), the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO
- if the FIFOs are not enabled (LCRH.FEN = 0), the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).
The received data byte is read by performing reads from this register along with the corresponding status information. The status information can also be read by a read of the RSR register.

Table 22-5. DR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	OE	R	X	UART Overrun Error: This bit is set to 1 if data is received and the receive FIFO is already full. The FIFO contents remain valid because no more data is written when the FIFO is full, , only the contents of the shift register are overwritten. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.
10	BE	R	X	UART Break Error: This bit is set to 1 if a break condition was detected, indicating that the received data input (UARTRXD input pin) was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO (that is., the oldest received data character since last read). When a break occurs, a 0 character is loaded into the FIFO. The next character is enabled after the receive data input (UARTRXD input pin) goes to a 1 (marking state), and the next valid start bit is received.
9	PE	R	X	UART Parity Error: When set to 1, it indicates that the parity of the received data character does not match the parity that the LCRH.EPS and LCRH.SPS select. In FIFO mode, this error is associated with the character at the top of the FIFO (that is, the oldest received data character since last read).
8	FE	R	X	UART Framing Error: When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO (that is., the oldest received data character since last read).
7-0	DATA	R/W	X	Data transmitted or received: On writes, the transmit data character is pushed into the FIFO. On reads, the oldest received data character since the last read is returned.

22.7.2 RSR Register (Offset = 4h) [Reset = 00000000h]

RSR is shown in Table 22-6.

Return to the Summary Table.

Status
This register is mapped to the same address as ECR register. Reads from this address are associated with RSR register and return the receive status. Writes to this address are associated with ECR register and clear the receive status flags (framing, parity, break, and overrun errors).
If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from the Data Register, DR prior to reading the RSR. The status information for overrun is set immediately when an overrun condition occurs.

Table 22-6. RSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	OE	R	0h	UART Overrun Error: This bit is set to 1 if data is received and the receive FIFO is already full. The FIFO contents remain valid because no more data is written when the FIFO is full, , only the contents of the shift register are overwritten. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.
2	BE	R	0h	UART Break Error: This bit is set to 1 if a break condition was detected, indicating that the received data input (UARTRXD input pin) was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). When a break occurs, a 0 character is loaded into the FIFO. The next character is enabled after the receive data input (UARTRXD input pin) goes to a 1 (marking state), and the next valid start bit is received.
1	PE	R	0h	UART Parity Error: When set to 1, it indicates that the parity of the received data character does not match the parity that the LCRH.EPS and LCRH.SPS select.
0	FE	R	0h	UART Framing Error: When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1).

22.7.3 ECR Register (Offset = 4h) [Reset = 00000000h]

ECR is shown in Table 22-7.

Return to the Summary Table.

Error Clear
This register is mapped to the same address as RSR register. Reads from this address are associated with RSR register and return the receive status. Writes to this address are associated with ECR register and clear the receive status flags (framing, parity, break, and overrun errors).

Table 22-7. ECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	OE	W	0h	The framing (FE), parity (PE), break (BE) and overrun (OE) errors are cleared to 0 by any write to this register.
2	BE	W	0h	The framing (FE), parity (PE), break (BE) and overrun (OE) errors are cleared to 0 by any write to this register.
1	PE	W	0h	The framing (FE), parity (PE), break (BE) and overrun (OE) errors are cleared to 0 by any write to this register.
0	FE	W	0h	The framing (FE), parity (PE), break (BE) and overrun (OE) errors are cleared to 0 by any write to this register.

22.7.4 FR Register (Offset = 18h) [Reset = 00000090h]

FR is shown in Table 22-8.

Return to the Summary Table.

Flag
Reads from this register return the UART flags.

Table 22-8. FR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	TXFE	R	1h	UART Transmit FIFO Empty: The meaning of this bit depends on the state of LCRH.FEN . - If the FIFO is disabled, this bit is set when the transmit holding register is empty. - If the FIFO is enabled, this bit is set when the transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.
6	RXFF	R	0h	UART Receive FIFO Full: The meaning of this bit depends on the state of LCRH.FEN. - If the FIFO is disabled, this bit is set when the receive holding register is full. - If the FIFO is enabled, this bit is set when the receive FIFO is full.
5	TXFF	R	0h	UART Transmit FIFO Full: Transmit FIFO full. The meaning of this bit depends on the state of LCRH.FEN. - If the FIFO is disabled, this bit is set when the transmit holding register is full. - If the FIFO is enabled, this bit is set when the transmit FIFO is full.
4	RXFE	R	1h	UART Receive FIFO Empty: Receive FIFO empty. The meaning of this bit depends on the state of LCRH.FEN. - If the FIFO is disabled, this bit is set when the receive holding register is empty. - If the FIFO is enabled, this bit is set when the receive FIFO is empty.
3	BUSY	R	0h	UART Busy: If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.
2-1	RESERVED	R	0h	Reserved
0	CTS	R	X	Clear To Send: This bit is the complement of the active-low UART CTS input pin. That is, the bit is 1 when CTS input pin is LOW.

22.7.5 IBRD Register (Offset = 24h) [Reset = 00000000h]

IBRD is shown in Table 22-9.

Return to the Summary Table.

Integer Baud-Rate Divisor
If this register is modified while transmission or reception is on-going, the baud rate will not be updated until transmission or reception of the current character is complete.

Table 22-9. IBRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DIVINT	R/W	0h	The integer baud rate divisor: The baud rate divisor is calculated using the formula below: Baud rate divisor = (UART reference clock frequency) / (16 * Baud rate) Baud rate divisor must be minimum 1 and maximum 65535. That is, DIVINT=0 does not give a valid baud rate. Similarly, if DIVINT=0xFFFF, any non-zero values in FBRD.DIVFRAC will be illegal. A valid value must be written to this field before the UART can be used for RX or TX operations.

22.7.6 FBRD Register (Offset = 28h) [Reset = 00000000h]

FBRD is shown in Table 22-10.

Return to the Summary Table.

Fractional Baud-Rate Divisor
If this register is modified while trasmission or reception is on-going, the baudrate will not be updated until transmission or reception of the current character is complete.

Table 22-10. FBRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-0	DIVFRAC	R/W	0h	Fractional Baud-Rate Divisor: The baud rate divisor is calculated using the formula below: Baud rate divisor = (UART reference clock frequency) / (16 * Baud rate) Baud rate divisor must be minimum 1 and maximum 65535. That is, IBRD.DIVINT=0 does not give a valid baud rate. Similarly, if IBRD.DIVINT=0xFFFF, any non-zero values in DIVFRAC will be illegal. A valid value must be written to this field before the UART can be used for RX or TX operations.

22.7.7 LCRH Register (Offset = 2Ch) [Reset = 00000000h]

LCRH is shown in Table 22-11.

Return to the Summary Table.

Line Control

Table 22-11. LCRH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	SPS	R/W	0h	UART Stick Parity Select: 0: Stick parity is disabled 1: The parity bit is transmitted and checked as invert of EPS field (i.e. the parity bit is transmitted and checked as 1 when EPS = 0). This bit has no effect when PEN disables parity checking and generation.
6-5	WLEN	R/W	0h	UART Word Length: These bits indicate the number of data bits transmitted or received in a frame. 0h = 5 : Word Length 5 bits 1h = 6 : Word Length 6 bits 2h = 7 : Word Length 7 bits 3h = 8 : Word Length 8 bits
4	FEN	R/W	0h	UART Enable FIFOs 0h = FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers. 1h = Transmit and receive FIFO buffers are enabled (FIFO mode)
3	STP2	R/W	0h	UART Two Stop Bits Select: If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2	EPS	R/W	0h	UART Even Parity Select 0h = Odd parity: The UART generates or checks for an odd number of 1s in the data and parity bits. 1h = Even parity: The UART generates or checks for an even number of 1s in the data and parity bits.
1	PEN	R/W	0h	UART Parity Enable This bit controls generation and checking of parity bit. 0h = Parity is disabled and no parity bit is added to the data frame 1h = Parity checking and generation is enabled.
0	BRK	R/W	0h	UART Send Break If this bit is set to 1, a low-level is continually output on the UARTTXD output pin, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

22.7.8 CTL Register (Offset = 30h) [Reset = 00000300h]

CTL is shown in Table 22-12.

Return to the Summary Table.

Control

Table 22-12. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	CTSEN	R/W	0h	CTS hardware flow control enable 0h = CTS hardware flow control disabled 1h = CTS hardware flow control enabled
14	RTSEN	R/W	0h	RTS hardware flow control enable 0h = RTS hardware flow control disabled 1h = RTS hardware flow control enabled
13-12	RESERVED	R	0h	Reserved
11	RTS	R/W	0h	Request to Send This bit is the complement of the active-low UART RTS output. That is, when the bit is programmed to a 1 then RTS output on the pins is LOW.
10	RESERVED	R	0h	Reserved
9	RXE	R/W	1h	UART Receive Enable If the UART is disabled in the middle of reception, it completes the current character before stopping. 0h = UART Receive disabled 1h = UART Receive enabled
8	TXE	R/W	1h	UART Transmit Enable If the UART is disabled in the middle of transmission, it completes the current character before stopping. 0h = UART Transmit disabled 1h = UART Transmit enabled
7	LBE	R/W	0h	UART Loop Back Enable: Enabling the loop-back mode connects the UARTTXD output from the UART to UARTRXD input of the UART. 0h = Loop Back disabled 1h = Loop Back enabled
6-1	RESERVED	R	0h	Reserved
0	UARTEN	R/W	0h	UART Enable 0h = UART disabled 1h = UART enabled

22.7.9 IFLS Register (Offset = 34h) [Reset = 00000012h]

IFLS is shown in Table 22-13.

Return to the Summary Table.

Interrupt FIFO Level Select

Table 22-13. IFLS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-3	RXSEL	R/W	2h	Receive interrupt FIFO level select: This field sets the trigger points for the receive interrupt. Values 0b101-0b111 are reserved. 0h = 1_8 : Receive FIFO becomes >= 1/8 full 1h = 2_8 : Receive FIFO becomes >= 1/4 full 2h = 4_8 : Receive FIFO becomes >= 1/2 full 3h = 6_8 : Receive FIFO becomes >= 3/4 full 4h = 7_8 : Receive FIFO becomes >= 7/8 full
2-0	TXSEL	R/W	2h	Transmit interrupt FIFO level select: This field sets the trigger points for the transmit interrupt. Values 0b101-0b111 are reserved. 0h = 1_8 : Transmit FIFO becomes <= 1/8 full 1h = 2_8 : Transmit FIFO becomes <= 1/4 full 2h = 4_8 : Transmit FIFO becomes <= 1/2 full 3h = 6_8 : Transmit FIFO becomes <= 3/4 full 4h = 7_8 : Transmit FIFO becomes <= 7/8 full

22.7.10 IMSC Register (Offset = 38h) [Reset = 00000000h]

IMSC is shown in Table 22-14.

Return to the Summary Table.

Interrupt Mask Set/Clear

Table 22-14. IMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	EOTIM	R/W	0h	End of Transmission interrupt mask. A read returns the current mask for UART's EoT interrupt. On a write of 1, the mask of the EoT interrupt is set which means the interrupt state will be reflected in MIS.EOTMIS. A write of 0 clears the mask which means MIS.EOTMIS will not reflect the interrupt.
10	OEIM	R/W	0h	Overrun error interrupt mask. A read returns the current mask for UART's overrun error interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.OEMIS. A write of 0 clears the mask which means MIS.OEMIS will not reflect the interrupt.
9	BEIM	R/W	0h	Break error interrupt mask. A read returns the current mask for UART's break error interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.BEMIS. A write of 0 clears the mask which means MIS.BEMIS will not reflect the interrupt.
8	PEIM	R/W	0h	Parity error interrupt mask. A read returns the current mask for UART's parity error interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.PEMIS. A write of 0 clears the mask which means MIS.PEMIS will not reflect the interrupt.
7	FEIM	R/W	0h	Framing error interrupt mask. A read returns the current mask for UART's framing error interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.FEMIS. A write of 0 clears the mask which means MIS.FEMIS will not reflect the interrupt.
6	RTIM	R/W	0h	Receive timeout interrupt mask. A read returns the current mask for UART's receive timeout interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.RTMIS. A write of 0 clears the mask which means this bitfield will not reflect the interrupt. The raw interrupt for receive timeout RIS.RTRIS cannot be set unless the mask is set (RTIM = 1). This is because the mask acts as an enable for power saving. That is, the same status can be read from MIS.RTMIS and RIS.RTRIS.
5	TXIM	R/W	0h	Transmit interrupt mask. A read returns the current mask for UART's transmit interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.TXMIS. A write of 0 clears the mask which means MIS.TXMIS will not reflect the interrupt.
4	RXIM	R/W	0h	Receive interrupt mask. A read returns the current mask for UART's receive interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.RXMIS. A write of 0 clears the mask which means MIS.RXMIS will not reflect the interrupt.
3-2	RESERVED	R	0h	Reserved
1	CTSMIM	R/W	0h	Clear to Send (CTS) modem interrupt mask. A read returns the current mask for UART's clear to send interrupt. On a write of 1, the mask of the overrun error interrupt is set which means the interrupt state will be reflected in MIS.CTSMMIS. A write of 0 clears the mask which means MIS.CTSMMIS will not reflect the interrupt.
0	RESERVED	R	0h	Reserved

22.7.11 RIS Register (Offset = 3Ch) [Reset = 0000000Dh]

RIS is shown in Table 22-15.

Return to the Summary Table.

Raw Interrupt Status

Table 22-15. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	EOTRIS	R	0h	End of Transmission interrupt status: This field returns the raw interrupt state of UART's end of transmission interrupt. End of transmission flag is set when all the Transmit data in the FIFO and on the TX Line is tranmitted.
10	OERIS	R	0h	Overrun error interrupt status: This field returns the raw interrupt state of UART's overrun error interrupt. Overrun error occurs if data is received and the receive FIFO is full.
9	BERIS	R	0h	Break error interrupt status: This field returns the raw interrupt state of UART's break error interrupt. Break error is set when a break condition is detected, indicating that the received data input (UARTRXD input pin) was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits).
8	PERIS	R	0h	Parity error interrupt status: This field returns the raw interrupt state of UART's parity error interrupt. Parity error is set if the parity of the received data character does not match the parity that the LCRH.EPS and LCRH.SPS select.
7	FERIS	R	0h	Framing error interrupt status: This field returns the raw interrupt state of UART's framing error interrupt. Framing error is set if the received character does not have a valid stop bit (a valid stop bit is 1).
6	RTRIS	R	0h	Receive timeout interrupt status: This field returns the raw interrupt state of UART's receive timeout interrupt. The receive timeout interrupt is asserted when the receive FIFO is not empty, and no more data is received during a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data, or when a 1 is written to ICR.RTIC. The raw interrupt for receive timeout cannot be set unless the mask is set (IMSC.RTIM = 1). This is because the mask acts as an enable for power saving. That is, the same status can be read from MIS.RTMIS and RTRIS.
5	TXRIS	R	0h	Transmit interrupt status: This field returns the raw interrupt state of UART's transmit interrupt. When FIFOs are enabled (LCRH.FEN = 1), the transmit interrupt is asserted if the number of bytes in transmit FIFO is equal to or lower than the programmed trigger level (IFLS.TXSEL). The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt through ICR.TXIC. When FIFOs are disabled (LCRH.FEN = 0), that is they have a depth of one location, the transmit interrupt is asserted if there is no data present in the transmitters single location. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt through ICR.TXIC.

Table 22-15. RIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	RXRIS	R	0h	Receive interrupt status: This field returns the raw interrupt state of UART's receive interrupt. When FIFOs are enabled (LCRH.FEN = 1), the receive interrupt is asserted if the receive FIFO reaches the programmed trigger level (IFLS.RXSEL). The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt through ICR.RXIC. When FIFOs are disabled (LCRH.FEN = 0), that is they have a depth of one location, the receive interrupt is asserted if data is received thereby filling the location. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt through ICR.RXIC.
3-2	RESERVED	R	0h	Reserved
1	CTSRMIS	R	X	Clear to Send (CTS) modem interrupt status: This field returns the raw interrupt state of UART's clear to send interrupt.
0	RESERVED	R	0h	Reserved

22.7.12 MIS Register (Offset = 40h) [Reset = 00000000h]

MIS is shown in Table 22-16.

Return to the Summary Table.

Masked Interrupt Status

Table 22-16. MIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	EOTMIS	R	0h	End of Transmission interrupt status: This field returns the masked interrupt state of the overrun interrupt which is the AND product of raw interrupt state RIS.EOTRIS and the mask setting IMSC.EOTIM.
10	OEMIS	R	0h	Overrun error masked interrupt status: This field returns the masked interrupt state of the overrun interrupt which is the AND product of raw interrupt state RIS.OERIS and the mask setting IMSC.OEIM.
9	BEMIS	R	0h	Break error masked interrupt status: This field returns the masked interrupt state of the break error interrupt which is the AND product of raw interrupt state RIS.BERIS and the mask setting IMSC.BEIM.
8	PEMIS	R	0h	Parity error masked interrupt status: This field returns the masked interrupt state of the parity error interrupt which is the AND product of raw interrupt state RIS.PERIS and the mask setting IMSC.PEIM.
7	FEMIS	R	0h	Framing error masked interrupt status: Returns the masked interrupt state of the framing error interrupt which is the AND product of raw interrupt state RIS.FERIS and the mask setting IMSC.FEIM.
6	RTMIS	R	0h	Receive timeout masked interrupt status: Returns the masked interrupt state of the receive timeout interrupt. The raw interrupt for receive timeout cannot be set unless the mask is set (IMSC.RTIM = 1). This is because the mask acts as an enable for power saving. That is, the same status can be read from RTMIS and RIS.RTRIS.
5	TXMIS	R	0h	Transmit masked interrupt status: This field returns the masked interrupt state of the transmit interrupt which is the AND product of raw interrupt state RIS.TXRIS and the mask setting IMSC.TXIM.
4	RXMIS	R	0h	Receive masked interrupt status: This field returns the masked interrupt state of the receive interrupt which is the AND product of raw interrupt state RIS.RXRIS and the mask setting IMSC.RXIM.
3-2	RESERVED	R	0h	Reserved
1	CTSMMIS	R	0h	Clear to Send (CTS) modem masked interrupt status: This field returns the masked interrupt state of the clear to send interrupt which is the AND product of raw interrupt state RIS.CTSRMIS and the mask setting IMSC.CTSMIM.
0	RESERVED	R	0h	Reserved

22.7.13 ICR Register (Offset = 44h) [Reset = 00000000h]

ICR is shown in Table 22-17.

Return to the Summary Table.

Interrupt Clear
On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

Table 22-17. ICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reserved
11	EOTIC	W	X	End of Transmission interrupt clear: Writing 1 to this field clears the overrun error interrupt (RIS.EOTRIS). Writing 0 has no effect.
10	OEIC	W	X	Overrun error interrupt clear: Writing 1 to this field clears the overrun error interrupt (RIS.OERIS). Writing 0 has no effect.
9	BEIC	W	X	Break error interrupt clear: Writing 1 to this field clears the break error interrupt (RIS.BERIS). Writing 0 has no effect.
8	PEIC	W	X	Parity error interrupt clear: Writing 1 to this field clears the parity error interrupt (RIS.PERIS). Writing 0 has no effect.
7	FEIC	W	X	Framing error interrupt clear: Writing 1 to this field clears the framing error interrupt (RIS.FERIS). Writing 0 has no effect.
6	RTIC	W	X	Receive timeout interrupt clear: Writing 1 to this field clears the receive timeout interrupt (RIS.RTRIS). Writing 0 has no effect.
5	TXIC	W	X	Transmit interrupt clear: Writing 1 to this field clears the transmit interrupt (RIS.TXRIS). Writing 0 has no effect.
4	RXIC	W	X	Receive interrupt clear: Writing 1 to this field clears the receive interrupt (RIS.RXRIS). Writing 0 has no effect.
3-2	RESERVED	W	X	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior. Write 0
1	CTSMIC	W	X	Clear to Send (CTS) modem interrupt clear: Writing 1 to this field clears the clear to send interrupt (RIS.CTSRMIS). Writing 0 has no effect.
0	RESERVED	W	X	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior. Write 0.

22.7.14 DMACTL Register (Offset = 48h) [Reset = 00000000h]

DMACTL is shown in Table 22-18.

Return to the Summary Table.

DMA Control

Table 22-18. DMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	DMAONERR	R/W	0h	DMA on error. If this bit is set to 1, the DMA receive request outputs (for single and burst requests) are disabled when the UART error interrupt is asserted (more specifically if any of the error interrupts RIS.PERIS, RIS.BERIS, RIS.FERIS or RIS.OERIS are asserted).
1	TXDMAE	R/W	0h	Transmit DMA enable. If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0h	Receive DMA enable. If this bit is set to 1, DMA for the receive FIFO is enabled.

This page intentionally left blank.

Chapter 23

Serial Peripheral Interface (SPI)

This chapter describes the Serial Peripheral Interface (SPI).

23.1 Introduction 1870
23.2 Block Diagram 1871
23.3 Signal Description 1872
23.4 Functional Description 1872
23.5 µDMA Operation 1882
23.6 Initialization and Configuration 1882
23.7 SPI Registers 1884

23.1 Introduction

The Serial Peripheral Interface (SPI) module provides a standardized serial interface to transfer data between the CC13x4x10 and CC26x4x10 family and other external devices using SPI protocol (such as sensors, memories, ADCs or DACs).

The four SPI modules of the CC13x4x10 and CC26x4x10 device platform have the following features:

- Programmable interface operation for Motorola SPI (3-wire and 4-wire), MICROWIRE, or TI Synchronous Serial format
- Configurable as a master or a slave on the interface
- Programmable clock bit rate and prescaler
- Separate transmit (TX) and receive (RX) first-in first-out buffers (FIFOs), each 32 bits wide and 8 locations deep
- Programmable data frame size from 4 bits to 32 bits (master mode) or 7 to 32 bits (slave mode)
- Internal loopback test mode for diagnostic and debug testing
- Interrupts for transmit and receive FIFOs, overrun and time-out interrupts, and DMA done interrupts
- Efficient transfers using micro direct memory access controller (µDMA):
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted when there are a configurable number of entries in the FIFO
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains a configurable amount of entries

23.2 Block Diagram

Figure 23-1 shows the SPI block diagram.

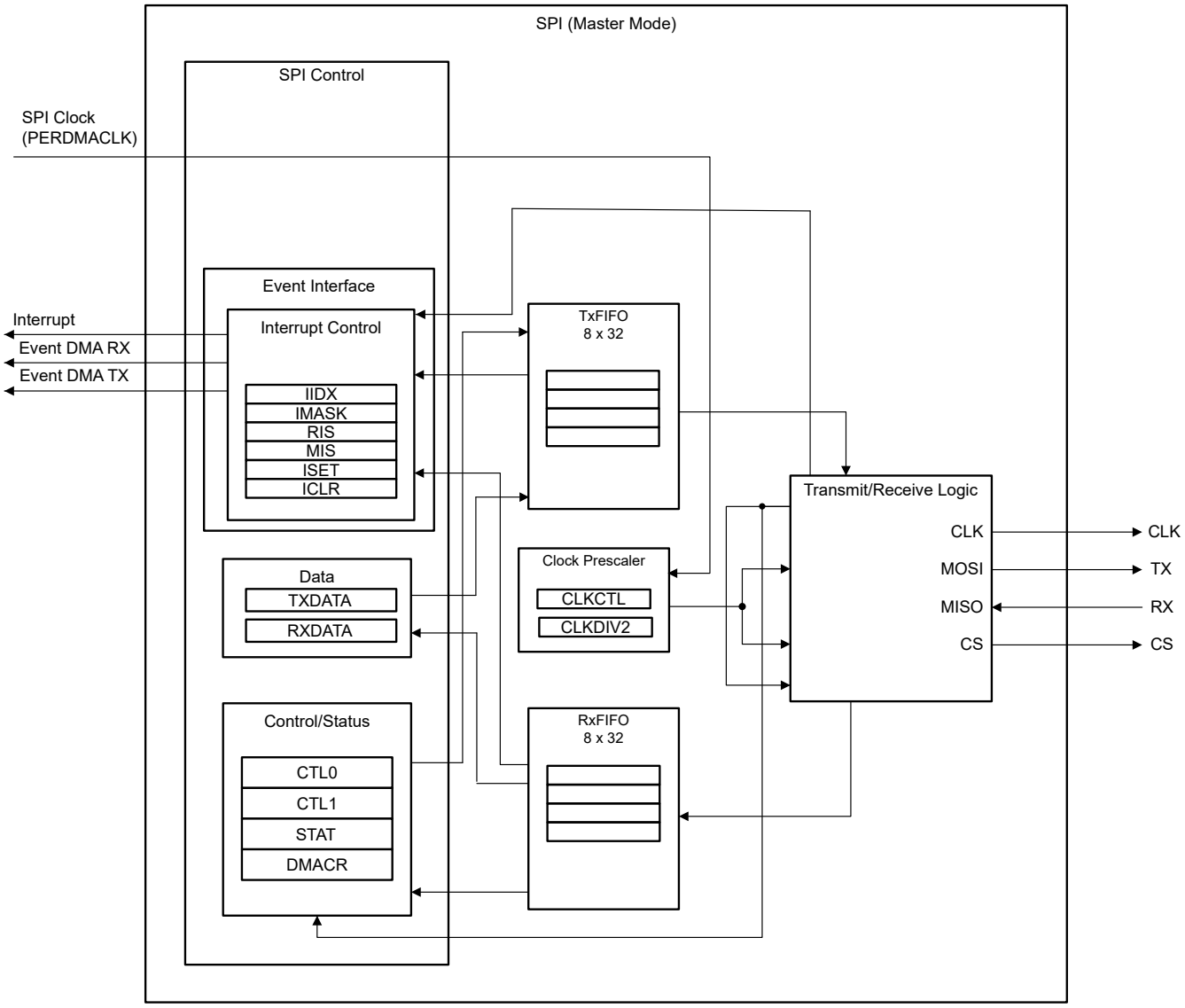


Figure 23-1. SPI Module Block Diagram (Master Mode)

23.3 Signal Description

Table 23-1 lists the external signals of one instance of the SPI and describes the function of each. The SPI signals for all four instances are selected in the IOC module through the IOCFGn registers. For more information on configuration of DIOs, see Chapter 15.

Table 23-1. SPI Signals

Signal Name	Pin Number	Pin Type(1)	Description
SPIn_CLK	Assigned in the I/O Controller	I/O	**SPI clock** Master mode: CLK is an output Slave mode: CLK is an input
SPIn_CS		I/O	**SPI chip select** Master mode: CS is an output Slave mode: CS is an input
SPIn_TX		O	**SPI TX Output** Master mode: TX drives MOSI Slave mode: TX drives MISO
SPIn_RX		I	**SPI RX Input** Master mode: RX drives MISO Slave mode: RX drives MOSI

(1) I: Input, O: Output, I/O: Bidirectional

23.4 Functional Description

The SPI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. Internal FIFO memories buffer the transmit and receive paths, allowing independent storage of up to eight 32-bit values in both transmit and receive modes. The SPI also supports the µDMA interface. The TX and RX FIFOs can be programmed as destination or source addresses in the µDMA module. The µDMA operation is enabled by setting the appropriate bits in the SPI:DMACR register.

23.4.1 Bit Rate Generation

The SPI includes a programmable bit rate clock divider and prescaler to generate the serial output clock.

The serial bit rate is derived by dividing down the input clock. First, the clock is divided by a division RATIO from 1 to 8, which is programmed in SPI:CLKDIV2 (1 means that the clock is not divided). The clock is further divided by a value from 1 to 1024, which is 1 + SCR, where SCR is the value programmed in SPI:CLKCTL. See Section 23.7 for details on the SPI registers.

Equation 8 defines the frequency of the output clock SPIn_CLK.

$$SPIn\_CLK = PERDMACLK / [RATIO \times (1 + SCR)] \quad (8)$$

Note

For both master and slave modes, the core clock (PERDMACLK) must be at least two times faster than SPIn_CLK.

23.4.2 FIFO Operation

23.4.2.1 Transmit FIFO

The common TX FIFO is a 32 bit wide, 8 location deep, first-in first-out memory buffer. The CPU writes data to the FIFO by writing the SPI TX Data register, SPI:TXDATA, and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master (or a slave), parallel data is written into the TX FIFO before serial conversion and transmission to the attached slave or master, respectively, through the SPIn_MOSI (or SPIn_MISO) pin via SPIn_TX output.

In slave mode, the SPI transmits data each time the master initiates a transaction. If the TX FIFO is empty and the master initiates a transaction, the slave transmits garbage data. User or software is responsible to ensure valid data is available in the FIFO as needed. The SPI can be configured to generate an interrupt when a configurable level within the FIFO is selected via SPI:IFLS, or a µDMA single request when the FIFO is not FULL.

23.4.2.1.1 Repeated Transmit Operation

Using the CTL1.REPEATTX bits the last character transmitted can be repeated as many times as configured within the field. SPI transfer can be started by writing a data once into the TX FIFO. This feature can be used to transmit the same data repeatedly as if the data were written into the TXFIFO SPI:CTL1.REPEATTX number of times. It can be used to clean a transfer or to pull a certain amount of data from a peripheral. A value of 0 in CTL1.REPEATTX disables this feature. This feature is only available in master mode.

When REPEATTX is used it needs to be aligned with the data in the FIFO, hence the below shown sequence should be used:

- Check and wait till FIFO is empty
- Setup REPEATTX
- Write to TXDATA / TXFIFO
- Wait till requested data has been received

23.4.2.2 Receive FIFO

The common RX FIFO is a 32 bit wide, 8 location deep, first-in-first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the RX FIFO by reading the SPI:RXDATA register.

When configured as a master (or slave), serial data received through the SPIn_MISO (or SPIn_MOSI) pin via SPIn_RX is registered before parallel loading into the attached slave or master RX FIFO, respectively.

23.4.2.3 FIFO Flush

SPI includes a feature to reset TX and RX FIFO pointers to flush FIFOs. This has to be triggered when no SPI transactions are in progress. If a FIFO flush is triggered when a transaction is in progress, a second FIFO flush would be needed when no operations are ongoing, before restarting fresh SPI transfers.

SPI FIFO flush has to follow this sequence:

- SPI enable is set low via SPI:CTL1.ENABLE register bit
- SPI:CTL1.FIFORST register bit is forced high to trigger FIFO flush
- A wait of 4 to 5 CPU clock cycles is included (needed to ensure that the double synchronizers within the async FIFOs are cleared)
- SPI:CTL1.FIFORST is released by setting this to zero
- SPI enable can be now set high to restart SPI communication

23.4.3 Interrupts

The SPI can generate interrupts when the following conditions are observed:

- TX FIFO service (with the TX FIFO level configured via SPI:IFLS.TXIFLSEL)
- RX FIFO service (with the RX FIFO level configured via SPI:IFLS.RXIFLSEL)

- RX FIFO timeout
- RX FIFO overrun
- TX FIFO empty
- TX DMA done/RX DMA done
- Idle
- Parity error

All interrupt events are ORed together before sent to the event fabric, so the SPI generates a single interrupt request regardless of the number of active interrupts. The interrupt conditions listed above can be masked by setting the appropriate bit in the SPI:IMASK register. Setting the appropriate mask bit in the SPI:IMASK register enables the interrupt.

The status of the individual interrupt sources can be read from the SPI Raw Interrupt Status register (SPI:RIS) and the SPI Masked Interrupt Status register (SPI:MIS) (see Section 23.7 for details).

The transmit FIFO service interrupt request SPI:RIS.TX is not gated with the SPI enable signal, which allows data to be written to the transmit FIFO by an interrupt service routine (ISR), before enabling the SPI.

Note

TX and RX FIFO interrupts are best serviced by µDMA rather than CPU. In case CPU services TX and RX FIFO interrupts, TXEMPTY and RXFIFO_OVF can be configured as well, so that if the FIFO interrupt is missed by the CPU in corner cases, these act as a failsafe

The receive FIFO overflow interrupt SPI:RIS.RXFIFO_OVF is asserted when the FIFO is already full and an additional data frame is received, causing an overrun of the FIFO. Data is overwritten in the receive shift register, but not in the FIFO.

The parity error interrupt SPI:RIS.PER is set when a parity error is detected. SPI:CTL1.PEN bit can be written to enable the parity check, where the last bit received will be used as parity to test the integrity of the previous bits. SPI:CTL1.PES bit selects the parity mode as even or odd. When a parity fault is detected, the interrupt flag SPI:RIS.PER is set (to mark the data as invalid).

The idle interrupt SPI:RIS.IDLE is set when the SPI transmission has concluded and SPI module is back to idle mode. This is set when SPI:STAT.BUSY goes low.

The SPI slave Receive Timeout interrupt is set when SPI is in Slave mode and has not been receiving data for the number of functional clock cycles of the SPI clock (PERDMACLK) configured by SPI:CTL1.RXTIMEOUT. A value of 0 disables this function. The countdown is started when SPI is in the slave mode and the first CLK positive edge is detected and is countdown is restarted on each subsequent CLK positive edge. A timeout error is asserted if the count reaches zero before the next CLK toggles.

23.4.4 Data Format

Each data frame is between 4 and 32 bits long, depending on the size of data programmed. The control bit SPI:CTL1.MSB field can be programmed to define the direction of the data input and output as most-significant-bit (MSB) or least-significant-bit (LSB) first. If parity is enabled, the parity bit is always received as the last bit.

With the control register bits SPI:CTL0.DSS, the bit length per transfer is defined between 4 – 32 bits for master mode and 7 – 32 bits for slave mode.

23.4.5 Delayed Data Sampling

In cases when the input data arrives at the MISO pin with additional delay due to runtime conditions or path delays, on the following input data sampling stage, the previous data would be sampled at the sampling clock edge. To compensate for this, sampling of input data in master mode can be delayed using CLKCTL.DSAMPLE bits. The delayed sampling is only available in master mode. The delay can be adjusted in steps of undivided SPI input clocks programmed within SPI:CLKCTL.DSAMPLE. The programmed value of DSAMPLE must be within 0 to SCR+1. Typically, values of 1 and 2 are sufficient even for the highest supported SPI frequencies.

23.4.6 Frame Formats

The following four basic frame formats can be selected through SPI:CTL0.FRF field:

- TI synchronous serial
- Motorola SPI (3-wire)
- Motorola SPI (4-wire)
- National MICROWIRE

For all four formats, the serial clock (SPIn_CLK) is held inactive while the SPI is idle and SPIn_CLK transitions at the programmed frequency only during active transmission or reception of data. The IDLE state of SPIn_CLK provides a receive time-out indication that occurs when the RX FIFO still contains data after a time-out period.

For Motorola SPI (4-wire) and MICROWIRE frame formats, the serial frame (SPIn_CS) pin is active low and is asserted (pulled down) during the entire transmission of the frame.

For TI synchronous serial frame format, the SPIn_CS pin is pulsed for one serial clock period which starts at its rising edge before the transmission of each frame. For this frame format, both the SPI and the off-chip slave device drive their output data on the rising edge of SPIn_CLK and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other three frame formats, the MICROWIRE format uses a special master-slave messaging technique that operates at half-duplex. When a frame begins, an 8-bit control message is transmitted to the off-chip slave. No incoming data is received by the SPI during this transmission. After the message is sent, the off-chip slave decodes it and responds with the requested data after waiting one serial clock after the last bit of the 8-bit control message is sent. The returned data can be 4 to 32 bits long (master mode) or 7 to 32 bits (slave mode), making the total frame length anywhere from 13 to 41 bits.

23.4.6.1 Texas Instruments Synchronous Serial Frame Format

Figure 23-2 shows the TI synchronous serial frame format for a single transmitted frame.

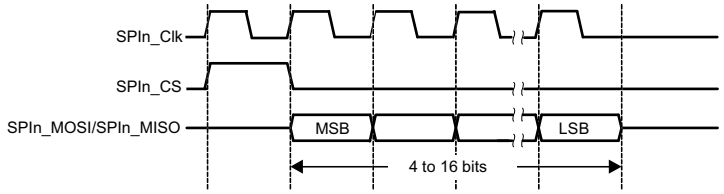


Figure 23-2. TI Synchronous Serial Frame Format (Single Transfer)

SPIn_CLK and SPIn_CS are forced low and the transmit data line SPIn_MOSI is put in tristate whenever the SPI is idle. When the bottom entry of the TX FIFO contains data, SPIn_CS is pulsed high for one SPIn_CLK period.

The transmitted value is also transferred from the TX FIFO to the serial shift register of the transmit logic. On the next rising edge of SPIn_CLK, the MSB of the 4-bit to 32-bit data frame is shifted out on the SPIn_MOSI pin. Likewise, the MSB of the received data is shifted onto the SPIn_MISO pin by the off-chip serial slave device.

Both the SPI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SPIn_CLK. The received data is transferred from the serial shifter to the RX FIFO on the first rising edge of SPIn_CLK after the least significant bit (LSB) is latched.

Figure 23-3 shows the TI synchronous serial frame format when back-to-back frames are transmitted.

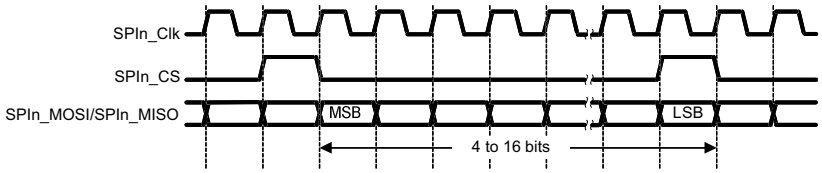


Figure 23-3. TI Synchronous Serial Frame Format (Continuous Transfer)

23.4.6.2 Motorola SPI Frame Format

The Motorola SPI is a 4-wire interface where the SPIn_CS signal behaves as a slave select. In the 3-wire interface the SPIn_CS is unused, and behaves like this signal is always asserted and SPI is always selected. The selection between the 3-wire and 4-wire Motorola SPI Frame Format can be made within the SPI:CTL0.FRF field. The main feature of the Motorola SPI format is that the inactive state and phase of the SPIn_CLK signal can be programmed through the SPO and SPH bits in the SPI:CTL0 control register.

23.4.6.2.1 SPO Clock Polarity Bit

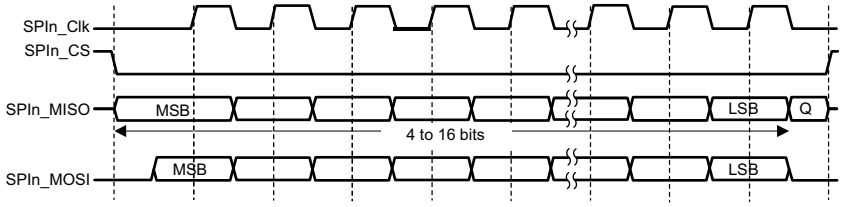
When the SPO clock polarity control bit is cleared, it produces a steady-state low value on the SPIn_CLK pin. If the SPO bit is set, it places a steady-state high value on the SPIn_CLK pin when data is not being transferred.

23.4.6.2.2 SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data, and allows it to change state. The state of this bit has the most impact on the first bit transmitted, by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is cleared, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

23.4.6.3 Motorola SPI Frame Format with SPO = 0 and SPH = 0

Figure 23-4 and Figure 23-5 show single and continuous transmission signal sequences for Motorola SPI format with SPO = 0 and SPH = 0, respectively.



Q is undefined.

Figure 23-4. Motorola™ SPI Format (Single Transfer) with SPO = 0 and SPH = 0

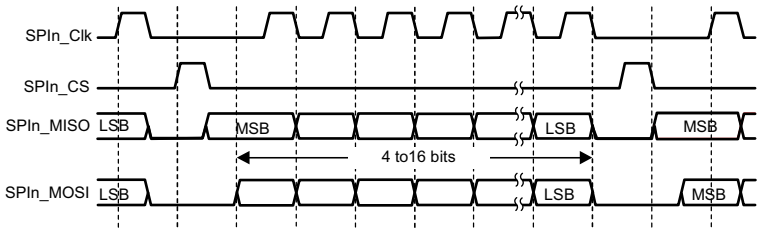


Figure 23-5. Motorola SPI Format (Continuous Transfer) with SPO = 0 and SPH = 0

In this configuration, the following occurs during idle periods:

- SPIn_CLK is forced low
- SPIn_CS is forced high
- The transmit data line SPIn_MOSI is forced low
- When the SPI is configured as a master, the SPI enables the SPIn_CLK
- When the SPI is configured as a slave, the SPI disables the SPIn_CLK

If the SPI is enabled and valid data is in the TX FIFO, the SPIn_CS master signal is driven low at the start of transmission which causes enabling of slave data onto the SPIn_MISO input line of the master. The master SPIn_MOSI output is enabled.

One-half SPIn_CLK period later, valid master data is transferred to the SPIn_MOSI pin. Once both the master and slave data are set, the SPIn_CLK master clock pin goes high after an additional one-half SPIn_CLK period.

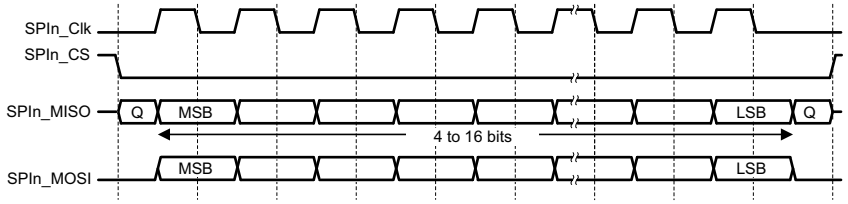
The data is now captured on the rising edges and propagated on the falling edges of the SPIn_CLK signal.

For a single-word transmission after all bits of the data word are transferred, the SPIn_CS line is returned to its IDLE high state one SPIn_CLK period after the last bit is captured.

For continuous back-to-back transmissions, the SPIn_CS signal must pulse high between each data word transfer because the slave-select pin freezes the data in its serial peripheral register and does not allow altering of the data if the SPH bit is clear. The master device must raise the SPIn_CS pin of the slave device between each data transfer to enable the serial peripheral data write. When the continuous transfer completes, the SPIn_CS pin is returned to its IDLE state one SPIn_CLK period after the last bit is captured.

23.4.6.4 Motorola SPI Frame Format with SPO = 0 and SPH = 1

Figure 23-6 shows the transfer signal sequence for Motorola SPI format with SPO = 0 and SPH = 1, which covers both single and continuous transfers.



Q is undefined.

Figure 23-6. Motorola SPI Frame Format with SPO = 0 and SPH = 1

In this configuration, the following occurs during idle periods:

- SPIn_CLK is forced low
- SPIn_CS is forced high
- The transmit data line SPIn_MOSI is typically forced low
- When the SPI is configured as a master, the SPI enables the SPIn_CLK
- When the SPI is configured as a slave, the SPI disables the SPIn_CLK

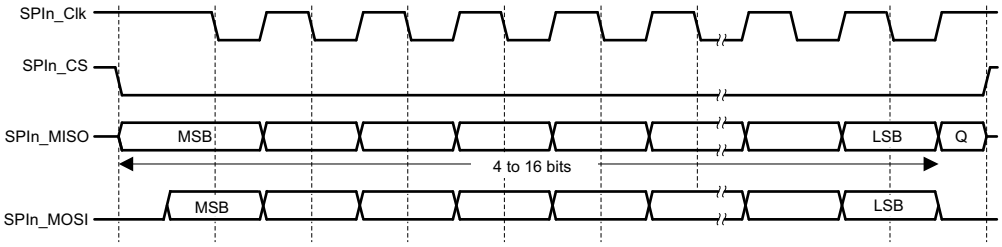
If the SPI is enabled and valid data is in the TX FIFO, the SPIn_CS master signal goes low at the start of transmission. The master SPIn_MOSI output is enabled. After an additional one-half SPIn_CLK period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, SPIn_CLK is enabled with a rising-edge transition. Data is then captured on the falling edges and propagated on the rising edges of the SPIn_CLK signal.

For a single-word transfer, after all bits are transferred, the SPIn_CS line is returned to its IDLE high state one SPIn_CLK period after the last bit is captured.

For continuous back-to-back transfers, the SPIn_CS pin is held low between successive data words and terminates like a single-word transfer.

23.4.6.5 Motorola SPI Frame Format with SPO = 1 and SPH = 0

Figure 23-7 and Figure 23-8 show single and continuous transmission signal sequences, respectively, for Motorola SPI format with SPO = 1 and SPH = 0.



Q is undefined.

Figure 23-7. Motorola SPI Frame Format (Single Transfer) with SPO = 1 and SPH = 0

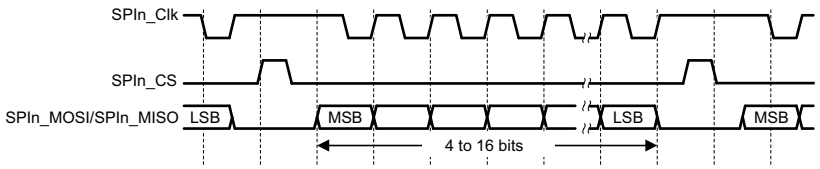


Figure 23-8. Motorola SPI Frame Format (Continuous Transfer) with SPO = 1 and SPH = 0

In this configuration, the following occurs during idle periods:

- SPIn_CLK is forced high
- SPIn_CS is forced high
- The transmit data line SPIn_MOSI is typically forced low
- When the SPI is configured as a master, the SPI enables the SPIn_CLK
- When the SPI is configured as a slave, the SPI disables the SPIn_CLK

If the SPI is enabled and valid data is in the TX FIFO, the SPIn_CS master signal goes low at the start of transmission and transfers slave data onto the SPIn_MISO line of the master immediately. The master SPIn_MOSI output is enabled.

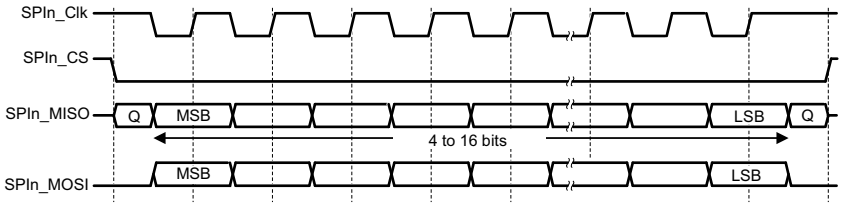
One-half SPIn_CLK period later, valid master data is transferred to the SPIn_MOSI line. When both the master and slave data have been set, the SPIn_CLK master clock pin becomes low after one additional half SPIn_CLK period. Data is captured on the falling edges and propagated on the rising edges of the SPIn_CLK signal.

For a single-word transmission after all bits of the data word are transferred, the SPIn_CS line is returned to its IDLE high state one SPIn_CLK period after the last bit is captured.

For continuous back-to-back transmissions, the SPIn_CS signal must pulse high between each data word transfer as the slave-select pin freezes the data in its serial peripheral register and keeps it from being altered if the SPH bit is clear. The master device must raise the SPIn_CS pin of the slave device between each data transfer to enable the serial peripheral data write. When the continuous transfer completes, the SPIn_CS pin returns to its IDLE state one SPIn_CLK period after the last bit is captured.

23.4.6.6 Motorola SPI Frame Format with SPO = 1 and SPH = 1

Figure 23-9 shows the transfer signal sequence for Motorola SPI format with SPO = 1 and SPH = 1, which covers both single and continuous transfers.



Q is undefined.

Figure 23-9. Motorola SPI Frame Format with SPO = 1 and SPH = 1

In this configuration, the following occurs during idle periods:

- SPIn_CLK is forced high
- SPIn_CS is forced high
- The transmit data line SPIn_MOSI is typically forced low
- When the SPI is configured as a master, the SPI enables the SPIn_CLK
- When the SPI is configured as a slave, the SPI disables the SPIn_CLK

If the SPI is enabled and valid data is in the TX FIFO, the start of transmission is signified by the SPIn_CS master signal going low. The master SPIn_MOSI output DIO is enabled. After an additional one-half SPIn_CLK period, both master and slave data are enabled onto their respective transmission lines. At the same time, SPIn_CLK is enabled with a falling-edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SPIn_CLK signal.

For a single word transmission, after all bits are transferred, the SPIn_CS line returns to its IDLE high state one SPIn_CLK period after the last bit is captured.

For continuous back-to-back transmissions, the SPIn_CS pin remains in its active low state until the final bit of the last word is captured and then returns to its IDLE state.

For continuous back-to-back transfers, the SPIn_CS pin is held low between successive data words and terminates like a single-word transfer.

23.4.6.7 MICROWIRE Frame Format

Figure 23-10 shows the MICROWIRE frame format for a single frame. Figure 23-11 shows the same format when back-to-back frames are transmitted.

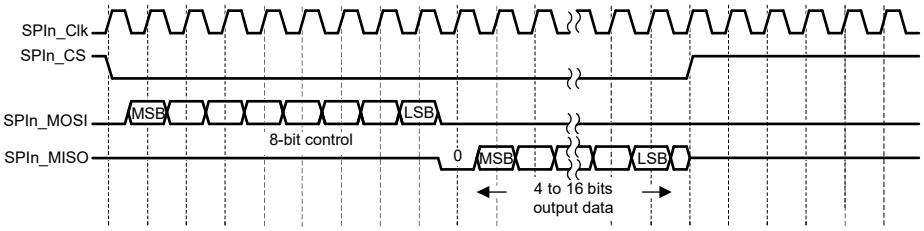


Figure 23-10. MICROWIRE Frame Format (Single Frame)

MICROWIRE format is similar to SPI format, except that transmission is half-duplex and uses a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SPI to the off-chip slave device. During this transmission, the SPI does not receive incoming data. After the message is sent, the off-chip slave decodes it and waits one serial clock after the last bit of the 8-bit control message is sent. The off-chip slave then responds with the required data. The returned data is 4 to 32 bits long, making the total frame length anywhere from 13 to 41 bits.

In this configuration, the following occurs during idle periods:

- SPIn_CLK is forced low
- SPIn_CS is forced high
- The transmit data line SPIn_MOSI is typically forced low

Writing a control byte to the TX FIFO triggers a transmission. The falling edge of SPIn_CS transfers the value in the bottom entry of the TX FIFO to the serial shift register of the transmit logic and shifts the MSB of the 8-bit control frame out onto the SPIn_MOSI pin. SPIn_CS remains low for the duration of the frame transmission. The SPIn_MISO pin remains in the tri-state condition during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on each rising edge of SPIn_CLK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait state and the slave responds by transmitting data back to the SPI. Each bit is driven onto the SPIn_MISO line on the falling edge of SPIn_CLK. The SPI latches each bit on the rising edge of SPIn_CLK. At the end of the frame for single transfers, the SPIc_CS signal is pulled high one clock period after the last bit is latched in the receive serial shifter transferring the data to the RX FIFO.

Note

The off-chip slave device can place the receive line in a tri-state condition either on the falling edge of SPIn_CLK (after the LSB has been latched by the receive shifter), or when the SPIn_CS pin goes high.

For continuous transfers, data transmission begins and ends like a single transfer, but the SPIn_CS line is held low and data transmits back-to-back. The control byte of the next frame follows the LSB of the received data from the current frame. After the LSB of the frame is latched into the SPI, each received value is transferred from the receive shifter on the falling edge of SPIn_CLK.

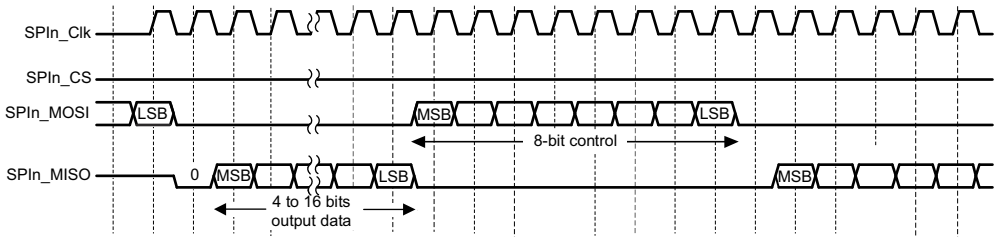


Figure 23-11. MICROWIRE Frame Format (Continuous Transfer)

In the MICROWIRE mode, the SPI slave samples the first bit of receive data on the rising edge of SPIn_CLK after SPIn_CS has gone low. Masters driving a free-running SPIn_CLK must ensure that the SPIn_CS signal has sufficient setup and hold margins compared to the rising edge of SPIn_CLK.

23.5 µDMA Operation

The SPI peripheral provides an interface to the µDMA controller with separate channels for transmit and receive. The SPI DMA Control register (SPI:DMACR) allows the µDMA to operate together with the SPI. When µDMA operation is enabled, the SPI asserts a µDMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the RX FIFO. Whenever data in the RX FIFO reaches the configured level set via SPI:IFLS.RXIFLSEL, a burst transfer request is asserted. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the TX FIFO. Whenever the TX FIFO reaches the configured level set via SPI:IFLS.TXIFLSEL, the burst request is asserted. The µDMA controller handles the single and burst µDMA transfer requests automatically depending on how the µDMA channel is configured.

To enable µDMA operation for the receive channel, set the SPI:DMACR.RXDMAE register bit. To enable µDMA operation for the transmit channel, set the SPI:DMACR.TXDMAE register bit. If the µDMA is enabled and appropriate bits are cleared in the DMA Done Mask register (UDMA:DONEMASK) the µDMA controller triggers an interrupt when a transfer completes. The interrupt occurs on the SPI interrupt vector. If interrupts are used for SPI operation and the µDMA is enabled, the SPI interrupt handler must be designed to handle the µDMA completion interrupt. The status of TX and RX DMA done interrupts can be read from the Channel Request Done register (UDMA:REQDONE). They can also be read from SPI:RIS.DMA_DONE_TX and SPI:RIS.DMA_DONE_RX register bits. These can be used for generating an interrupt as well. For clearing the TX and RX DMA done interrupts, the corresponding bits in the UDMA:REQDONE register must be 1.

For more details about programming the µDMA controller, see Chapter 16.

23.6 Initialization and Configuration

The following describes the necessary steps to enable and initialize the SPI.

TI recommends using the SPI driver in the SimpleLink™ CC13xx and CC26xx software development kit (SDK) when using the SPI.

1. Ensure the corresponding power domain is powered up properly. For details, see Chapter 7
2. Enable the appropriate SPI module in PRCM by writing to PRCM:SPICLKGR, PRCM:SPICLKGS, and PRCM:SPICLKGDS register and load the setting to the clock controller by writing to PRCM:CLKLOADCTL.
3. Configure the IOC module to route the SPIn_TX, SPIn_RX, SPIn_CS, and SPIn_CLK functionalities from I/Os to the SPI module. IOCFGn.PORT_ID must be written to the correct PORT_IDs. See Section 15.10.3 for details.

For each of the frame formats, the SPI is configured using the following steps:

1. Ensure that the ENABLE bit in the SPI:CTL1 register is clear before making any configuration changes.
2. Select whether the SPI is a master or slave:
   a. For master operations, SPI:CTL1.MS should be 1
   b. For slave mode (output enabled), SPI:CTL1.MS should be 0
   c. For slave mode (output disabled), set the SPI:CTL1.MS bit to 0x0 and SPI:CTL1.SOD bit to 0x1
3. Configure the clock prescale divisor by writing to the SPI:CLKDIV2.RATIO and SPI:CLKCTL.SCR fields
4. Write the SPI:CTL0 register with the following configuration:
   a. Desired clock phase and polarity, if using Motorola™ SPI mode (SPH and SPO)
   b. The protocol mode: Motorola SPI (4-wire or 3-wire) , TI SSF, MICROWIRE (FRF)
   c. The data size (DSS)
5. Optionally, configure the µDMA channel (see Chapter 16) and enable the DMA options in the SPI:DMACR register.
6. Enable the SPI by setting the ENABLE bit in the SPI:CTL1 register.

As an example, assume that the SPI configuration is required to operate with the following parameters:

- Master operation
- Texas Instruments SPI mode
- 1-Mbps bit rate
- 8 data bits

Assuming the system clock is 48 MHz, the bit-rate calculation is shown in Equation 9.

$$\text{SPIn\_CLK} = \text{PERDMACLK} / [\text{RATIO} \times (1 + \text{SCR})] \quad (9)$$
$$1000000 \text{ bps} = 48000000 \text{ Hz} / [2 \times (1 + 23)]$$

In this case, if RATIO = 0x2, SCR must be 0x18.

The configuration sequence is:

1. Ensure that the ENABLE bit in the SPI:CTL1 register is clear
2. Write the SPI:CTL1 register with a value of 0x00000004
3. Write the SPI:CLKDIV2 register with a value of 0x00000002
4. Write the SPI:CLKCTL.SCR register with a value of 0x00000018
5. Write the SPI:CTL0 register with a value of 0x0000 47
6. The SPI is then enabled by setting the ENABLE bit in the SPI:CTL1 register

23.7 SPI Registers

Table 23-2 lists the memory-mapped registers for the SPI registers. All register offset addresses not listed in Table 23-2 should be considered as reserved locations and the register contents should not be modified.

Table 23-2. SPI Registers

Offset	Acronym	Register Name	Section
20h	IIDX	This register provides the highest priority enabled interrupt index. Value 0x00 means no event pending. Interrupt 1 is the highest priority, and 31 is the least priority. That is, the least bit position that is set to 1 denotes the highest priority pending interrupt. The priority order is fixed. However, users can implement their own prioritization schemes using other registers that expose the full set of interrupts that have occurred.	Section 23.7.1
28h	IMASK	Interrupt Mask. If a bit is set, then corresponding interrupt is un-masked. Un-masking the interrupt causes the raw interrupt to be visible in IIDX, as well as MIS.	Section 23.7.2
30h	RIS	Raw interrupt status. Reflects all pending interrupts, regardless of masking. The RIS register allows the user to implement a poll scheme. A flag set in this register can be cleared by writing 1 to the ICLR register bit even if the corresponding IMASK bit is not enabled.	Section 23.7.3
38h	MIS	Masked interrupt status. This is an AND of the IMASK and RIS registers.	Section 23.7.4
40h	ISET	Interrupt set. Allows interrupts to be set by software (useful in diagnostics and safety checks). Writing a 1 to a bit in ISET will set the event and therefore the related RIS bit also gets set. If the interrupt is enabled through the mask, then the corresponding MIS bit is also set.	Section 23.7.5
48h	ICLR	Interrupt clear. Write a 1 to clear the corresponding Interrupt.	Section 23.7.6
E0h	EVT_MODE	Event mode register. It is used to select whether each line is disabled, in software mode (software clears the RIS) or in hardware mode (hardware clears the RIS)	Section 23.7.7
FCh	DESC	This register identifies the peripheral and its exact version.	Section 23.7.8
100h	CTL0	SPI Control Register 0	Section 23.7.9
104h	CTL1	SPI Control Register 1	Section 23.7.10
108h	CLKCTL	Clock prescaler and divider register. This register contains the settings for the Clock prescaler and divider settings.	Section 23.7.11
10Ch	IFLS	The IFLS register is the interrupt FIFO level select register. This register can be used to define the levels at which the TX, RX FIFO interrupt flags are triggered. Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.	Section 23.7.12
110h	STAT	Status Register	Section 23.7.13
114h	CLKDIV2	This register is used to specify a divide ratio of the SPI functional clock.	Section 23.7.14
118h	DMACR	DMA Control Register	Section 23.7.15
130h	RXDATA	RXDATA Register. Reading this register returns value in the RX FIFO pointed by the current FIFO read pointer. If the RX FIFO is empty, the last read value is returned. Writing has not effect and is ignored.	Section 23.7.16
140h	TXDATA	TXDATA Register. Writing into this register puts the data into the TX FIFO. Reading this register returns the last written value, pointed by the current FIFO write pointer.	Section 23.7.17

Complex bit access types are encoded to fit into small table cells. Table 23-3 shows the codes that are used for access types in this section.

Table 23-3. SPI Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

23.7.1 IIDX Register (Offset = 20h) [Reset = 00000000h]

IIDX is shown in Table 23-4.

Return to the Summary Table.

This register provides the highest priority enabled interrupt index. Value 0x00 means no event pending. Interrupt 1 is the highest priority, and 31 is the least priority. That is, the least bit position that is set to 1 denotes the highest priority pending interrupt. The priority order is fixed. However, users can implement their own prioritization schemes using other registers that expose the full set of interrupts that have occurred.
On each read, only one interrupt is indicated. On a read, the current interrupt (highest priority) is automatically cleared by the hardware and the corresponding interrupt flag in RIS and MIS are cleared as well. After a read from the CPU (not from the debug interface), the register is updated with the next highest priority interrupt, if none are pending, then it would display 0x0.

Table 23-4. IIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-0	STAT	R	0h	Interrupt index status 0h = No interrupt pending 1h = RX FIFO Overflow Event/interrupt pending 2h = Transmit Parity Event/interrupt pending 3h = SPI Receive Time-Out Event/interrupt pending 4h = Receive Event/interrupt pending 5h = Transmit Event/interrupt pending 6h = Transmit Buffer Empty Event/interrupt pending 7h = End of Transmit Event/interrupt pending 8h = DMA Done for Receive Event/interrupt pending 9h = DMA Done for Transmit Event/interrupt pending

23.7.2 IMASK Register (Offset = 28h) [Reset = 00000000h]

IMASK is shown in Table 23-5.

Return to the Summary Table.

Interrupt Mask. If a bit is set, then corresponding interrupt is un-masked. Un-masking the interrupt causes the raw interrupt to be visible in IIDX, as well as MIS.

Table 23-5. IMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	DMA_DONE_TX	R/W	0h	DMA Done event for TX event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
7	DMA_DONE_RX	R/W	0h	DMA Done event for RX event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
6	IDLE	R/W	0h	SPI Idle event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
5	TXEMPTY	R/W	0h	Transmit FIFO Empty event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
4	TX	R/W	0h	Transmit FIFO event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
3	RX	R/W	0h	Receive FIFO event. This interrupt is set if the selected Receive FIFO level has been reached 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
2	RTOUT	R/W	0h	SPI Receive Time-Out event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
1	PER	R/W	0h	Parity error event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask
0	RXFIFO_OVF	R/W	0h	RXFIFO overflow event mask. 0h = Clear Interrupt Mask 1h = Set Interrupt Mask

23.7.3 RIS Register (Offset = 30h) [Reset = 00000000h]

RIS is shown in Table 23-6.

Return to the Summary Table.

Raw interrupt status. Reflects all pending interrupts, regardless of masking. The RIS register allows the user to implement a poll scheme. A flag set in this register can be cleared by writing 1 to the ICLR register bit even if the corresponding IMASK bit is not enabled.

Table 23-6. RIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	DMA_DONE_TX	R	0h	DMA Done event for TX. This interrupt is set if the TX DMA channel sends the DONE signal. This allows the handling of the DMA event inside the mapped peripheral. 0h = Interrupt did not occur 1h = Interrupt occurred
7	DMA_DONE_RX	R	0h	DMA Done event for RX. This interrupt is set if the RX DMA channel sends the DONE signal. This allows the handling of the DMA event inside the mapped peripheral. 0h = Interrupt did not occur 1h = Interrupt occurred
6	IDLE	R	0h	SPI has completed transfers and changed into IDLE mode. This bit is set when STAT.BUSY goes low. 0h = Interrupt did not occur 1h = Interrupt occurred
5	TXEMPTY	R	0h	Transmit FIFO Empty interrupt mask. This interrupt is set if all data in the Transmit FIFO have been move to the shift register. 0h = Interrupt did not occur 1h = Interrupt occurred
4	TX	R	0h	Transmit FIFO event. This interrupt is set if the selected Transmit FIFO level has been reached. 0h = Interrupt did not occur 1h = Interrupt occurred
3	RX	R	0h	Receive FIFO event. This interrupt is set if the selected Receive FIFO level has been reached 0h = Interrupt did not occur 1h = Interrupt occurred
2	RTOUT	R	0h	SPI Receive Time-Out event. 0h = Interrupt did not occur 1h = Interrupt occurred
1	PER	R	0h	Parity error event: this bit is set if a parity error has been detected 0h = Interrupt did not occur 1h = Interrupt occurred
0	RXFIFO_OVF	R	0h	RXFIFO overflow event. This interrupt is set if an RX FIFO overflow has been detected. 0h = Interrupt did not occur 1h = Interrupt occurred

23.7.4 MIS Register (Offset = 38h) [Reset = 00000000h]

MIS is shown in Table 23-7.

Return to the Summary Table.

Masked interrupt status. This is an AND of the IMASK and RIS registers.

Table 23-7. MIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	DMA_DONE_TX	R	0h	Masked DMA Done event for TX. 0h = Interrupt did not occur 1h = Interrupt occurred
7	DMA_DONE_RX	R	0h	Masked DMA Done event for RX. 0h = Interrupt did not occur 1h = Interrupt occurred
6	IDLE	R	0h	Masked SPI IDLE mode event. 0h = Interrupt did not occur 1h = Interrupt occurred
5	TXEMPTY	R	0h	Masked Transmit FIFO Empty event. 0h = Interrupt did not occur 1h = Interrupt occurred
4	TX	R	0h	Masked Transmit FIFO event. This interrupt is set if the selected Transmit FIFO level has been reached. 0h = Interrupt did not occur 1h = Interrupt occurred
3	RX	R	0h	Masked receive FIFO event. This interrupt is set if the selected Receive FIFO level has been reached 0h = Interrupt did not occur 1h = Interrupt occurred
2	RTOUT	R	0h	Masked SPI Receive Time-Out Interrupt. 0h = Interrupt did not occur 1h = Interrupt occurred
1	PER	R	0h	Masked Parity error event: this bit if a parity error has been detected 0h = Interrupt did not occur 1h = Interrupt occurred
0	RXFIFO_OVF	R	0h	Masked RXFIFO overflow event. This interrupt is set if an RX FIFO overflow has been detected. 0h = Interrupt did not occur 1h = Interrupt occurred

23.7.5 ISET Register (Offset = 40h) [Reset = 00000000h]

ISET is shown in Table 23-8.

Return to the Summary Table.

Interrupt set. Allows interrupts to be set by software (useful in diagnostics and safety checks). Writing a 1 to a bit in ISET will set the event and therefore the related RIS bit also gets set. If the interrupt is enabled through the mask, then the corresponding MIS bit is also set.

Table 23-8. ISET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	DMA_DONE_TX	W	0h	Set DMA Done event for TX. 0h = Writing 0 has no effect 1h = Set Interrupt
7	DMA_DONE_RX	W	0h	Set DMA Done event for RX. 0h = Writing 0 has no effect 1h = Set Interrupt
6	IDLE	W	0h	Set SPI IDLE mode event. 0h = Writing 0 has no effect 1h = Set Interrupt
5	TXEMPTY	W	0h	Set Transmit FIFO Empty event. 0h = Writing 0 has no effect 1h = Set Interrupt
4	TX	W	0h	Set Transmit FIFO event. 0h = Writing 0 has no effect 1h = Set Interrupt
3	RX	W	0h	Set Receive FIFO event. 0h = Writing 0 has no effect 1h = Set Interrupt
2	RTOUT	W	0h	Set SPI Receive Time-Out event. 0h = Writing 0 has no effect 1h = Set Interrupt
1	PER	W	0h	Set Parity error event. 0h = Writing 0 has no effect 1h = Set Interrupt
0	RXFIFO_OVF	W	0h	Set RXFIFO overflow event. 0h = Writing 0 has no effect 1h = Set Interrupt

23.7.6 ICLR Register (Offset = 48h) [Reset = 00000000h]

ICLR is shown in Table 23-9.

Return to the Summary Table.

Interrupt clear. Write a 1 to clear the corresponding Interrupt.

Table 23-9. ICLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
8	DMA_DONE_TX	W	0h	Clear DMA Done event for TX. 0h = Writing 0 has no effect 1h = Clear Interrupt
7	DMA_DONE_RX	W	0h	Clear DMA Done event for RX. 0h = Writing 0 has no effect 1h = Clear Interrupt
6	IDLE	W	0h	Clear SPI IDLE mode event. 0h = Writing 0 has no effect 1h = Clear Interrupt
5	TXEMPTY	W	0h	Clear Transmit FIFO Empty event. 0h = Writing 0 has no effect 1h = Clear Interrupt
4	TX	W	0h	Clear Transmit FIFO event. 0h = Writing 0 has no effect 1h = Clear Interrupt
3	RX	W	0h	Clear Receive FIFO event. 0h = Writing 0 has no effect 1h = Clear Interrupt
2	RTOUT	W	0h	Clear SPI Receive Time-Out event. 0h = Writing 0 has no effect 1h = Clear Interrupt
1	PER	W	0h	Clear Parity error event. 0h = Writing 0 has no effect 1h = Clear Interrupt
0	RXFIFO_OVF	W	0h	Clear RXFIFO overflow event. 0h = Writing 0 has no effect 1h = Clear Interrupt

23.7.7 EVT_MODE Register (Offset = E0h) [Reset = 00000001h]

EVT_MODE is shown in Table 23-10.

Return to the Summary Table.

Event mode register. It is used to select whether each line is disabled, in software mode (software clears the RIS) or in hardware mode (hardware clears the RIS)
Note: The recommendation is to use SPI in the software mode

Table 23-10. EVT_MODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1-0	INT0_CFG	R/W	1h	Event line mode select for event corresponding to IPSTANDARD.INT_EVENT0 0h = The interrupt or event line is disabled. 1h = The interrupt or event line is in software mode. Software must clear the RIS. 2h = The interrupt or event line is in hardware mode. The hardware automatically clears the RIS flag.

23.7.8 DESC Register (Offset = FCh) [Reset = 14110010h]

DESC is shown in Table 23-11.

Return to the Summary Table.

This register identifies the peripheral and its exact version.

Table 23-11. DESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MODULEID	R	1411h	Module identification contains a unique peripheral identification number. The assignments are maintained in a central database for all of the platform modules to ensure uniqueness.
15-12	FEATUREVER	R	0h	Feature set version for this module instance.
11-8	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
7-4	MAJREV	R	1h	Major revision of the IP
3-0	MINREV	R	0h	Minor revision of the IP

23.7.9 CTL0 Register (Offset = 100h) [Reset = 00000000h]

CTL0 is shown in Table 23-12.

Return to the Summary Table.

SPI Control Register 0

Table 23-12. CTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
14	CSCLR	R/W	0h	Clear shift register counter when CS gets inactive. This bit is relevant only in the slave mode, CTL1.MS = 0. 0h = Disable automatic clear of shift register when CS gets inactive. 1h = Enable automatic clear of shift register when CS gets inactive.
13-12	RESERVED	R	0h	Reserved
11-10	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
9	SPH	R/W	0h	CLKOUT phase (Motorola SPI frame format only) This bit selects the clock edge that captures data and enables it to change state. It has the most impact on the first bit transmitted by either permitting or not permitting a clock transition before the first data capture edge. 0h = Data is captured on the first clock edge transition. 1h = Data is captured on the second clock edge transition.
8	SPO	R/W	0h	CLKOUT polarity (Motorola SPI frame format only) 0h = SPI produces a steady state LOW value on the CLKOUT when data is not being transferred. 1h = SPI produces a steady state HIGH value on the CLKOUT when data is not being transferred.
7	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
6-5	FRF	R/W	0h	Frame format Select 0h = Motorola SPI frame format (3 wire mode) 1h = Motorola SPI frame format (4 wire mode) 2h = TI synchronous serial frame format 3h = National MICROWIRE frame format

Table 23-12. CTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-0	DSS	R/W	0h	Data Size Select. Note: Master mode: Values 0 - 2 are reserved and shall not be used. This will map to 4 bit mode. A value of 3h corresponds to 4-bit data (and so on). Slave mode: DSS should be no less than 6 which means the minimum frame length is 7 bits. 3h = Data Size Select bits: 4 4h = Data Size Select bits: 5 5h = Data Size Select bits: 6 6h = Data Size Select bits: 7 7h = Data Size Select bits: 8 8h = Data Size Select bits: 9 9h = Data Size Select bits: 10 Ah = Data Size Select bits: 11 Bh = Data Size Select bits: 12 Ch = Data Size Select bits: 13 Dh = Data Size Select bits: 14 Eh = Data Size Select bits: 15 Fh = Data Size Select bits: 16 10h = Data Size Select bits: 17 11h = Data Size Select bits: 18 12h = Data Size Select bits: 19 13h = Data Size Select bits: 20 14h = Data Size Select bits: 21 15h = Data Size Select bits: 22 16h = Data Size Select bits: 23 17h = Data Size Select bits: 24 18h = Data Size Select bits: 25 19h = Data Size Select bits: 26 1Ah = Data Size Select bits: 27 1Bh = Data Size Select bits: 28 1Ch = Data Size Select bits: 29 1Dh = Data Size Select bits: 30 1Eh = Data Size Select bits: 31 1Fh = Data Size Select bits: 32

23.7.10 CTL1 Register (Offset = 104h) [Reset = 00000004h]

CTL1 is shown in Table 23-13.

Return to the Summary Table.

SPI Control Register 1

Table 23-13. CTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
29-24	RXTIMEOUT	R/W	0h	Receive Timeout (only for Slave mode). This register defines the number of clock cycles after which the Receive Timeout interrupt is set. A value of 0 disables this function. 0h = Smallest value 3Fh = Highest possible value
23-16	REPEATTX	R/W	0h	Counter to repeat last transfer. A value of 0 disables this feature. After a non-zero value (X) is written to this register, SPI transfer can be started with writing a data into the TX Buffer. The data will be transferred X+1 times in total. The behavior is identical as if the data were be written into the TX Buffer that many times as defined by the value here additionally. It can be used to clean a transfer or to pull a certain amount of data by a slave. This feature can be used only in the master mode.
15-11	RESERVED	R	0h	Reserved
10	FIFORST	R/W	0h	This bit is used to reset transmit and receive FIFO pointers. The pointers are held at a reset value until this bit is cleared to zero. 0h = Clear FIFO pointers reset trigger 1h = Set FIFO pointers reset trigger
9-8	RESERVED	R	0h	Reserved
7	PBS	R/W	0h	Parity Bit Select 0h = Bit 0 is used for Parity 1h = Bit 1 is used for Parity, Bit 0 is ignored
6	PES	R/W	0h	Even Parity Select 0h = Odd Parity mode 1h = Even Parity mode
5	PEN	R/W	0h	Parity enable if enabled the last bit will be used as parity to evaluate the right transmission of the previous bits. In case of a parity mismatch the parity error flag RIS.PER will be set. 0h = Disable Parity function 1h = Enable Parity function
4	MSB	R/W	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0h = LSB first 1h = MSB first
3	SOD	R/W	0h	Slave-mode: Data output disabled This bit is relevant only in the slave mode, MS=0. In multiple-slave systems, it is possible for an SPI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems the RX lines from multiple slaves could be tied together. To operate in such systems, this bitfield can be set if the SPI slave is not supposed to drive the TX line. 0h = SPI can drive the MISO output via TX in slave mode. 1h = SPI cannot drive the MISO output via TX in slave mode.
2	MS	R/W	1h	Master or slave mode select. This bit can be modified only when SPI is disabled, CTL1.ENABLE = 0. 0h = Select Slave Mode 1h = Select Master Mode

Table 23-13. CTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	LBM	R/W	0h	Loop back mode 0h = Disable loopback mode. Normal serial port operation enabled. 1h = Enable loopback mode.Output of transmit serial shifter is connected to input of receive serial shifter internally.
0	ENABLE	R/W	0h	SPI enable 0h = Disable module function 1h = Enable module function

23.7.11 CLKCTL Register (Offset = 108h) [Reset = 00000000h]

CLKCTL is shown in Table 23-14.

Return to the Summary Table.

Clock prescaler and divider register. This register contains the settings for the Clock prescaler and divider settings.

Table 23-14. CLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	DSAMPLE	R/W	0h	Delayed sampling. In master mode the data on the input pin will be sampled after the defined clock cycles. Note: As an example, if the SPI transmit frequency is set to 12 MHz in the master mode, DSAMPLE should be set to a value of 2
27-10	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
9-0	SCR	R/W	0h	Serial clock divider: This is used to generate the transmit and receive bit rate of the SPI. The SPI bit rate is (SPI's functional clock frequency)/((SCR+1)*2). SCR is a value from 0-1023. 0h = Smallest value 3FFh = Highest possible value

23.7.12 IFLS Register (Offset = 10Ch) [Reset = 00000012h]

IFLS is shown in Table 23-15.

Return to the Summary Table.

The IFLS register is the interrupt FIFO level select register. This register can be used to define the levels at which the TX, RX FIFO interrupt flags are triggered. Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

Table 23-15. IFLS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
5-3	RXIFLSEL	R/W	2h	SPI Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows: 0h = Reserved 1h = RX FIFO >= 1/4 full 2h = RX FIFO >= 1/2 full (default) 3h = RX FIFO >= 3/4 full 4h = Reserved 5h = RX FIFO is full 6h = Reserved 7h = Trigger when RX FIFO contains >= 1 byte
2-0	TXIFLSEL	R/W	2h	SPI Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows: 0h = Reserved 1h = TX FIFO <= 3/4 empty 2h = TX FIFO <= 1/2 empty (default) 3h = TX FIFO <= 1/4 empty 4h = Reserved 5h = TX FIFO is empty 6h = Reserved 7h = Trigger when TX FIFO has >= 1 byte free

23.7.13 STAT Register (Offset = 110h) [Reset = 0000000Fh]

STAT is shown in Table 23-16.

Return to the Summary Table.

Status Register

Table 23-16. STAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
4	BUSY	R	0h	Busy 0h = SPI is in idle mode. 1h = SPI is currently transmitting and/or receiving data, or transmit FIFO is not empty.
3	RNF	R	1h	Receive FIFO not full 0h = Receive FIFO is full. 1h = Receive FIFO is not full.
2	RFE	R	1h	Receive FIFO empty. 0h = Receive FIFO is not empty. 1h = Receive FIFO is empty.
1	TNF	R	1h	Transmit FIFO not full 0h = Transmit FIFO is full. 1h = Transmit FIFO is not full.
0	TFE	R	1h	Transmit FIFO empty. 0h = Transmit FIFO is not empty. 1h = Transmit FIFO is empty.

23.7.14 CLKDIV2 Register (Offset = 114h) [Reset = 00000000h]

CLKDIV2 is shown in Table 23-17.

Return to the Summary Table.

This register is used to specify a divide ratio of the SPI functional clock.

Table 23-17. CLKDIV2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
2-0	RATIO	R/W	0h	Selects divide ratio of module clock 0h = Do not divide clock source 1h = Divide clock source by 2 2h = Divide clock source by 3 3h = Divide clock source by 4 4h = Divide clock source by 5 5h = Divide clock source by 6 6h = Divide clock source by 7 7h = Divide clock source by 8

23.7.15 DMACR Register (Offset = 118h) [Reset = 00000000h]

DMACR is shown in Table 23-18.

Return to the Summary Table.

DMA Control Register

Table 23-18. DMACR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Software should not rely on the value of a reserved. Writing any other value than the reset value may result in undefined behavior.
1	TXDMAE	R/W	0h	Transmit FIFO DMA enable when set.
0	RXDMAE	R/W	0h	Receive FIFO DMA enable when set.

23.7.16 RXDATA Register (Offset = 130h) [Reset = 00000000h]

RXDATA is shown in Table 23-19.

Return to the Summary Table.

RXDATA Register. Reading this register returns value in the RX FIFO pointed by the current FIFO read pointer. If the RX FIFO is empty, the last read value is returned. Writing has not effect and is ignored.

Table 23-19. RXDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Received Data

23.7.17 TXDATA Register (Offset = 140h) [Reset = 00000000h]

TXDATA is shown in Table 23-20.

Return to the Summary Table.

TXDATA Register. Writing into this register puts the data into the TX FIFO. Reading this register returns the last written value, pointed by the current FIFO write pointer.

Table 23-20. TXDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATA	R/W	0h	Transmit Data

Chapter 24

Inter-Integrated Circuit (I^2C)

This chapter describes the Inter-Integrated Circuit interface.

24.1 Introduction....1906
24.2 Block Diagram....1906
24.3 Functional Description....1907
24.4 Initialization and Configuration....1918
24.5 I2C Registers....1919

24.1 Introduction

The I2C bus provides bidirectional data transfer through a 2-wire design, a serial data line (SDA) and a serial clock line (SCL), and interfaces to external I2C devices such as serial memory (RAM and ROM), networking devices, LCDs, tone generators, and so on. The I2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The CC13x4x10 and CC26x4x10 device platform includes one I2C module, which provides the ability to interact (both transmit and receive) with other I2C devices on the bus.

The CC13x4x10 and CC26x4x10 device platform includes one I2C module with the following features:

- Devices on the I2C bus can be designated as either a master or a slave:
  - Supports both transmitting and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I2C modes:
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Two transmission speeds: standard (100 kbps) and fast (400 kbps)
- Master and slave interrupt generation:
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error).
  - Slave generates interrupts when data has been transferred or requested by a master or when a Start or Stop condition is detected.
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

24.2 Block Diagram

Figure 24-1 shows the I2C block diagram.

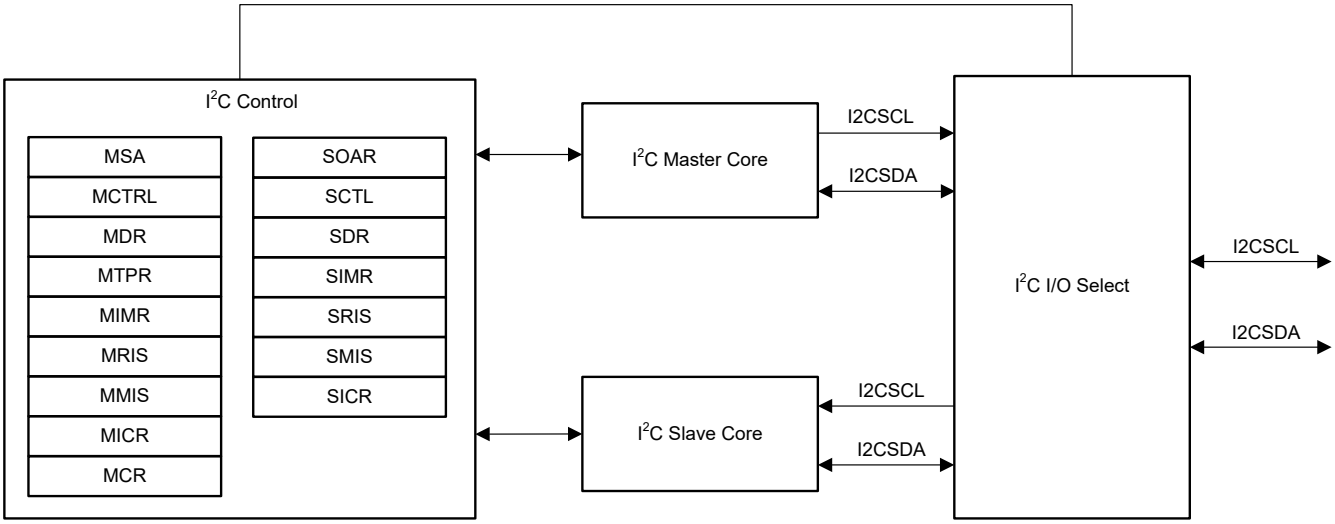


Figure 24-1. I2C Block Diagram

24.3 Functional Description

The I2C module is comprised of both master and slave functions. For proper operation, the SDA pin must be configured as an open-drain signal. Figure 24-2 shows a typical I2C bus configuration.

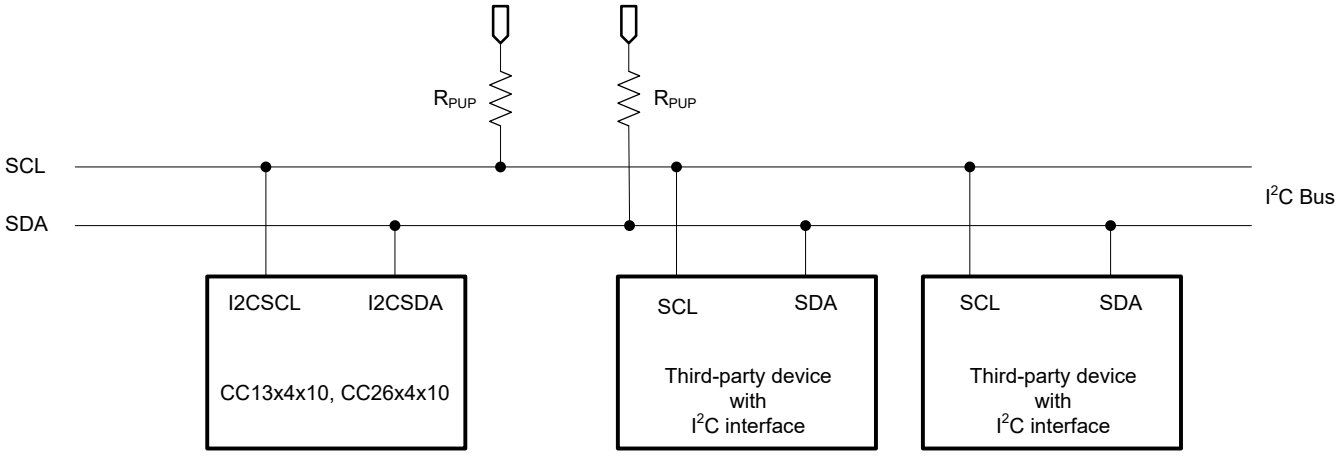


Figure 24-2. I2C Bus Configuration

24.3.1 I2C Bus Functional Overview

The I2C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on the CC13x4x10 and CC26x4x10 controllers. SDA is the bidirectional serial data line and SCL line is the bidirectional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I2C bus is 9 bits long, consisting of 8 data bits and 1 acknowledge bit. The number of bytes per transfer (defined as the time between a valid Start and Stop condition, described in Section 24.3.1.1) is unrestricted, an acknowledge bit must follow each byte, and data must be transferred by the MSB first. When a receiver cannot receive another complete byte, the receiver can hold the clock line SCL low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

24.3.1.1 Start and Stop Conditions

The protocol of the I2C bus defines two states to begin and end a transaction: Start and Stop. A high-to-low transition on the SDA line while the SCL is high is defined as a Start condition, and a low-to-high transition on the SDA line while the SCL line is high is defined as a Stop condition. The bus is considered busy after a Start condition and free after a Stop condition (see Figure 24-3).

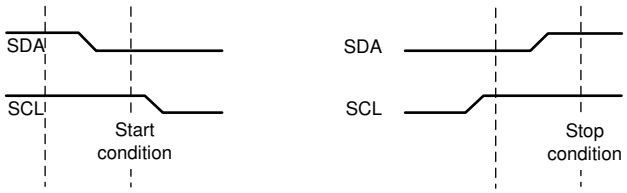


Figure 24-3. Start and Stop Conditions

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a Repeated Start condition. To generate a single transmit cycle, the I2C Master Slave Address I2C:MSA register is written with the desired address, the R/S bit is cleared, and the control register, I2C:MCTRL, is written with ACK = X (0 or 1), STOP = 1, START = 1, and RUN = 1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data is readable from the I2C Master Data I2C:MDR register. When the I2C module operates in master receiver mode, the ACK bit is normally set, causing the I2C bus controller to transmit an acknowledge automatically after each byte. When the I2C bus controller requires no further data transmission from the slave transmitter, the ACK bit must be cleared.

When operating in slave mode, 2 bits in the I2C Slave Raw Interrupt Status I2C:SRIS register indicate detection of Start and Stop conditions on the bus, while 2 bits in the I2C Slave Masked Interrupt Status I2C:SMIS register allow promotion of Start and Stop conditions to controller interrupts (when interrupts are enabled).

24.3.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 24-4. After the Start condition, a slave address is transmitted. This address is 7 bits long followed by an eighth bit, which is a data direction bit (the R/S bit in the I2C:MSA register). If the RS bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a Stop condition generated by the master; however, a master can initiate communications with another device on the bus, by generating a Repeated Start condition and addressing another slave without first generating a Stop condition. Various combinations of receive and transmit formats are then possible within a single transfer.

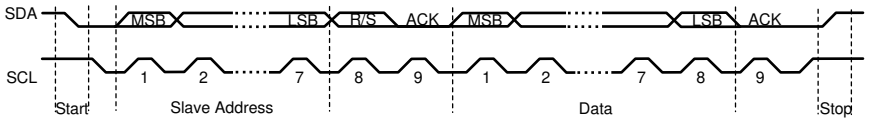


Figure 24-4. Complete Data Transfer with a 7-Bit Address

The first 7 bits of the first byte comprise the slave address (see Figure 24-5). The eighth bit determines the direction of the message. A 0 in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a 1 in this position means that the master receives data from the slave.

Figure 24-5. R/S Bit in First Byte

24.3.1.3 Data Validity

The SDA line must contain stable data during the high period of the clock, and the data line can change only when SCL is low (see Figure 24-6).

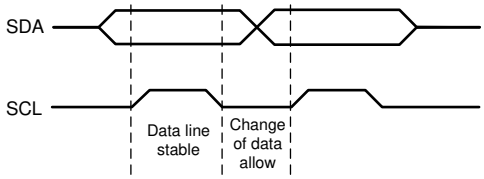


Figure 24-6. Data Validity During Bit Transfer on the I2C Bus

24.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle generated by the master. During the acknowledge cycle, the transmitter (master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted by the receiver during the acknowledge cycle must comply with the data validity requirements described in Section 24.3.1.3.

When a slave receiver does not acknowledge the slave address, the slave must leave SDA high so that the master can generate a Stop condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to let the master generate a Stop or a Repeated Start condition.

24.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. Two or more masters can generate a Start condition within minimum hold time of the Start condition. In these situations, an arbitration scheme occurs on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place 1 (high) on SDA while another master transmits 0 (low) switches off its data output stage, and waits until the bus is idle again.

Arbitration can occur over several bits. The first stage of arbitration is a comparison of address bits; if both masters are trying to address the same device, arbitration continues to the comparison of data bits.

24.3.2 Available Speed Modes

The I2C bus can run in either standard mode (100 kbps) or fast mode (400 kbps). The selected mode must match the speed of the other I2C devices on the bus.

24.3.2.1 Standard and Fast Modes

Standard and fast modes are selected using a value in the I2C Master Timer Period I2C:MTPR register that results in an SCL frequency of 100 kbps for standard mode, or 400 kbps for fast mode.

The I2C clock rate is determined by the parameters CLK_PRD, TIMER_PRD, SCL_LP, and SCL_HP where:

- CLK_PRD is the system clock period.
- TIMER_PRD is the programmed value in the I2C:MTPR register.
- SCL_LP is the low phase of SCL (fixed at 6).
- SCL_HP is the high phase of SCL (fixed at 4).

The I2C clock period is calculated as follows:

$$SCL\_PERIOD = 2 \times (1 + TIMER\_PRD) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD \quad (10)$$

For example:
CLK_PRD = 50 ns
TIMER_PRD = 2
SCL_LP = 6
SCL_HP = 4

yields a SCL frequency of:
1 / SCL_PERIOD = 333 kHz

Table 24-1 lists examples of the timer periods used to generate both standard and fast-mode SCL frequencies, based on various system clock frequencies.

Table 24-1. Examples of I2C Master Timer Period versus Speed Mode

System Clock (MHz)	Timer Period	Standard Mode (kbps)	Timer Period	Fast Mode (kbps)
4	0x01	100	–	–
8	0x03	100	0x01	–
16	0x07	100	0x01	400

24.3.3 Interrupts

The I^2C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost
- Master transaction error
- Master bus time-out
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I^2C master and I^2C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller (INTC).

24.3.3.1 I^2C Master Interrupts

The I^2C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I^2C master interrupt, software must set the IM bit in the I^2C Master Interrupt Mask register, I2C:MIMR. When an interrupt condition is met, software must check the I^2C Master Control and Status register (I2C:MSTAT) ERR and ARBLST bits to verify that an error did not occur during the last transaction, and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction was not acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by setting the IC bit in the I^2C Master Interrupt Clear register (I2C:MICR) to 1.

If the application does not require the use of interrupts, the raw interrupt status is always visible through the I^2C Master Raw Interrupt Status register (I2C:MRIS).

24.3.3.2 I^2C Slave Interrupts

The slave module can generate an interrupt when data is received or requested. This interrupt is enabled by setting the in the I^2C Slave Interrupt Mask register (I2C:SIMR). Software determines whether the module must write (transmit) or read (receive) data from the I^2C Slave Data register (I2C:SDR) DATAIM bit, by checking the RREQ and TREQ bits of the I^2C Slave Control and Status register (I2C:SSTAT). If the slave module is in receive mode and the first byte of a transfer is received, the FBR and RREQ bits are set. The interrupt is cleared by setting the I^2C Slave Interrupt Clear register (I2C:SICR) DATAIC bit.

In addition, the slave module generates an interrupt when a Start and a Stop condition is detected. These interrupts are enabled by setting the I2C:SIMR register STARTIM and STOPIM bits; these interrupts are cleared by setting the I2C:SICR register STOPIC and STARTIC bits to 1.

If the application does not require the use of interrupts, the raw interrupt status is always visible through the I^2C Slave Raw Interrupt Status register (I2C:SRIS).

24.3.4 Loopback Operation

The I^2C modules can be placed into an internal-loopback mode for diagnostic or debug work by setting the I^2C Master Configuration register (I2C:MCR) LPBK bit. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

24.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I^2C transfer types in both master and slave mode. To do this, the SDA and SCL signal configuration must be done in the IOC:IOCFG register.

24.3.5.1 I^2C Master Command Sequences

Figure 24-7 through Figure 24-12 show the command sequences available for the I^2C master.

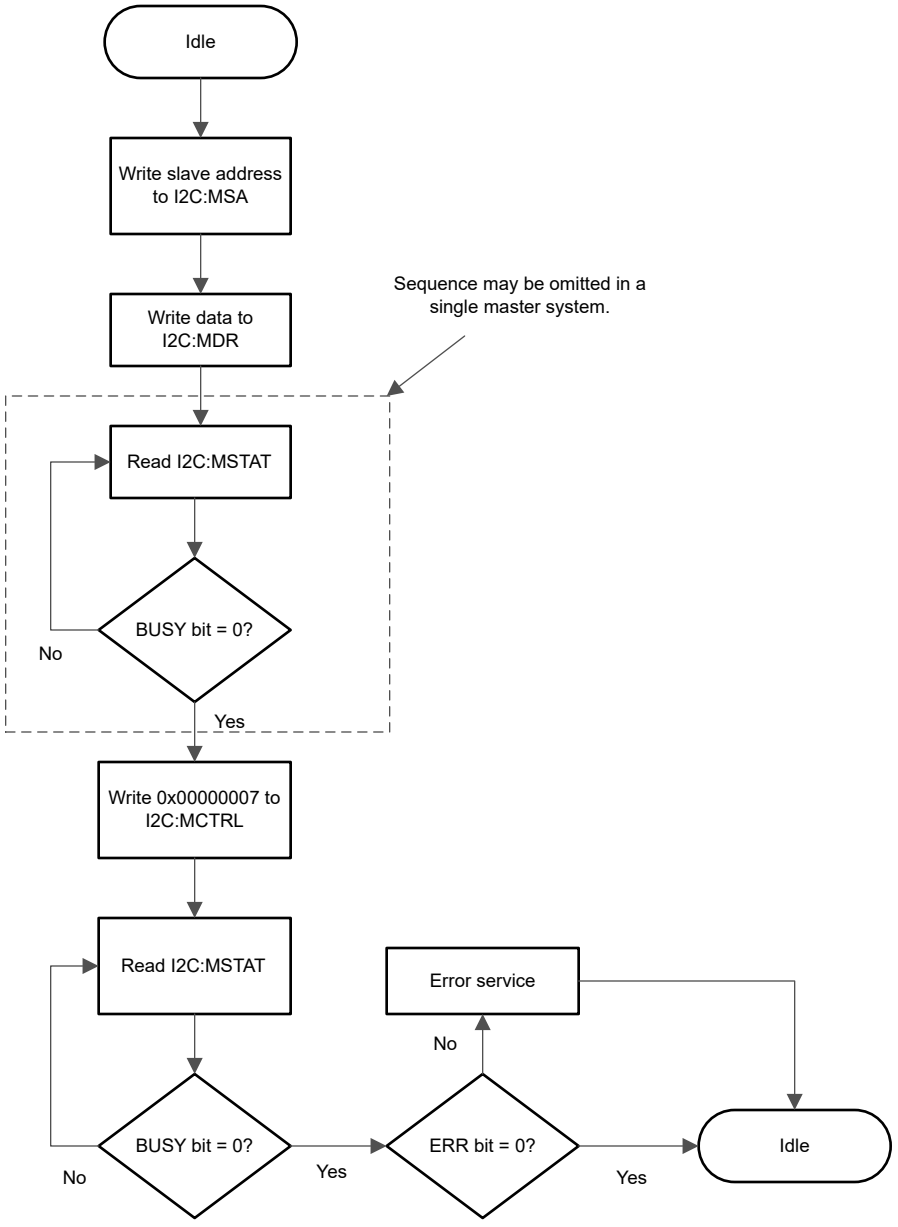


Figure 24-7. Master Single Transmit

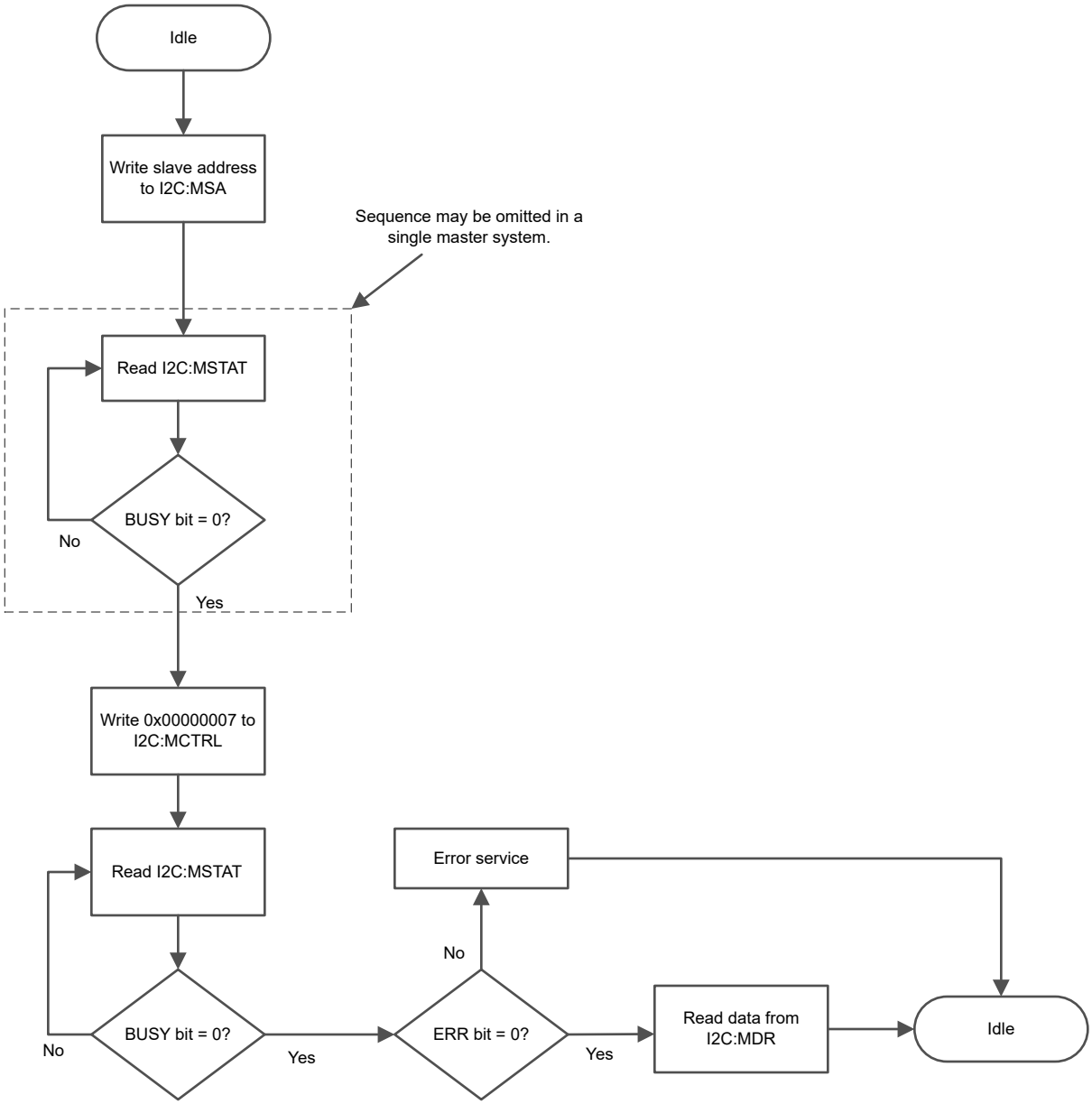


Figure 24-8. Master Single Receive

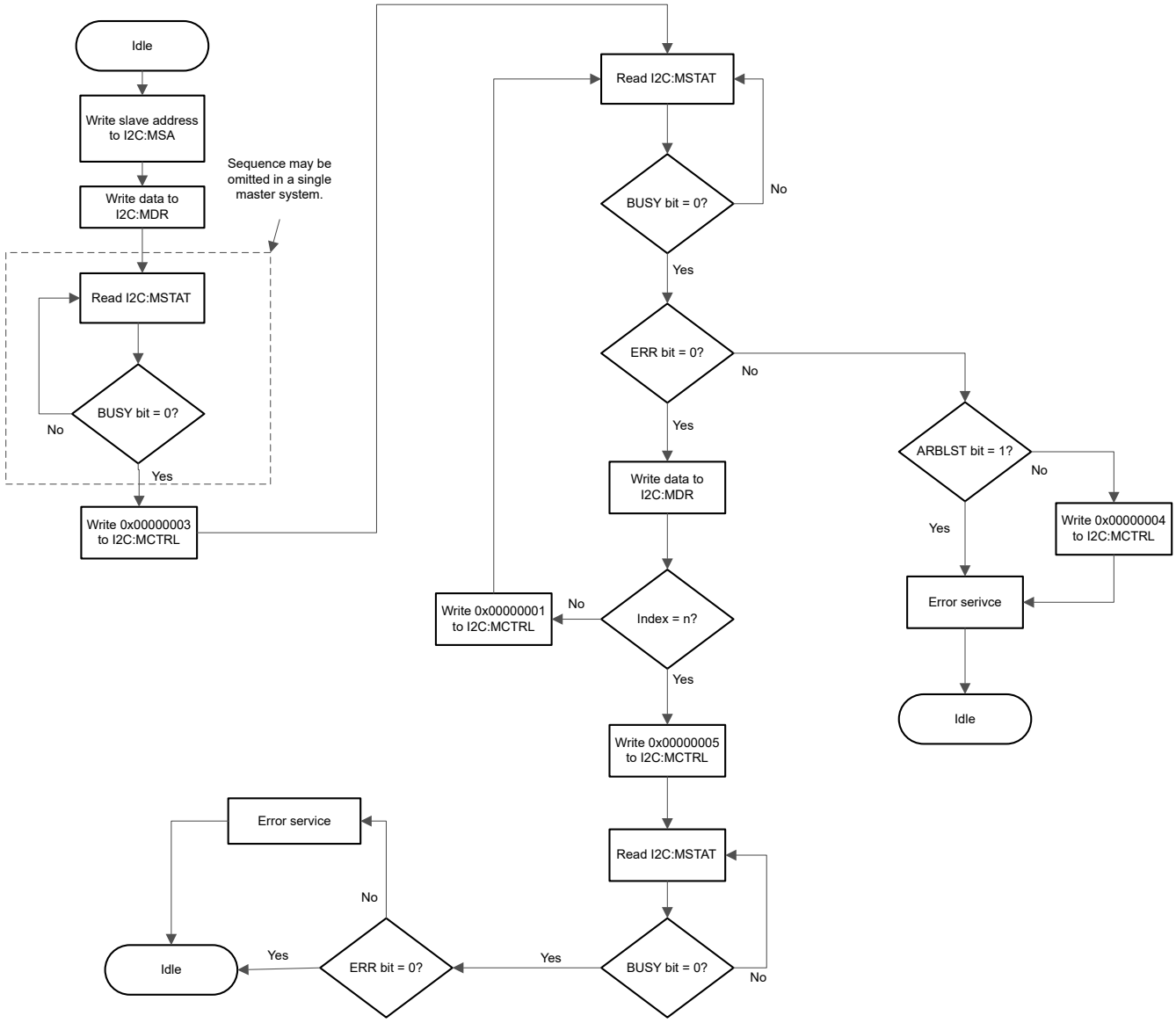


Figure 24-9. Master Transmit with Repeated Start Condition

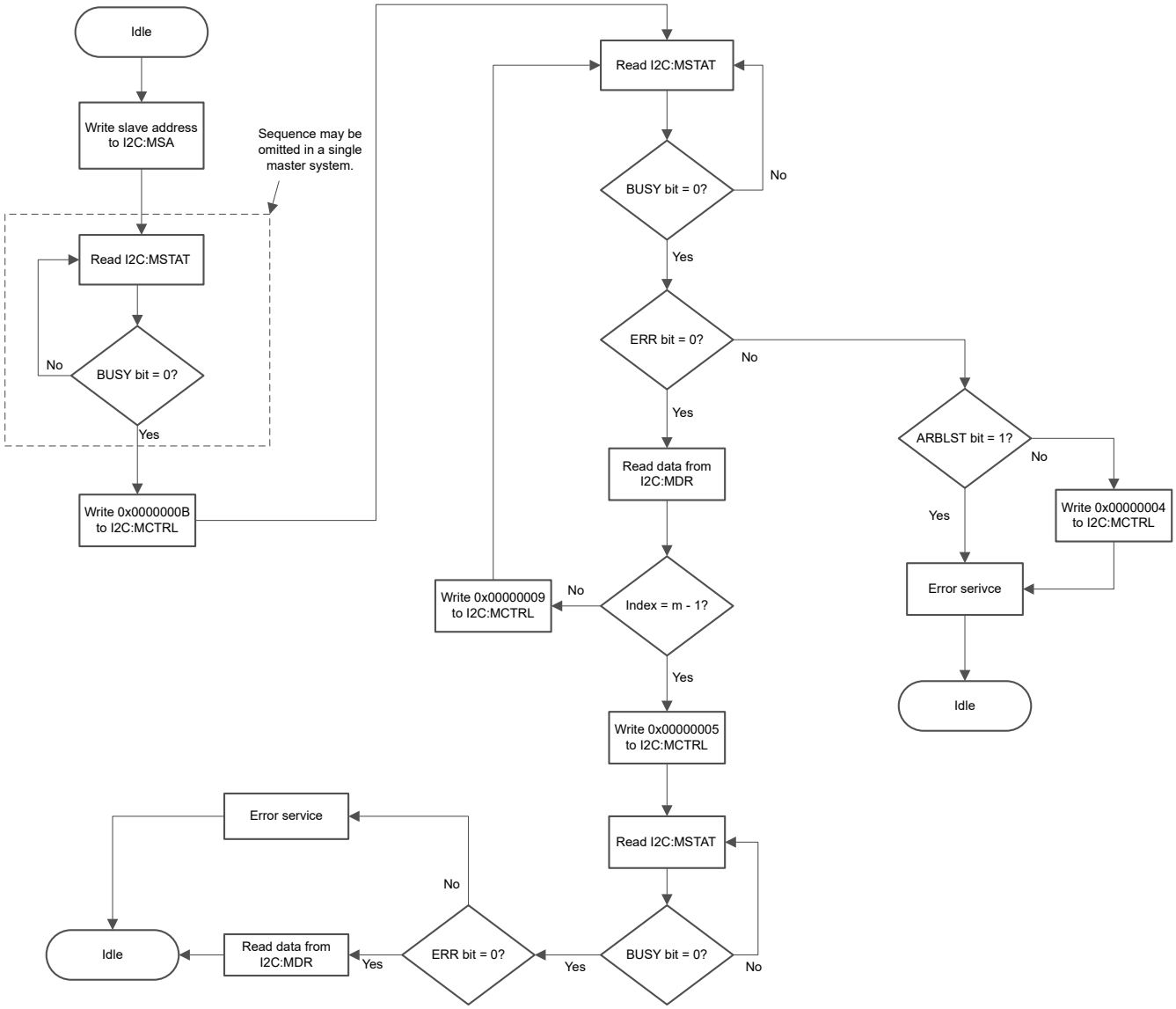


Figure 24-10. Master Receive with Repeated Start Condition

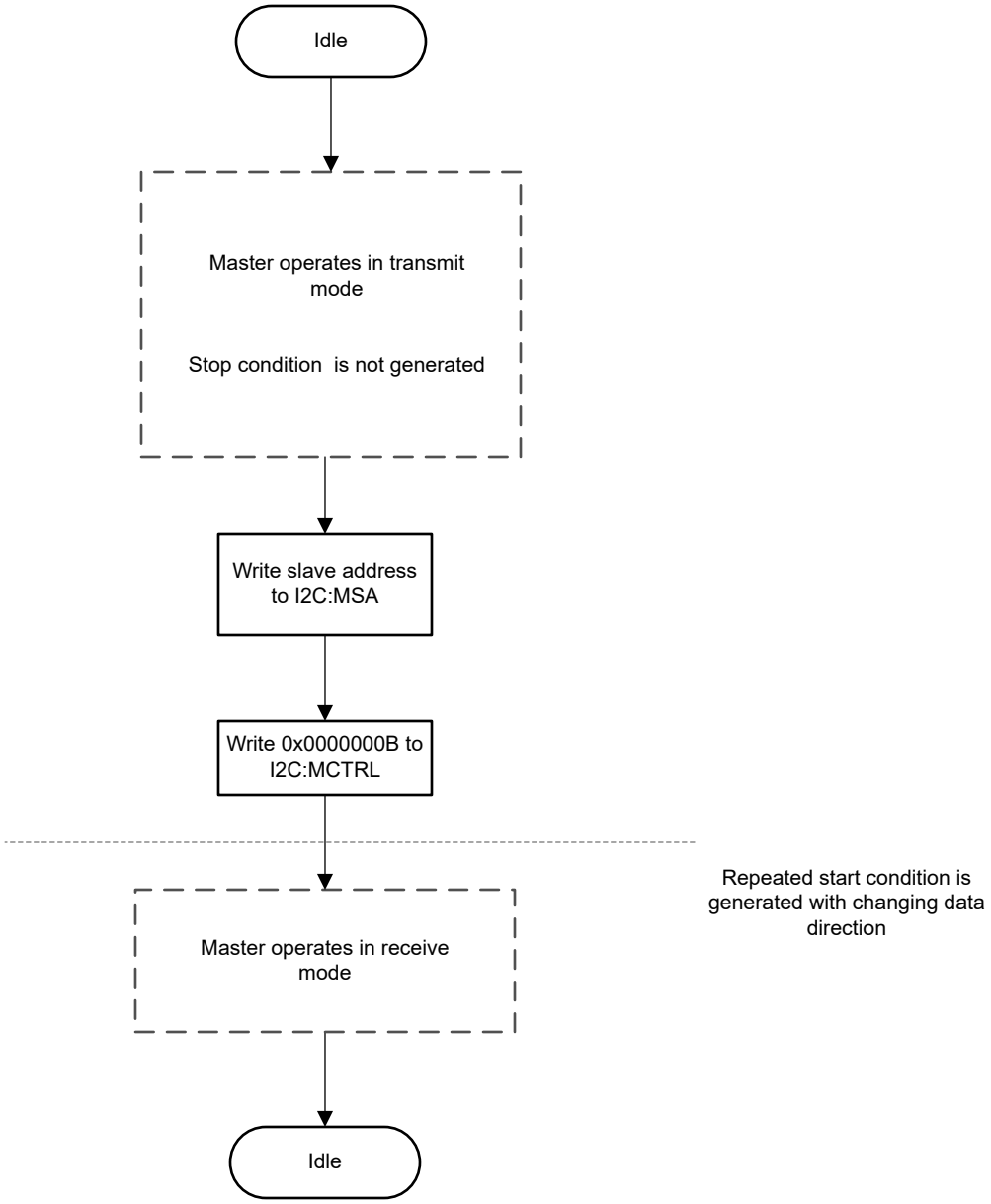


Figure 24-11. Master Receive with Repeated Start after Transmit with Repeated Start Condition

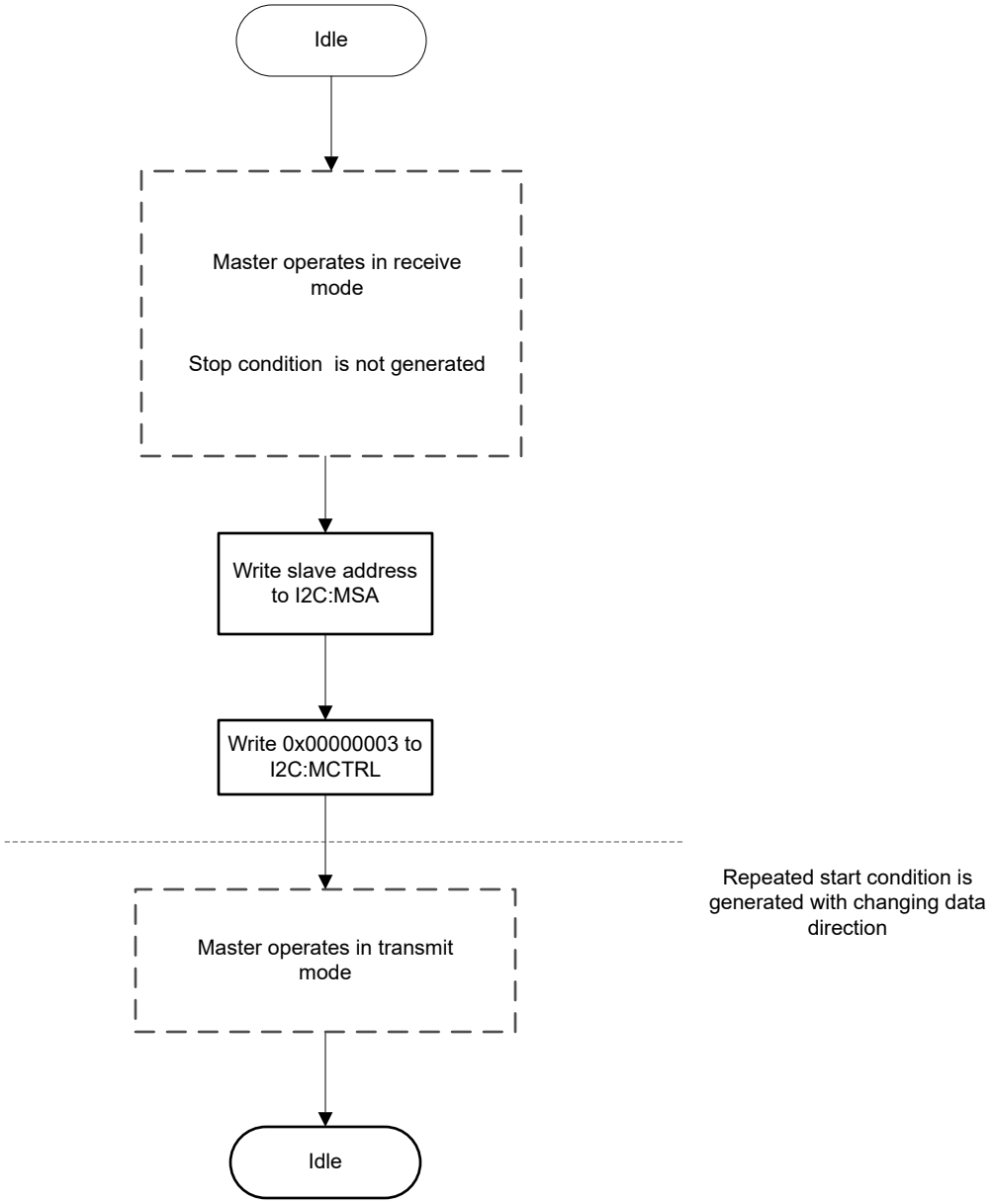


Figure 24-12. Master Transmit with Repeated Start after Receive with Repeated Start Condition

24.3.5.2 I^2C Slave Command Sequences

Figure 24-13 shows the command sequence available for the I^2C slave.

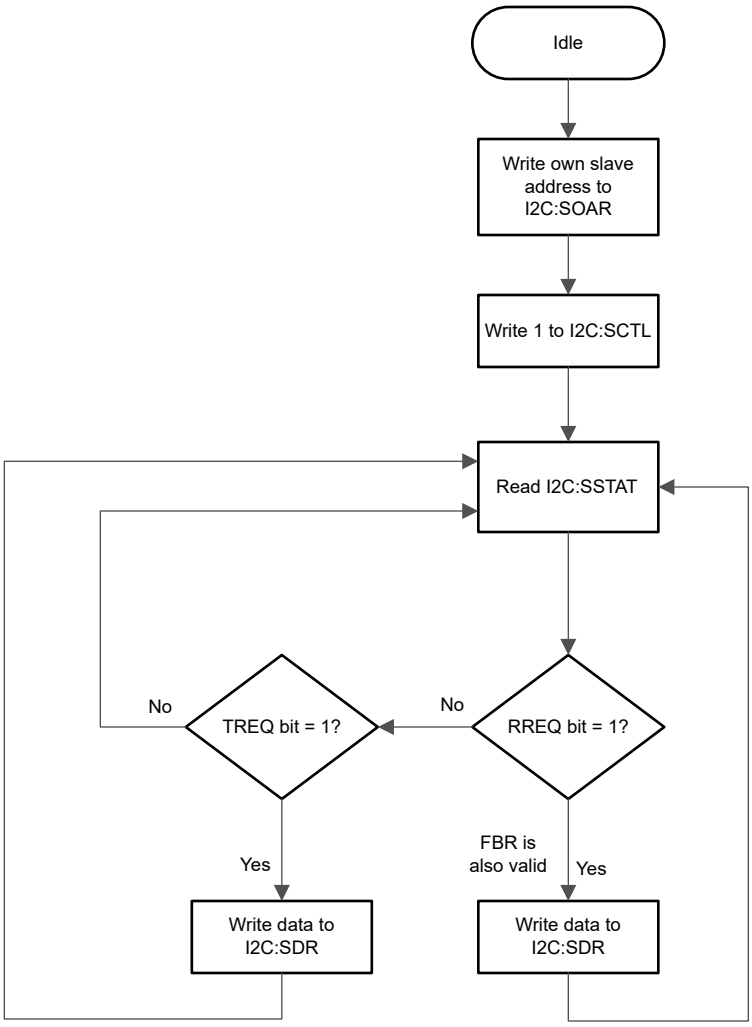


Figure 24-13. Slave Command Sequence

24.4 Initialization and Configuration

The following described the necessary steps to configure the I^2C module to transmit a single byte as a master, assuming that the system clock is 24 MHz.

TI recommends using the I^2C driver in the SimpleLink™ CC13xx and CC26xx software development kit (SDK)

1. Enable the serial power domain and enable the I^2C module in PRCM by writing to PRCM:I2CCLKGR, PRCM:I2CCLKGS, and PRCM:I2CCLKGDS register, and load the setting to the clock controller by writing to the PRCM:CLKLOADCTL register
2. Configure the IOC module to route the SDA and SCL signals from I/Os to the I^2C module
3. Initialize the I^2C master by writing the I2C:MCR register with a value of 0x00000010
4. Set the desired SCL clock speed of 100 kbps by writing the I2C:MTPR register with the correct value. The value written to the I2C:MTPR register represents the number of system clock periods in one SCL clock period. The TPR value is determined by Equation 11, Equation 12, and Equation 13.

   $$TPR = [PERDMACLK / (2 \times (SCL\_LP + SCL\_HP) \times SCL\_CLK)] - 1 \quad (11)$$

   $$TPR = [24\ MHz / (2 \times (6 + 4) \times 100000)] - 1 \quad (12)$$

   $$TPR = 11 \quad (13)$$

   Write the I2C:MTPR register with the value of 0x0000000B
5. Specify the slave address of the master and that the next operation is a transmit by writing the I2C:MSA register with a value of 0x00000076, which sets the slave address to 0x3B
6. Place data (byte) to be transmitted in the data register by writing the I2C:MDR register with the desired data
7. Initiate a single-byte transmit of the data from master to slave by writing the I2C:MCTRL register with a value of 0x00000007 (Stop, Start, Run)
8. Wait until the transmission completes by polling the I2C:MSTAT BUSBSY register bit until it is cleared
9. Check the I2C:MSTAT ERR register bit to confirm the transmit was acknowledged

24.5 I2C Registers

Table 24-2 lists the memory-mapped registers for the I2C registers. All register offset addresses not listed in Table 24-2 should be considered as reserved locations and the register contents should not be modified.

Table 24-2. I2C Registers

Offset	Acronym	Register Name	Section
0h	SOAR	Slave Own Address	Section 24.5.1
4h	SSTAT	Slave Status	Section 24.5.2
4h	SCTL	Slave Control	Section 24.5.3
8h	SDR	Slave Data	Section 24.5.4
Ch	SIMR	Slave Interrupt Mask	Section 24.5.5
10h	SRIS	Slave Raw Interrupt Status	Section 24.5.6
14h	SMIS	Slave Masked Interrupt Status	Section 24.5.7
18h	SICR	Slave Interrupt Clear	Section 24.5.8
800h	MSA	Master Slave Address	Section 24.5.9
804h	MSTAT	Master Status	Section 24.5.10
804h	MCTRL	Master Control	Section 24.5.11
808h	MDR	Master Data	Section 24.5.12
80Ch	MTPR	I2C Master Timer Period	Section 24.5.13
810h	MIMR	Master Interrupt Mask	Section 24.5.14
814h	MRIS	Master Raw Interrupt Status	Section 24.5.15
818h	MMIS	Master Masked Interrupt Status	Section 24.5.16
81Ch	MICR	Master Interrupt Clear	Section 24.5.17
820h	MCR	Master Configuration	Section 24.5.18

Complex bit access types are encoded to fit into small table cells. Table 24-3 shows the codes that are used for access types in this section.

Table 24-3. I2C Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

24.5.1 SOAR Register (Offset = 0h) [Reset = 00000000h]

SOAR is shown in Table 24-4.

Return to the Summary Table.

Slave Own Address
This register consists of seven address bits that identify this I2C device on the I2C bus.

Table 24-4. SOAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6-0	OAR	R/W	0h	I2C slave own address This field specifies bits a6 through a0 of the slave address.

24.5.2 SSTAT Register (Offset = 4h) [Reset = 00000000h]

SSTAT is shown in Table 24-5.

Return to the Summary Table.

Slave Status
Note: This register shares address with SCTL, meaning that this register functions as a control register when written, and a status register when read.

Table 24-5. SSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	FBR	R	0h	First byte received 0: The first byte has not been received. 1: The first byte following the slave's own address has been received. This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the SDR register. Note: This bit is not used for slave transmit operations.
1	TREQ	R	0h	Transmit request 0: No outstanding transmit request. 1: The I2C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the SDR register.
0	RREQ	R	0h	Receive request 0: No outstanding receive data 1: The I2C controller has outstanding receive data from the I2C master and is using clock stretching to delay the master until data has been read from the SDR register.

24.5.3 SCTL Register (Offset = 4h) [Reset = 00000000h]

SCTL is shown in Table 24-6.

Return to the Summary Table.

Slave Control
Note: This register shares address with SSTAT, meaning that this register functions as a control register when written, and a status register when read.

Table 24-6. SCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	W	0h	Software should not rely on the value of a reserved field. Writing any other value may result in undefined behavior.
0	DA	W	0h	Device active 0: Disables the I2C slave operation 1: Enables the I2C slave operation

24.5.4 SDR Register (Offset = 8h) [Reset = 00000000h]

SDR is shown in Table 24-7.

Return to the Summary Table.

Slave Data
This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

Table 24-7. SDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	DATA	R/W	0h	Data for transfer This field contains the data for transfer during a slave receive or transmit operation. When written the register data is used as transmit data. When read, this register returns the last data received. Data is stored until next update, either by a system write for transmit or by an external master for receive.

24.5.5 SIMR Register (Offset = Ch) [Reset = 00000000h]

SIMR is shown in Table 24-8.

Return to the Summary Table.

Slave Interrupt Mask
This register controls whether a raw interrupt is promoted to a controller interrupt.

Table 24-8. SIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	STOPIM	R/W	0h	Stop condition interrupt mask 0: The SRIS.STOPRIS interrupt is suppressed and not sent to the interrupt controller. 1: The SRIS.STOPRIS interrupt is enabled and sent to the interrupt controller. 0h = Disable Interrupt 1h = Enable Interrupt
1	STARTIM	R/W	0h	Start condition interrupt mask 0: The SRIS.STARTRIS interrupt is suppressed and not sent to the interrupt controller. 1: The SRIS.STARTRIS interrupt is enabled and sent to the interrupt controller. 0h = Disable Interrupt 1h = Enable Interrupt
0	DATAIM	R/W	0h	Data interrupt mask 0: The SRIS.DATARIS interrupt is suppressed and not sent to the interrupt controller. 1: The SRIS.DATARIS interrupt is enabled and sent to the interrupt controller.

24.5.6 SRIS Register (Offset = 10h) [Reset = 00000000h]

SRIS is shown in Table 24-9.

Return to the Summary Table.

Slave Raw Interrupt Status
This register shows the unmasked interrupt status.

Table 24-9. SRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	STOPRIS	R	0h	Stop condition raw interrupt status 0: No interrupt 1: A Stop condition interrupt is pending. This bit is cleared by writing a 1 to SICR.STOPIC.
1	STARTRIS	R	0h	Start condition raw interrupt status 0: No interrupt 1: A Start condition interrupt is pending. This bit is cleared by writing a 1 to SICR.STARTIC.
0	DATARIS	R	0h	Data raw interrupt status 0: No interrupt 1: A data received or data requested interrupt is pending. This bit is cleared by writing a 1 to the SICR.DATAIC.

24.5.7 SMIS Register (Offset = 14h) [Reset = 00000000h]

SMIS is shown in Table 24-10.

Return to the Summary Table.

Slave Masked Interrupt Status
This register show which interrupt is active (based on result from SRIS and SIMR).

Table 24-10. SMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	STOPMIS	R	0h	Stop condition masked interrupt status 0: An interrupt has not occurred or is masked/disabled. 1: An unmasked Stop condition interrupt is pending. This bit is cleared by writing a 1 to the SICR.STOPIC.
1	STARTMIS	R	0h	Start condition masked interrupt status 0: An interrupt has not occurred or is masked/disabled. 1: An unmasked Start condition interrupt is pending. This bit is cleared by writing a 1 to the SICR.STARTIC.
0	DATAMIS	R	0h	Data masked interrupt status 0: An interrupt has not occurred or is masked/disabled. 1: An unmasked data received or data requested interrupt is pending. This bit is cleared by writing a 1 to the SICR.DATAIC.

24.5.8 SICR Register (Offset = 18h) [Reset = 00000000h]

SICR is shown in Table 24-11.

Return to the Summary Table.

Slave Interrupt Clear
This register clears the raw interrupt SRIS.

Table 24-11. SICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	STOPIC	W	0h	Stop condition interrupt clear Writing 1 to this bit clears SRIS.STOPRIS and SMIS.STOPMIS.
1	STARTIC	W	0h	Start condition interrupt clear Writing 1 to this bit clears SRIS.STARTRIS SMIS.STARTMIS.
0	DATAIC	W	0h	Data interrupt clear Writing 1 to this bit clears SRIS.DATARIS SMIS.DATAMIS.

24.5.9 MSA Register (Offset = 800h) [Reset = 00000000h]

MSA is shown in Table 24-12.

Return to the Summary Table.

Master Slave Address
This register contains seven address bits of the slave to be accessed by the master (a6-a0), and an RS bit determining if the next operation is a receive or transmit.

Table 24-12. MSA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-1	SA	R/W	0h	I2C master slave address Defines which slave is addressed for the transaction in master mode
0	RS	R/W	0h	Receive or Send This bit-field specifies if the next operation is a receive (high) or a transmit/send (low) from the addressed slave SA. 0h = Transmit/send data to slave 1h = Receive data from slave

24.5.10 MSTAT Register (Offset = 804h) [Reset = 00000020h]

MSTAT is shown in Table 24-13.

Return to the Summary Table.

Master Status

Table 24-13. MSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	BUSBSY	R	0h	Bus busy 0: The I2C bus is idle. 1: The I2C bus is busy. The bit changes based on the MCTRL.START and MCTRL.STOP conditions.
5	IDLE	R	1h	I2C idle 0: The I2C controller is not idle. 1: The I2C controller is idle.
4	ARBLST	R	0h	Arbitration lost 0: The I2C controller won arbitration. 1: The I2C controller lost arbitration.
3	DATACK_N	R	0h	Data Was Not Acknowledge 0: The transmitted data was acknowledged. 1: The transmitted data was not acknowledged.
2	ADRACK_N	R	0h	Address Was Not Acknowledge 0: The transmitted address was acknowledged. 1: The transmitted address was not acknowledged.
1	ERR	R	0h	Error 0: No error was detected on the last operation. 1: An error occurred on the last operation.
0	BUSY	R	0h	I2C busy 0: The controller is idle. 1: The controller is busy. When this bit-field is set, the other status bits are not valid. Note: The I2C controller requires four SYSBUS clock cycles to assert the BUSY status after I2C master operation has been initiated through MCTRL register. Hence after programming MCTRL register, application is requested to wait for four SYSBUS clock cycles before issuing a controller status inquiry through MSTAT register. Any prior inquiry would result in wrong status being reported.

24.5.11 MCTRL Register (Offset = 804h) [Reset = 00000000h]

MCTRL is shown in Table 24-14.

Return to the Summary Table.

Master Control
This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I2C bus controller as stated in MSTAT. When written, the control register configures the I2C controller operation.
To generate a single transmit cycle, the I2C Master Slave Address (MSA) register is written with the desired address, the MSA.RS bit is cleared, and this register is written with
* ACK=X (0 or 1),
* STOP=1,
* START=1,
* RUN=1
to perform the operation and stop.
When the operation is completed (or aborted due an error), an interrupt becomes active and the data may be read from the MDR register.

Table 24-14. MCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	ACK	W	0h	Data acknowledge enable 0: The received data byte is not acknowledged automatically by the master. 1: The received data byte is acknowledged automatically by the master. This bit-field must be cleared when the I2C bus controller requires no further data to be transmitted from the slave transmitter. 0h = Disable acknowledge 1h = Enable acknowledge
2	STOP	W	0h	This bit-field determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. 0: The controller does not generate the Stop condition. 1: The controller generates the Stop condition. 0h = Disable STOP 1h = Enable STOP
1	START	W	0h	This bit-field generates the Start or Repeated Start condition. 0: The controller does not generate the Start condition. 1: The controller generates the Start condition. 0h = Disable START 1h = Enable START
0	RUN	W	0h	I2C master enable 0: The master is disabled. 1: The master is enabled to transmit or receive data. 0h = Disable Master 1h = Enable Master

24.5.12 MDR Register (Offset = 808h) [Reset = 00000000h]

MDR is shown in Table 24-15.

Return to the Summary Table.

Master Data
This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state.

Table 24-15. MDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	DATA	R/W	0h	When Read: Last RX Data is returned When Written: Data is transferred during TX transaction

24.5.13 MTPR Register (Offset = 80Ch) [Reset = 00000001h]

MTPR is shown in Table 24-16.

Return to the Summary Table.

I2C Master Timer Period
This register specifies the period of the SCL clock.

Table 24-16. MTPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	TPR_7	R/W	0h	Must be set to 0 to set TPR. If set to 1, a write to TPR will be ignored.
6-0	TPR	R/W	1h	SCL clock period This field specifies the period of the SCL clock. SCL_PRD = 2*(1+TPR)*(SCL_LP + SCL_HP)*CLK_PRD where: SCL_PRD is the SCL line period (I2C clock). TPR is the timer period register value (range of 1 to 127) SCL_LP is the SCL low period (fixed at 6). SCL_HP is the SCL high period (fixed at 4). CLK_PRD is the system clock period in ns.

24.5.14 MIMR Register (Offset = 810h) [Reset = 00000000h]

MIMR is shown in Table 24-17.

Return to the Summary Table.

Master Interrupt Mask
This register controls whether a raw interrupt is promoted to a controller interrupt.

Table 24-17. MIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	IM	R/W	0h	Interrupt mask 0: The MRIS.RIS interrupt is suppressed and not sent to the interrupt controller. 1: The master interrupt is sent to the interrupt controller when the MRIS.RIS is set. 0h = Disable Interrupt 1h = Enable Interrupt

24.5.15 MRIS Register (Offset = 814h) [Reset = 00000000h]

MRIS is shown in Table 24-18.

Return to the Summary Table.

Master Raw Interrupt Status
This register show the unmasked interrupt status.

Table 24-18. MRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	RIS	R	0h	Raw interrupt status 0: No interrupt 1: A master interrupt is pending. This bit is cleared by writing 1 to the MICR.IC bit .

24.5.16 MMIS Register (Offset = 818h) [Reset = 00000000h]

MMIS is shown in Table 24-19.

Return to the Summary Table.

Master Masked Interrupt Status
This register show which interrupt is active (based on result from MRIS and MIMR).

Table 24-19. MMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	MIS	R	0h	Masked interrupt status 0: An interrupt has not occurred or is masked. 1: A master interrupt is pending. This bit is cleared by writing 1 to the MICR.IC bit .

24.5.17 MICR Register (Offset = 81Ch) [Reset = 00000000h]

MICR is shown in Table 24-20.

Return to the Summary Table.

Master Interrupt Clear
This register clears the raw and masked interrupt.

Table 24-20. MICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	IC	W	0h	Interrupt clear Writing 1 to this bit clears MRIS.RIS and MMIS.MIS . Reading this register returns no meaningful data.

24.5.18 MCR Register (Offset = 820h) [Reset = 00000000h]

MCR is shown in Table 24-21.

Return to the Summary Table.

Master Configuration
This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

Table 24-21. MCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	SFE	R/W	0h	I2C slave function enable 0h = Slave mode is disabled. 1h = Slave mode is enabled.
4	MFE	R/W	0h	I2C master function enable 0h = Master mode is disabled. 1h = Master mode is enabled.
3-1	RESERVED	R	0h	Reserved
0	LPBK	R/W	0h	I2C loopback 0: Normal operation 1: Loopback operation (test mode) 0h = Disable Test Mode 1h = Enable Test Mode

This page intentionally left blank.

Chapter 25

Inter-IC Sound (I^2S)

This chapter describes the Inter-IC Sound (I^2S) module.

25.1 Introduction 1940
25.2 Block Diagram 1941
25.3 Signal Description 1942
25.4 Functional Description 1942
25.5 Memory Interface 1948
25.6 Samplestamp Generator 1951
25.7 Error Detection 1954
25.8 Usage 1954
25.9 I2S Registers 1955

25.1 Introduction

The I^2S module provides a standardized serial interface to transfer audio samples between the CC13x4x10 and CC26x4x10 device platform and the external audio devices (such as a codec, DAC, or ADC).

The I^2S module can also receive pulse density modulation (PDM) signals from devices (such as digital microphones) followed by conversion to pulse code modulation (PCM) in software.

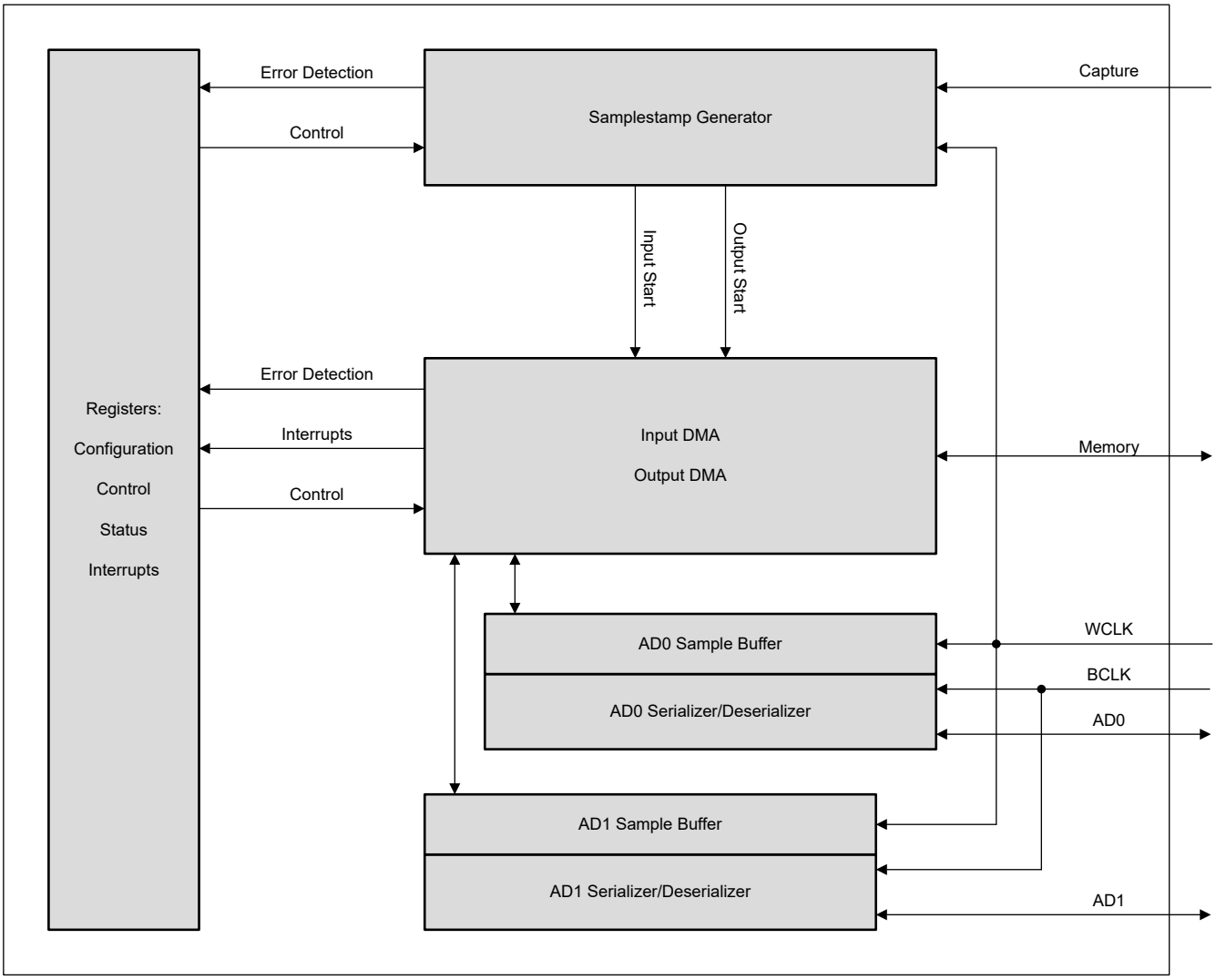
The I^2S module has the following features:

- Audio clock signals are internally generated by the PRCM module or externally by another device.
- One or two data pins, which can be configured independently as input or output
- I^2S, left-justified (LJF), and right-justified (RJF) serial interface formats that support up to two audio channels per data pin
- DSP serial interface format that supports up to eight audio channels per data pin
- Up to 24-bit sample word length, with truncation or zero-padding if not matching:
  - Serial interface: 8 to 24 bits per sample word, most significant bit (MSB) transferred first
  - Memory: 16 or 24 bits per sample word, stored in little-endian byte order
- DMA with double-buffered pointers
- Error detection for DMA and audio clock signal integrity
- A Samplestamp generator that allows maintaining of constant audio latency when audio samples are produced by the I^2S module on one CC13x4x10 or CC26x4x10 device and consumed by the I^2S module on another CC13x4x10 or CC26x4x10 device.

25.2 Block Diagram

Figure 25-1 shows a block diagram of the I^2S module.

Figure 25-1. Simplified I^2S Module Block Diagram



25.3 Signal Description

The serial audio interface consists of two or three clock signals and one or two data signals, depending on how the I^2S module is used. The clock signals can be generated either internally (by the PRCM module of the CC13x4x10 and CC26x4x10 device platform) or externally (by the audio device or another clock source).

Table 25-1 lists details about the serial audio pin interface.

Table 25-1. Serial Audio Pin Interface

CC13x4x10 and CC26x4x10 Pin	Function and Direction	Description
MCLK (optional)	Master clock output	Master clock for sample conversion in the external audio device. This signal is not used internally by the CC13x4x10 or CC26x4x10 device platform, and hence, will never be an input to the CC13x4x10 or CC26x4x10 device platform. Some external audio devices do not require this signal.
BCLK	Bit clock input/output	Bit clock for the WCLK and the AD0 and AD1 signals. This signal is an input when using an external clock source, and it is an output when using the internal clock source.
WCLK	Word clock input/output	Sample framing signal that defines the audio sample frequency and the sample word boundaries in the AD0 and AD1 serial data streams. The WCLK frequency is identical to the audio sample frequency. This signal is an input when using an external clock source, and it is an output when using the internal clock source.
AD0 AD1	Serial data input/output	Serial data signals responsible for transferring audio sample words. Each pin can be configured independently as an input, an output, or it can be unused. All pins use the same interface format (for example, LJF). The AD0 and AD1 pins are hereafter referred to as *ADx pins*.

The CC13x4x10 and CC26x4x10 device platform cannot dynamically place the ADx pins in a tri-state condition. Therefore, TDM mode is supported for ADx input pins where only external audio devices drive these signals, but TDM mode is not supported for ADx output pins.

25.4 Functional Description

25.4.1 Dependencies

Enable system clock to the I^2S module before accessing any I^2S module registers:

- For system CPU run mode: PRCM:I2SCLKGR.CLK_EN = 1 (loaded with PRCM:CLKLOADCTL)
- For system CPU sleep mode: PRCM:I2SCLKGS.CLK_EN = 1 (loaded with PRCM:CLKLOADCTL)
- For system CPU deep-sleep mode: PRCM:I2SCLKGDS.CLK_EN = 1 (loaded with PRCM:CLKLOADCTL)

For more details on the CPU run, sleep and deep-sleep mode, see Table 7-3.

25.4.1.1 System CPU Deep-Sleep Mode

If the system bus is turned off, the I^2S DMA loses access to RAM and Flash.

The system bus is turned off if all of the following conditions are true:

- PRCM:PDCTL1.CPU_ON = 0
- PRCM:PDCTL1.VIMS_MODE = 0
- PRCM:SECDMACLKGDS.DMA_CLK_EN = 0
- PRCM:SECDMACLKGDS.CRYPTO_CLK_EN = 0
- RFCORE does not request access to BUS
- System CPU is in deep-sleep mode (see Table 7-3)

For lowest current consumption, set PRCM:SECDMACLKGDS.DMA_CLK_EN = 1 (loaded with PRCM:CLKLOADCTL) to keep the system bus enabled for the I^2S DMA.

25.4.2 Pin Configuration

The I2S:AIFDIRCFG register configures whether each ADx signal is input, output, or unused.

Each used I^2S signal must be mapped to a physical I/O pin, DIOn, by configuring the corresponding IOC:IOCFGn register.

25.4.3 Serial Format Configuration

The WCLK and ADx signals are updated on one edge of the BCLK and sampled on the opposite edge.

The sample words transferred on the ADx pins are aligned with the WCLK signal, according to the configured serial interface format. The first WCLK edge of a sample word is either rising or falling, depending on the configured serial interface format.

The period from the first WCLK edge of an audio sample (one or more channels) to the first WCLK edge of the next audio sample is called a frame. A frame consists of either one or two phases. A phase is divided into the following intervals:

- Data delay (optional): The BCLK periods between the first WCLK edge and MSB of the (first) audio channel data transferred during the phase
- Word: The BCLK periods during which sample words are transferred on the ADx pin or pins
  - For single-phase, from 1 to 8 sample words are transferred back-to-back.
  - For dual-phase, one sample word is transferred. The least significant bit (LSB) of the sample word can extend into the data delay interval of the next phase.
- Idle (optional): The BCLK periods between the word interval and the next phase

A sample word on the serial interface can contain from 8 to 24 bits.

25.4.4 I^2S

I^2S is a dual-phase format with a 50% WCLK duty cycle and the start of an MSB of each sample word aligned with each edge of WCLK plus one BCLK period. For any given frame, the left channel is transferred first when WCLK is low, and the right channel is transferred next when WCLK is high. Figure 25-2 shows the I^2S serial format.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK.

The I^2S format is unique in the sense that the CC13x4x10 and CC26x4x10 device platform can automatically detect the number of BCLK periods per WCLK period. Therefore, I^2S supports any BCLK rate from an external audio clock source and also variable sample word length:

- If the configured sample word length is higher than the number of bits per WCLK period, the sample words are truncated.
- If the configured sample word length is lower than the number of bits per WCLK period, the sample words are zero-padded.

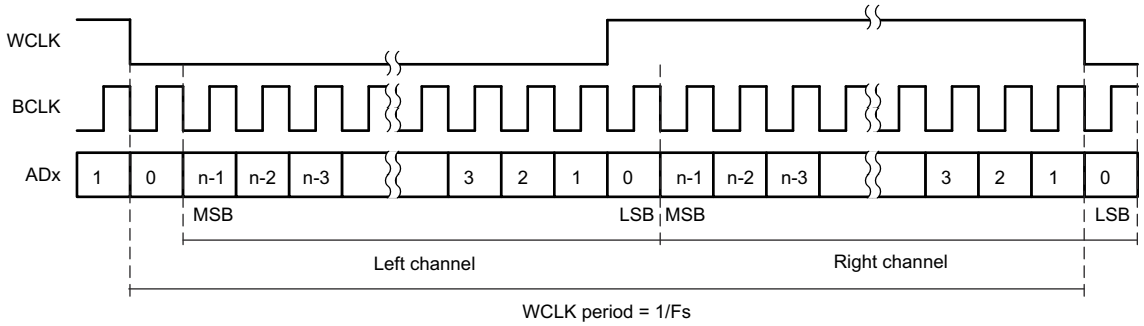


Figure 25-2. I^2S Serial Format

25.4.4.1 Register Configuration

The register configuration follows:

- I2S:AIFWCLKSRC.WCLK_INV = 1
- I2S:AIFFMTCFG.DUAL_PHASE = 1
- I2S:AIFFMTCFG.SMPL_EDGE = 1
- I2S:AIFFMTCFG.WORD_LEN = Maximum number of bits per sample word
- I2S:AIFFMTCFG.DATA_DELAY = 1

25.4.5 Left-Justified (LJF)

LJF is a dual-phase format with a 50% WCLK duty cycle and the start of an MSB of each sample word aligned with each edge of WCLK. For any given frame, the left channel is transferred first when WCLK is high, and the right channel is transferred next when WCLK is low.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK. Figure 25-3 shows the LJF serial format.

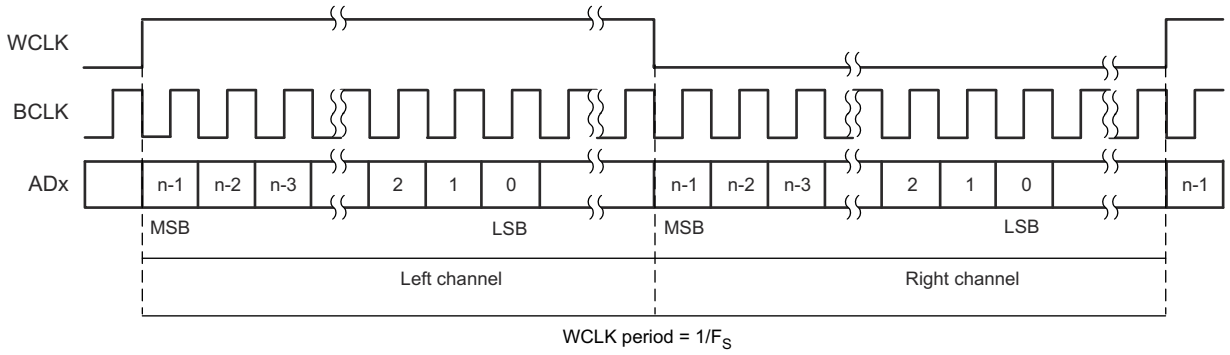


Figure 25-3. LJF Serial Format

25.4.5.1 Register Configuration

The register configuration follows:

- I2S:AIFWCLKSRC.WCLK_INV = 0
- I2S:AIFFMTCFG.DUAL_PHASE = 1
- I2S:AIFFMTCFG.SMPL_EDGE = 1
- I2S:AIFFMTCFG.WORD_LEN = Maximum number of bits per sample word
- I2S:AIFFMTCFG.DATA_DELAY = 0

WORD_LEN must be equal to or less than the number of BCLK periods per phase.

25.4.6 Right-Justified (RJF)

RJF is a dual-phase format with a 50% WCLK duty cycle and the end of an LSB of each sample word aligned with each edge of WCLK. For any given frame, the left channel is transferred first when WCLK is high, and the right channel is transferred next when WCLK is low.

Data is sampled on the rising edge of BCLK and updated on the falling edge of BCLK. Figure 25-4 shows the RJF serial format.

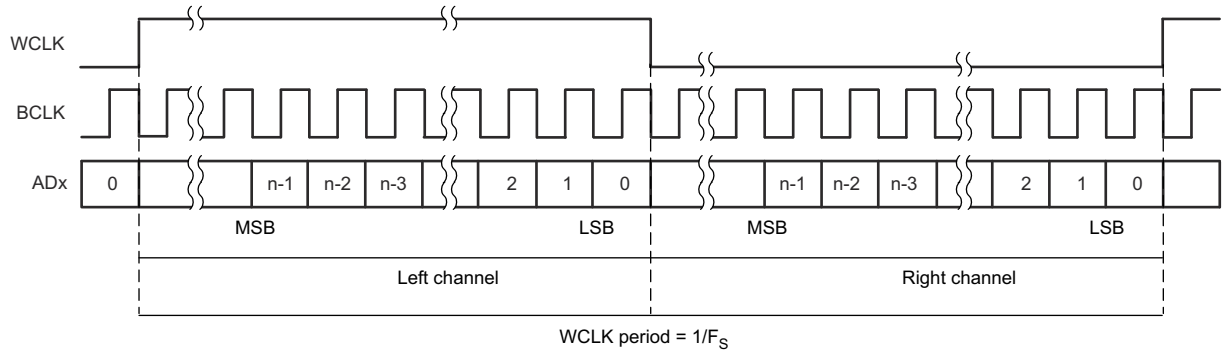


Figure 25-4. RJF Serial Format

25.4.6.1 Register Configuration

The register configuration follows:

- I2S:AIFWCLKSRC.WCLK_INV = 0
- I2S:AIFFMTCFG.DUAL_PHASE = 1
- I2S:AIFFMTCFG.SMPL_EDGE = 1
- I2S:AIFFMTCFG.WORD_LEN = Exact number of bits per sample word
- I2S:AIFFMTCFG.DATA_DELAY = Number of BCLK periods per phase minus the value of I2S:AIFFMTCFG.WORD_LEN

DATA_DELAY + WORD_LEN must be equal to or less than the number of BCLK periods per phase.

25.4.7 DSP

DSP is a single-phase format where WCLK is high for one BCLK period, and the MSB of the first sample word is typically aligned with this WCLK pulse, or it follows in the next BCLK period. Sample words for subsequent audio channels are then transferred back-to-back, followed by an idle period until the next phase or frame begins.

Data is sampled on the falling edge of BCLK and updated on the rising edge of BCLK. Figure 25-5 shows the DSP serial format with zero data delay.

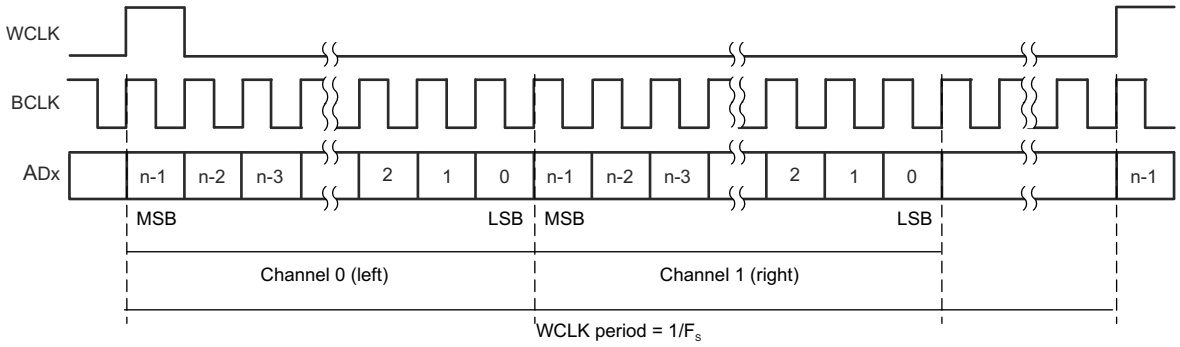


Figure 25-5. DSP Serial Format (Zero Data Delay)

25.4.7.1 Register Configuration

The register configuration follows:

- I2S:AIFWCLKSRC.WCLK_INV = 0
- I2S:AIFFMTCFG.DUAL_PHASE = 0
- I2S:AIFFMTCFG.SMPL_EDGE = 0
- I2S:AIFFMTCFG.WORD_LEN = Exact number of bits per sample word
- I2S:AIFFMTCFG.DATA_DELAY = 0 or 1

DATA_DELAY + (WORD_LEN × *channel count*) must be equal to or less than the number of BCLK periods per phase.

The *channel count* is determined by the MSB set in the I2S:AIFWMASK0 and I2S:AIFWMASK1 registers.

25.4.8 Clock Configuration

The audio clocks signals, MCLK, BCLK, and WCLK, can be generated either internally by the PRCM module or by an external clock source.

The internally generated audio clock signals might not be suitable for all applications for the reasons that follow:

- Jitter performance
- Clock configuration only provides support for frequencies that can be divided down from 48 MHz.
- Frequency that cannot be tuned to maintain constant audio latency

When using the internally generated audio clock, the 48 MHz high-frequency system clock, SCLK_HF, must always be derived from XOSC_HF. This derivation is done to reduce jitter and to avoid side-effects of switching the SCLK_HF source from RCOSC_HF to XOSC_HF (missing clock cycles and frequency shift).

25.4.8.1 Internal Audio Clock Source

Internal audio clock source must be selected for both the I^2S module and the PRCM module:

- I2S:AIFWCLKSRC = 2
- PRCM:I2SBCLKSEL.SRC = 1

The setting for the PRCM:I2SCLKCTL.SMPL_ON_POSEDGE register specifies on which edge of BCLK the WCLK signal shall be sampled. This setting must be equal to the setting for the I2S:AIFFMTCFG.SMPL_EDGE register.

The MCLK, BCLK, WCLK frequencies, and WCLK duty cycle are configured as follows:

- MCLK frequency = 48 MHz / PRCM:I2SMCLKDIV.MDIV
- BCLK frequency = 48 MHz / PRCM:I2SBCLKDIV.BDIV
- For WCLK the configuration depends on the duty cycle (PRCM:I2SWCLKDIV.WDIV is referred to as WDIV):
  - Single phase (DSP format): PRCM:I2SCLKCTL.WCLK_PHASE = 0
    - WCLK is high for 1 BCLK period and low for WDIV[9:0] (1 to 1023) BCLK periods.
    - WCLK frequency = BCLK frequency / (1 + PRCM:I2SWCLKDIV.WDIV[9:0])
  - Dual phase (I^2S, LJF, and RJF formats): PRCM:I2SCLKCTL.WCLK_PHASE = 1
    - WCLK is high for WDIV[9:0] (1 to 1023) BCLK periods and low for WDIV[9:0] (1 to 1023) BCLK periods.
    - WCLK frequency = BCLK frequency / (2 × WDIV[9:0])
  - User-defined: PRCM:I2SCLKCTL.WCLK_PHASE = 2
    - WCLK is high for WDIV[7:0] (1 to 255) BCLK periods and low for WDIV[15:8] (1 to 255) BCLK periods.
    - WCLK frequency = BCLK frequency / (WDIV[7:0] + WDIV[15:8])

The signal generation of the clock signals MCLK, BCLK, and WCLK must be enabled by setting PRCM:I2SCLKCTL.EN = 1. The MCLK, BCLK, and WCLK signals are static low when PRCM:I2SCLKCTL.EN = 0.

25.4.8.2 External Audio Clock Source

External audio clock source must be selected both for the I^2S module and for the PRCM module:

- I2S:AIFWCLKSRC = 1
- PRCM:I2SBCLKSEL.SRC = 0

25.5 Memory Interface

The integrated direct memory access controller (DMA) independently handles input samples (from one or two ADx pins to RAM) and output samples (from RAM or Flash to one or two ADx pins).

There is one shift-register and one sample word buffer for each ADx pin. The DMA stores input sample words to memory while the next sample words are received, and it loads output sample words from memory while the last loaded sample words are transmitted.

The DMA operates on blocks of memory. While the DMA works on one block of memory, software must write the start address of the next memory block to I2S:AIFINPTRNEXT for input samples and to I2S:AIFOUTPTRNEXT for output samples.

25.5.1 Sample Word Length

The sample word length in memory (16 or 24 bits) is configured independently of sample word length on the serial interface (8 to 24 bits). Sample words are truncated when the destination is shorter than the source and zero-padded when the destination is longer than the source.

The I2S:AIFFMTCFG.MEM_LEN_24 field configures whether sample words in memory are 16 bit or 24 bit:

- 0: Each 16-bit sample word is moved to or from memory using one 16-bit transfer. The DMA pointers written to the I2S:AIFINPTRNEXT and I2S:AIFOUTPTRNEXT registers must be halfword aligned.
- 1: Each 24-bit sample word is moved to or from memory using one 16-bit transfer and one 8-bit transfer in a locked bus transfer. The DMA pointers written to the I2S:AIFINPTRNEXT and I2S:AIFOUTPTRNEXT registers do not need to be aligned to any memory size.

25.5.2 Channel Mapping

For each ADx pin, the corresponding I2S:AIFWMASKx register determines which sample words are present in memory:

- For each frame when I2S:AIFFMTCFG.DUAL_PHASE = 0 (DSP format):
  - Input: The I2S:AIFWMASKx.MASK register determines whether or not channels are stored to memory.
  - Output: The I2S:AIFWMASKx.MASK register determines whether or not channels are fetched from memory. The ADx output is low for excluded channels.
- For each frame when I2S:AIFFMTCFG.DUAL_PHASE = 1 (I^2S, LJF, and RJF formats):
  - Mono: I2S:AIFWMASKx.MASK = 0x01
    - Input: Left (0) channel is stored to memory.
    - Output: Left (0) channel is fetched from memory and is repeated for the right channel.
  - Stereo: I2S:AIFWMASKx.MASK = 0x03
    - Input: Left (0) and right (1) channels are stored to memory.
    - Output: Left (0) and right (1) channels are fetched from memory.

25.5.3 Sample Storage in Memory

Sample words are stored to memory in little-endian byte order, meaning that the least significant byte (LSByte) is stored at the lower byte address, and the most significant byte (MSByte) is stored at the higher byte address.

If both ADx pins are configured as input or both ADx pins are configured as output, the sample words for each audio channel are stored AD0 first and AD1 last.

Figure 25-6, Figure 25-7, Figure 25-8, Figure 25-9, and Figure 25-10 show examples of sample storage for typical use cases (F = frame index, C = channel index, P = ADx pin index).

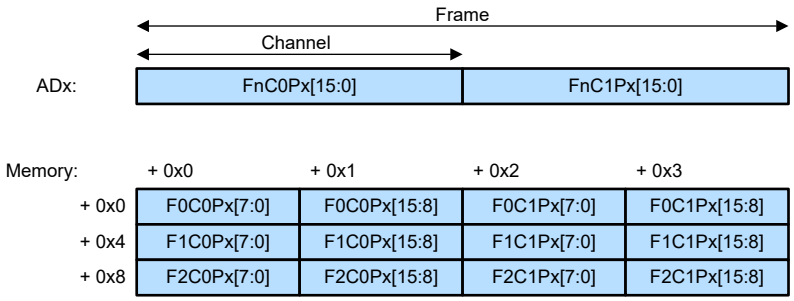


Figure 25-6. 16-Bit Stereo I2S, LJF, and RJF Formats on One ADx Pin, Showing Three Frames in Memory

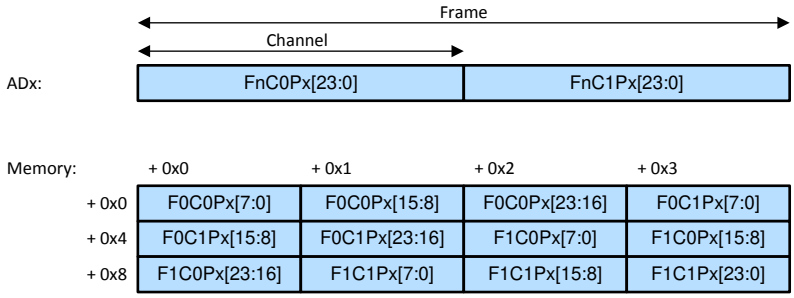


Figure 25-7. 24-Bit Stereo I^2S, LJF, and RJF Formats on One ADx Pin, Showing Two Frames in Memory

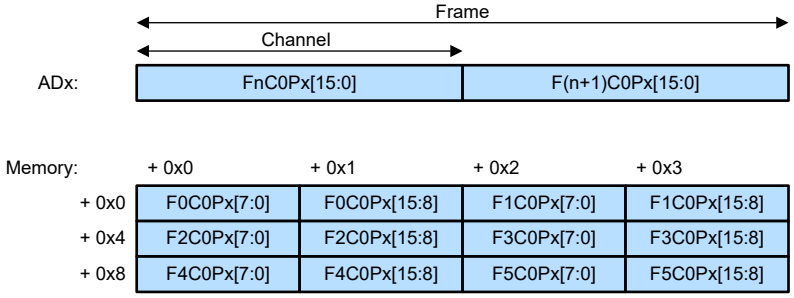


Figure 25-8. 16-Bit Mono I2S, LJF, and RJF Formats on One ADx Pin, Showing Six Frames in Memory

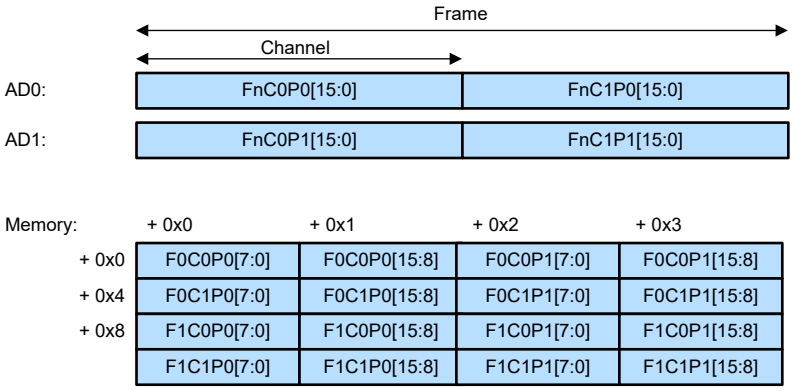


Figure 25-9. 16-Bit I^2S Format on AD0 and AD1 Pins, Showing Two Frames in Memory

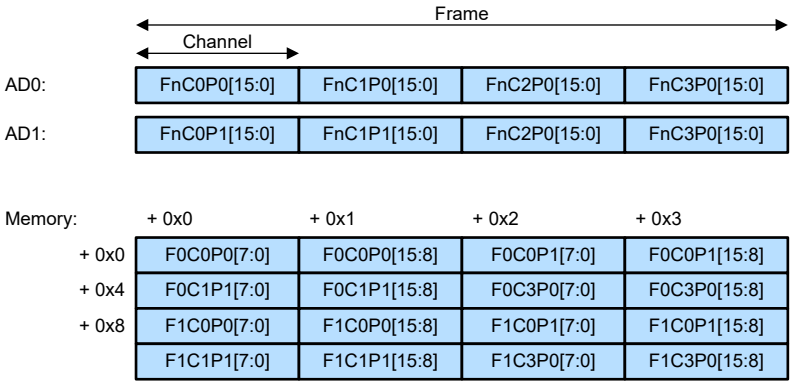


Figure 25-10. 16-Bit DSP Format on AD0 and AD1 Pins, Showing Two Frames in Memory

25.5.4 DMA Operation

The DMA operates on blocks of memory. Each input and output DMA memory block contains the input and output sample words, respectively, for I2S:AIFDMACFG.END_FRAME_IDX + 1 frames.

Writing a nonzero value to I2S:AIFDMACFG.END_FRAME_IDX initializes the DMA and prepares it to be started. Writing zero to I2S:AIFDMACFG.END_FRAME_IDX disables the DMA and resets the serial audio interface.

If input ADx pins are used, software must write memory block start addresses for input DMA to I2S:AIFINPTRNEXT. The current input DMA memory location can be observed in I2S:AIFINPTR.

If output ADx pins are used, software must write memory block start addresses for output DMA to I2S:AIFOUTPTRNEXT. The current output DMA memory location can be observed in I2S:AIFOUTPTR.

This writing operation or DMA operation allows the software to implement sample block ring buffers in memory with an arbitrary number of blocks for input and output samples.

25.5.4.1 Start-Up

All other audio interface-related register configuration (pins, serial format, clocks, sample word sizes, and channel mapping) must be completed before writing a nonzero value to the I2S:AIFDMACFG.END_FRAME_IDX register.

To prepare input and output DMA for start-up, the software must preload the first and second DMA pointers to be used and must arm the DMA:

- Write the first memory block start addresses to be used to I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT

- Set I2S:AIFDMACFG.END_FRAME_IDX = the number of frames per block minus one.
  - This loads I2S:AIFINPTRNEXT into I2S:AIFINPTR, and I2S:AIFOUTPTRNEXT into I2S:AIFOUTPTR, and the output DMA will immediately prefetch sample words for the first two audio channels.
- Write the second memory block start addresses to be used to I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT.

The input and output DMA are then started independently by the samplestamp generator, as described in Section 25.6.2.

25.5.4.2 Operation

To maintain DMA operation, software must provide new memory block start addresses each time a memory block is finished.

When a block is finished, the following occurs:

- For the input memory interface block:
  - I2S:AIFINPTR = I2S:AIFINPTRNEXT
  - I2S:AIFINPTRNEXT = 0x00000000
  - I2S:IRQFLAGS.AIF_DMA_IN is set to generate an I2S_IRQ interrupt
- For the output memory interface block:
  - I2S:AIFOUTPTR = I2S:AIFOUTPTRNEXT
  - I2S:AIFOUTPTRNEXT = 0x00000000
  - I2S:IRQFLAGS.AIF_DMA_OUT is set to generate an I2S_IRQ interrupt

To handle this operation, software must either poll if the I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT registers are zero, or use the I2S:IRQFLAGS.AIF_DMA_IN and (or) I2S:IRQFLAGS.AIF_DMA_OUT interrupt requests.

Software must write the new memory block start addresses to I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT before the running block finishes. If the running block finishes while I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT are zero, the affected DMA channels stop and I2S:IRQFLAGS.PTR_ERR is set.

25.5.4.3 Shutdown

Before DMA shutdown, all output external audio devices (for example, a DAC) should be muted, or silence should the transmitted on output ADx pins.

The DMA must not be stopped while there could be ongoing DMA memory transfer.

When using the internal audio clock source or an external audio clock source that cannot stop unexpectedly, software should use the following procedure to stop the DMA:

- Stop writing to the I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT registers
- Optional: Wait for I2S:IRQFLAGS.PTR_ERR to occur
- Wait for I2S:AIFINPTRNEXT and I2S:AIFOUTPTRNEXT to become zero
- Write I2S:AIFDMACFG.END_FRAME_IDX = 0

When using an external audio clock source that can stop unexpectedly, software should use the following procedure to stop the DMA:

- Stop writing to the I2S:AIFINPTRNEXT and (or) I2S:AIFOUTPTRNEXT registers
- Stop the external audio clock source
- Wait for I2S:IRQFLAGS.WCLK_TIMEOUT to occur, or for I2S:AIFINPTRNEXT and I2S:AIFOUTPTRNEXT to become zero, whichever happens first
- Write I2S:AIFDMACFG.END_FRAME_IDX = 0

25.6 Samplestamp Generator

The samplestamp generator is used to start input and output DMA operation. The samplestamp generator is also used to synchronize I^2S modules over a wireless network, so correct and fixed audio latency can be achieved. Synchronization over a wireless network is an optional feature that can be bypassed.

The samplestamp generator is enabled and is running while I2S:STMPCTL.STMP_EN = 1. Counter and capture registers are reset when software writes I2S:STMPCTL.STMP_EN = 0.

Figure 25-11 shows a simplified block diagram of the samplestamp generator:

- The lower part of the diagram is related to start-up of input and output DMA
- The upper part of the diagram is related to RFCORE event capture for I^2S module synchronization

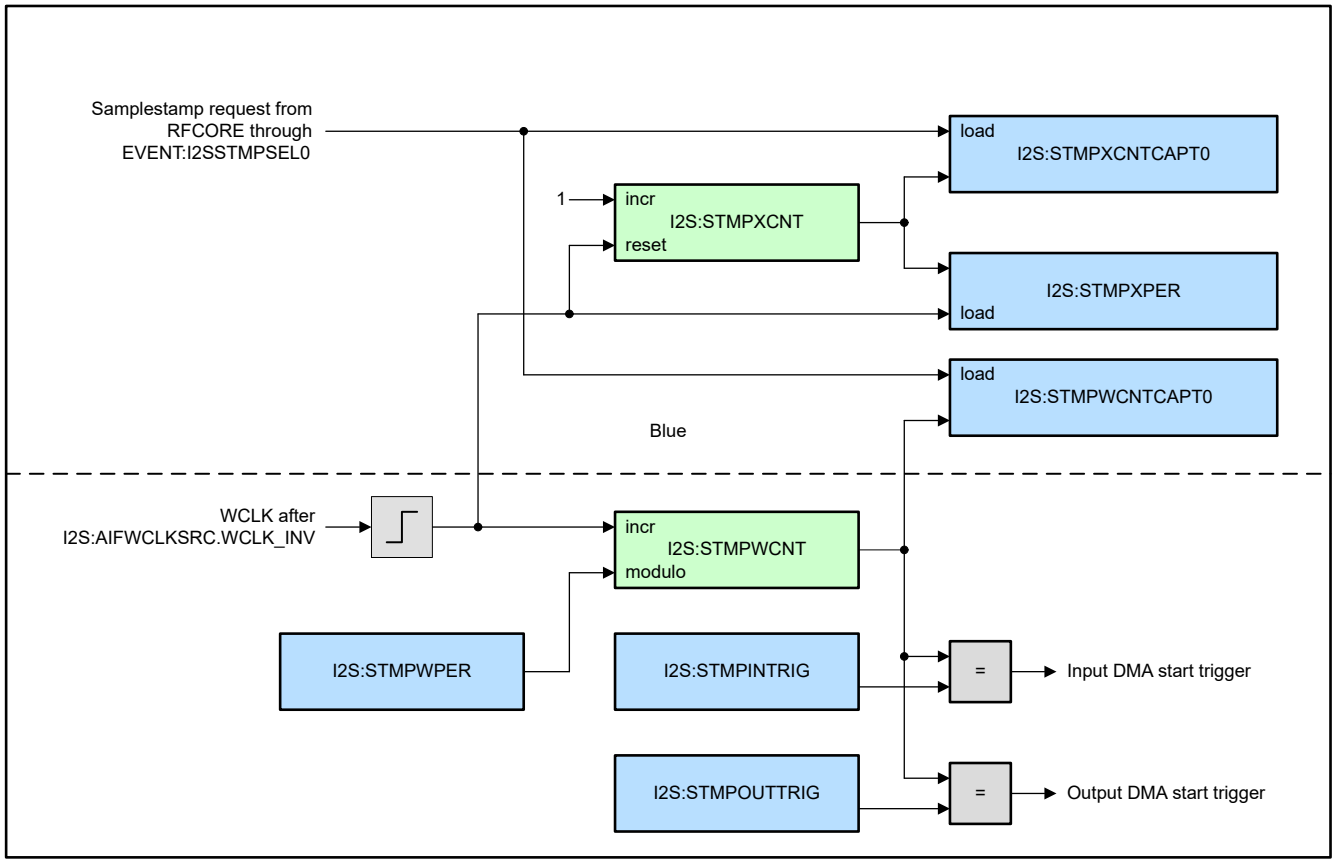


Figure 25-11. Samplestamp Generator

25.6.1 Samplestamp Counters

The samplestamp generator counts frames (WCLK periods) and 48 MHz clock cycles (crystal oscillator periods) within each frame:

- I2S:STMPWCNT increments at the first WCLK edge of each frame, module the period value I2S:STMPWPER
- I2S:STMPXCNT resets to 0 at the first WCLK edge of each frame, and then increments by 1 for each 48 MHz clock cycle. Reading I2S:STMPWCNT latches the read value of I2S:STMPXCNT

Software can modify the value of I2S:STMPWCNT by writing an absolute value to I2S:STMPWSET or a relative value to I2S:STMPWADD.

25.6.2 Start-Up Triggers

The I2S:STMPINTRIG and I2S:STMPOUTTRIG registers contain I2S:STMPWCNT compare values that are used to start the input and output DMA, respectively:

- When I2S:STMPWCNT equals I2S:STMPINTRIG, the input DMA begins storing sample words to memory in the next frame: (I2S:STMPINTRIG + 1) % I2S:STMPWPER
- When I2S:STMPWCNT equals I2S:STMPOUTTRIG, the output DMA begins outputting sample words from memory in the next frame: (I2S:STMPOUTTRIG + 1) % I2S:STMPWPER

To avoid false start-up triggers, I2S:STMPINTRIG and I2S:STMPOUTTRIG must initially be equal to or higher than I2S:STMPWPER.

The I2S:STMPCTL.IN_RDY and I2S:STMPCTL.OUT_RDY status bits are set when the input and output DMA are ready to be started and cleared when DMA start triggers have occurred.

25.6.3 Samplestamp Capture

A capture request signal can be routed from RFCORE to trigger samplestamp capture. The EVENT:I2SSTMPSEL0 register selects the RFCORE signal to be used. Whenever this capture request signal is high:

- The current value of I2S:STMPXCNT is copied into I2S:STMPXCNTCAPT0
- The current value of I2S:STMPWCNT is copied into I2S:STMPWCNTCAPT0

Also, on the first WCLK edge of each frame, the current value of I2S:STMPXCNT is captured in I2S:STMPXPER, and I2S:STMPXCNT then restarts counting from 0.

Using these values, a fixed-point samplestamp value can be calculated:

I2S:STMPWCNTCAPT0 + (I2S:STMPXCNTCAPT0 / I2S:STMPXPER)

Notice that the value of I2S:STMPXPER will not normally be captured at the same time as the other values. Therefore, I2S:STMPXPER can be less than I2S:STMPXCNTCAPT0.

25.6.4 Achieving Constant Audio Latency

The following actions can be taken to achieve the same constant audio latency in either direction over a wireless network (from the I^2S pins on one CC13x4x10 and CC26x4x10 device platform to the I^2S pins on another CC13x4x10 and CC26x4x10 device platform):

- One node must be defined as audio clock master and the other node must be defined as audio clock slave. The slave must use an external audio clock source with adjustable rate.
- For both nodes, set I2S:STMPWPER = N × (I2S:AIFDMACFG.END_FRAME_IDX + 1), where N is a whole number.
  - The value of I2S:STMPWPER equals audio latency in number of frames.
  - The value of I2S:STMPWPER also equals the memory buffer size in number of samples.
- Perform samplestamp capture on the master when it transmits the RF packet synchronization word, and include the value of the fixed-point samplestamp in the transmitted packet.
- Perform samplestamp capture on the slave when it receives the RF packet synchronization word, and store the samplestamp value of the master in the RF packet. Calculate the difference between the samplestamp values of the master and slave, which is used to:
  - Initially offset the I2S:STMPWCNT counter of the slave so that it matches the samplestamp value of the master.
  - While running, adjust the external audio clock source rate so that the difference between the samplestamp values of the slave and the master approach 0.
- For both nodes, set up DMA pointers and DMA start triggers so that the value of STMPWCNT represents the input and output buffer positions of the current frame on the ADx pins.

25.7 Error Detection

The I^2S module can detect errors related to the following:

- DMA operation
- Audio clock signal integrity

The following errors are detected:

- WCLK frequency error (STMPXPERMIN:VALUE)
- Noise on the WCLK signal (IRQFLAGS:WCLK_ERR)
- Audio clock loss (IRQFLAGS:WCLK_TIMEOUT)
- DMA pointer not loaded in time (IRQFLAGS:PTR_ERR)
- DMA transfer not completed in time (IRQFLAGS:BUS_ERR)

25.8 Usage

25.8.1 Start-Up Sequence

Perform the following steps in the indicated order to begin I^2S module operation:

1. Set up dependencies (see Section 25.4.1)
2. Configure the pins (see Chapter 15)
3. Configure the serial format (see Section 25.4.3)
4. Configure the clock (see Section 25.4.8)
5. Configure the sample word length (see Section 25.5.1)
6. Configure the channel mapping (see Section 25.5.2)
7. Perform the DMA start-up sequence (see Section 25.5.4.1)
8. Set up the samplestamp generator:
   - Set the I2S:STMPWPER register
   - Set the I2S:STMPINTRIG and I2S:STMPOUTTRIG > I2S:STMPWPER to avoid false DMA start triggers
   - Set I2S:STMPCTL.EN = 1
   - If needed, follow the guidelines for achieving constant audio latency (see Section 25.6.4)
   - Otherwise, just set I2S:STMPINTRIG and I2S:STMPOUTTRIG to match the current (I2S:STMPWCNT + 2) % I2S:STMPWPER

Note

DMA interrupts will begin after the DMA has completed the first sample block or blocks.

25.8.2 Shutdown Sequence

Perform the following steps in the indicated order to end I^2S module operation:

1. DMA shutdown sequence (see Section 25.5.4.3)
2. Set I2S:STMPCTL.EN = 0
3. Disable the internal or external audio clock source
4. Disable dependencies (see Section 25.4.1)

25.9 I2S Registers

Table 25-2 lists the memory-mapped registers for the I2S registers. All register offset addresses not listed in Table 25-2 should be considered as reserved locations and the register contents should not be modified.

Table 25-2. I2S Registers

Offset	Acronym	Register Name	Section
0h	AIFWCLKSRC	WCLK Source Selection	Section 25.9.1
4h	AIFDMACFG	DMA Buffer Size Configuration	Section 25.9.2
8h	AIFDIRCFG	Pin Direction	Section 25.9.3
Ch	AIFFMTCFG	Serial Interface Format Configuration	Section 25.9.4
10h	AIFWMASK0	Word Selection Bit Mask for Pin 0	Section 25.9.5
14h	AIFWMASK1	Word Selection Bit Mask for Pin 1	Section 25.9.6
1Ch	AIFPWMVALUE	Audio Interface PWM Debug Value	Section 25.9.7
20h	AIFINPTRNEXT	DMA Input Buffer Next Pointer	Section 25.9.8
24h	AIFINPTR	DMA Input Buffer Current Pointer	Section 25.9.9
28h	AIFOUTPTRNEXT	DMA Output Buffer Next Pointer	Section 25.9.10
2Ch	AIFOUTPTR	DMA Output Buffer Current Pointer	Section 25.9.11
34h	STMPCTL	Samplestamp Generator Control Register	Section 25.9.12
38h	STMPXCNTCAPT0	Captured XOSC Counter Value, Capture Channel 0	Section 25.9.13
3Ch	STMPXPER	XOSC Period Value	Section 25.9.14
40h	STMPWCNTCAPT0	Captured WCLK Counter Value, Capture Channel 0	Section 25.9.15
44h	STMPWPER	WCLK Counter Period Value	Section 25.9.16
48h	STMPINTRIG	WCLK Counter Trigger Value for Input Pins	Section 25.9.17
4Ch	STMPOUTTRIG	WCLK Counter Trigger Value for Output Pins	Section 25.9.18
50h	STMPWSET	WCLK Counter Set Operation	Section 25.9.19
54h	STMPWADD	WCLK Counter Add Operation	Section 25.9.20
58h	STMPXPERMIN	XOSC Minimum Period Value	Section 25.9.21
5Ch	STMPWCNT	Current Value of WCNT	Section 25.9.22
60h	STMPXCNT	Current Value of XCNT	Section 25.9.23
64h	STMPXCNTCAPT1	Internal	Section 25.9.24
68h	STMPWCNTCAPT1	Internal	Section 25.9.25
70h	IRQMASK	Interrupt Mask Register	Section 25.9.26
74h	IRQFLAGS	Raw Interrupt Status Register	Section 25.9.27
78h	IRQSET	Interrupt Set Register	Section 25.9.28
7Ch	IRQCLR	Interrupt Clear Register	Section 25.9.29

Complex bit access types are encoded to fit into small table cells. Table 25-3 shows the codes that are used for access types in this section.

Table 25-3. I2S Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

25.9.1 AIFWCLKSRC Register (Offset = 0h) [Reset = 00000000h]

AIFWCLKSRC is shown in Table 25-4.

Return to the Summary Table.

WCLK Source Selection

Table 25-4. AIFWCLKSRC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	WCLK_INV	R/W	0h	Inverts WCLK source (pad or internal) when set. 0: Not inverted 1: Inverted
1-0	WCLK_SRC	R/W	0h	Selects WCLK source for AIF (should be the same as the BCLK source). The BCLK source is defined in the PRCM:I2SBCLKSEL.SRC 0h = None ('0') 1h = External WCLK generator, from pad 2h = Internal WCLK generator, from module PRCM 3h = Not supported. Will give same WCLK as 'NONE' ('00')

25.9.2 AIFDMACFG Register (Offset = 4h) [Reset = 00000000h]

AIFDMACFG is shown in Table 25-5.

Return to the Summary Table.

DMA Buffer Size Configuration

Table 25-5. AIFDMACFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	END_FRAME_IDX	R/W	0h	Defines the length of the DMA buffer. Writing a non-zero value to this register field enables and initializes AIF. Note that before doing so, all other configuration must have been done, and AIFINPTRNEXT/ AIFOUTPTRNEXT must have been loaded.

25.9.3 AIFDIRCFG Register (Offset = 8h) [Reset = 00000000h]

AIFDIRCFG is shown in Table 25-6.

Return to the Summary Table.

Pin Direction

Table 25-6. AIFDIRCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5-4	AD1	R/W	0h	Configures the AD1 audio data pin usage: 0x3: Reserved 0h = Not in use (disabled) 1h = Input mode 2h = Output mode
3-2	RESERVED	R	0h	Reserved
1-0	AD0	R/W	0h	Configures the AD0 audio data pin usage: 0x3: Reserved 0h = Not in use (disabled) 1h = Input mode 2h = Output mode

25.9.4 AIFFMTCFG Register (Offset = Ch) [Reset = 00000170h]

AIFFMTCFG is shown in Table 25-7.

Return to the Summary Table.

Serial Interface Format Configuration

Table 25-7. AIFFMTCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	DATA_DELAY	R/W	1h	The number of BCLK periods between a WCLK edge and MSB of the first word in a phase: 0x00: LJF and DSP format 0x01: I2S and DSP format 0x02: RJF format ... 0xFF: RJF format Note: When 0, MSB of the next word will be output in the idle period between LSB of the previous word and the start of the next word. Otherwise logical 0 will be output until the data delay has expired.
7	MEM_LEN_24	R/W	0h	The size of each word stored to or loaded from memory: 0h = 16BIT : 16-bit (one 16 bit access per sample) 1h = 24BIT : 24-bit (one 8 bit and one 16 bit locked access per sample)
6	SMPL_EDGE	R/W	1h	On the serial audio interface, data (and wclk) is sampled and clocked out on opposite edges of BCLK. 0h = Data is sampled on the negative edge and clocked out on the positive edge. 1h = Data is sampled on the positive edge and clocked out on the negative edge.
5	DUAL_PHASE	R/W	1h	Selects dual- or single-phase format. 0: Single-phase: DSP format 1: Dual-phase: I2S, LJF and RJF formats
4-0	WORD_LEN	R/W	10h	Number of bits per word (8-24): In single-phase format, this is the exact number of bits per word. In dual-phase format, this is the maximum number of bits per word. Values below 8 and above 24 give undefined behavior. Data written to memory is always aligned to 16 or 24 bits as defined by MEM_LEN_24. Bit widths that differ from this alignment will either be truncated or zero padded.

25.9.5 AIFWMASK0 Register (Offset = 10h) [Reset = 00000003h]

AIFWMASK0 is shown in Table 25-8.

Return to the Summary Table.

Word Selection Bit Mask for Pin 0

Table 25-8. AIFWMASK0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	MASK	R/W	3h	Bit-mask indicating valid channels in a frame on AD0. In single-phase mode, each bit represents one channel, starting with LSB for the first word in the frame. A frame can contain up to 8 channels. Channels that are not included in the mask will not be sampled and stored in memory, and clocked out as '0'. In dual-phase mode, only the two LSBs are considered. For a stereo configuration, set both bits. For a mono configuration, set bit 0 only. In mono mode, only channel 0 will be sampled and stored to memory, and channel 0 will be repeated when clocked out. In mono mode, only channel 0 will be sampled and stored to memory, and channel 0 will be repeated in the second phase when clocked out. If all bits are zero, no input words will be stored to memory, and the output data lines will be constant '0'. This can be utilized when PWM debug output is desired without any actively used output pins.

25.9.6 AIFWMASK1 Register (Offset = 14h) [Reset = 00000003h]

AIFWMASK1 is shown in Table 25-9.

Return to the Summary Table.

Word Selection Bit Mask for Pin 1

Table 25-9. AIFWMASK1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	MASK	R/W	3h	Bit-mask indicating valid channels in a frame on AD1. In single-phase mode, each bit represents one channel, starting with LSB for the first word in the frame. A frame can contain up to 8 channels. Channels that are not included in the mask will not be sampled and stored in memory, and clocked out as '0'. In dual-phase mode, only the two LSBs are considered. For a stereo configuration, set both bits. For a mono configuration, set bit 0 only. In mono mode, only channel 0 will be sampled and stored to memory, and channel 0 will be repeated when clocked out. In mono mode, only channel 0 will be sampled and stored to memory, and channel 0 will be repeated in the second phase when clocked out. If all bits are zero, no input words will be stored to memory, and the output data lines will be constant '0'. This can be utilized when PWM debug output is desired without any actively used output pins.

25.9.7 AIFPWMVALUE Register (Offset = 1Ch) [Reset = 00000000h]

AIFPWMVALUE is shown in Table 25-10.

Return to the Summary Table.

Audio Interface PWM Debug Value

Table 25-10. AIFPWMVALUE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	PULSE_WIDTH	R/W	0h	The value written to this register determines the width of the active high PWM pulse (pwm_debug), which starts together with MSB of the first output word in a DMA buffer: 0x0000: Constant low 0x0001: Width of the pulse (number of BCLK cycles, here 1). ... 0xFFFE: Width of the pulse (number of BCLK cycles, here 65534). 0xFFFF: Constant high

25.9.8 AIFINPTRNEXT Register (Offset = 20h) [Reset = 00000000h]

AIFINPTRNEXT is shown in Table 25-11.

Return to the Summary Table.

DMA Input Buffer Next Pointer

Table 25-11. AIFINPTRNEXT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PTR	R/W	0h	Pointer to the first byte in the next DMA input buffer. The read value equals the last written value until the currently used DMA input buffer is completed, and then becomes null when the last written value is transferred to the DMA controller to start on the next buffer. This event is signalized by IRQFLAGS.AIF_DMA_IN. At startup, the value must be written once before and once after configuring the DMA buffer size in AIFDMACFG. The next pointer must be written to this register while the DMA function uses the previously written pointer. If not written in time, IRQFLAGS.PTR_ERR will be raised and all input pins will be disabled.

25.9.9 AIFINPTR Register (Offset = 24h) [Reset = 00000000h]

AIFINPTR is shown in Table 25-12.

Return to the Summary Table.

DMA Input Buffer Current Pointer

Table 25-12. AIFINPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PTR	R	0h	Value of the DMA input buffer pointer currently used by the DMA controller. Incremented by 1 (byte) or 2 (word) for each AHB access.

25.9.10 AIFOUTPTRNEXT Register (Offset = 28h) [Reset = 00000000h]

AIFOUTPTRNEXT is shown in Table 25-13.

Return to the Summary Table.

DMA Output Buffer Next Pointer

Table 25-13. AIFOUTPTRNEXT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PTR	R/W	0h	Pointer to the first byte in the next DMA output buffer. The read value equals the last written value until the currently used DMA output buffer is completed, and then becomes null when the last written value is transferred to the DMA controller to start on the next buffer. This event is signalized by IRQFLAGS.AIF_DMA_OUT. At startup, the value must be written once before and once after configuring the DMA buffer size in AIFDMACFG. At this time, the first two samples will be fetched from memory. The next pointer must be written to this register while the DMA function uses the previously written pointer. If not written in time, IRQFLAGS.PTR_ERR will be raised and all output pins will be disabled.

25.9.11 AIFOUTPTR Register (Offset = 2Ch) [Reset = 00000000h]

AIFOUTPTR is shown in Table 25-14.

Return to the Summary Table.

DMA Output Buffer Current Pointer

Table 25-14. AIFOUTPTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	PTR	R	0h	Value of the DMA output buffer pointer currently used by the DMA controller Incremented by 1 (byte) or 2 (word) for each AHB access.

25.9.12 STMPCTL Register (Offset = 34h) [Reset = 00000000h]

STMPCTL is shown in Table 25-15.

Return to the Summary Table.

Samplestamp Generator Control Register

Table 25-15. STMPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reserved
2	OUT_RDY	R	0h	Low until the output pins are ready to be started by the samplestamp generator. When started (that is STMPOUTTRIG equals the WCLK counter) the bit goes back low.
1	IN_RDY	R	0h	Low until the input pins are ready to be started by the samplestamp generator. When started (that is STMPINTRIG equals the WCLK counter) the bit goes back low.
0	STMP_EN	R/W	0h	Enables the samplestamp generator. The samplestamp generator must only be enabled after it has been properly configured. When cleared, all samplestamp generator counters and capture values are cleared.

25.9.13 STMPXCNTCAPT0 Register (Offset = 38h) [Reset = 00000000h]

STMPXCNTCAPT0 is shown in Table 25-16.

Return to the Summary Table.

Captured XOSC Counter Value, Capture Channel 0

Table 25-16. STMPXCNTCAPT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CAPT_VALUE	R	0h	The value of the samplestamp XOSC counter (STMPXCNT.CURR_VALUE) last time an event was pulsed (event source selected in [EVENT.I2SSTMPSEL0.EV] for channel 0). This number corresponds to the number of 24 MHz clock cycles since the last positive edge of the selected WCLK. The value is cleared when STMPCTL.STMP_EN = 0. Note: Due to buffering and synchronization, WCLK is delayed by a small number of BCLK periods and clk periods. Note: When calculating the fractional part of the sample stamp, STMPXPER may be less than this bit field.

25.9.14 STMPXPER Register (Offset = 3Ch) [Reset = 00000000h]

STMPXPER is shown in Table 25-17.

Return to the Summary Table.

XOSC Period Value

Table 25-17. STMPXPER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R	0h	The number of 24 MHz clock cycles in the previous WCLK period (that is - the next value of the XOSC counter at the positive WCLK edge, had it not been reset to 0). The value is cleared when STMPCTL.STMP_EN = 0.

25.9.15 STMPWCNTCAPT0 Register (Offset = 40h) [Reset = 00000000h]

STMPWCNTCAPT0 is shown in Table 25-18.

Return to the Summary Table.

Captured WCLK Counter Value, Capture Channel 0

Table 25-18. STMPWCNTCAPT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CAPT_VALUE	R	0h	The value of the samplestamp WCLK counter (STMPWCNT.CURR_VALUE) last time an event was pulsed (event source selected in EVENT:I2SSTMPSEL0.EV for channel 0). This number corresponds to the number of positive WCLK edges since the samplestamp generator was enabled (not taking modification through STMPWADD/STMPWSET into account). The value is cleared when STMPCTL.STMP_EN = 0.

25.9.16 STMPWPER Register (Offset = 44h) [Reset = 00000000h]

STMPWPER is shown in Table 25-19.

Return to the Summary Table.

WCLK Counter Period Value

Table 25-19. STMPWPER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	Used to define when STMPWCNT is to be reset so number of WCLK edges are found for the size of the sample buffer. This is thus a modulo value for the WCLK counter. This number must correspond to the size of the sample buffer used by the system (that is the index of the last sample plus 1).

25.9.17 STMPINTRIG Register (Offset = 48h) [Reset = 00000000h]

STMPINTRIG is shown in Table 25-20.

Return to the Summary Table.

WCLK Counter Trigger Value for Input Pins

Table 25-20. STMPINTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	IN_START_WCNT	R/W	0h	Compare value used to start the incoming audio streams. This bit field shall equal the WCLK counter value during the WCLK period in which the first input word(s) are sampled and stored to memory (that is the sample at the start of the very first DMA input buffer). The value of this register takes effect when the following conditions are met: - One or more pins are configured as inputs in AIFDIRCFG. - AIFDMACFG has been configured for the correct buffer size, and at least 32 BCLK cycle ticks have happened. Note: To avoid false triggers, this bit field should be set higher than STMPWPER.VALUE.

25.9.18 STMPOUTTRIG Register (Offset = 4Ch) [Reset = 00000000h]

STMPOUTTRIG is shown in Table 25-21.

Return to the Summary Table.

WCLK Counter Trigger Value for Output Pins

Table 25-21. STMPOUTTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	OUT_START_WCNT	R/W	0h	Compare value used to start the outgoing audio streams. This bit field must equal the WCLK counter value during the WCLK period in which the first output word(s) read from memory are clocked out (that is the sample at the start of the very first DMA output buffer). The value of this register takes effect when the following conditions are met: - One or more pins are configured as outputs in AIFDIRCFG. - AIFDMACFG has been configured for the correct buffer size, and 32 BCLK cycle ticks have happened. - 2 samples have been preloaded from memory (examine the AIFOUTPTR register if necessary). Note: The memory read access is only performed when required, that is channels 0/1 must be selected in AIFWMASK0/AIFWMASK1. Note: To avoid false triggers, this bit field should be set higher than STMPWPER.VALUE.

25.9.19 STMPWSET Register (Offset = 50h) [Reset = 00000000h]

STMPWSET is shown in Table 25-22.

Return to the Summary Table.

WCLK Counter Set Operation

Table 25-22. STMPWSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	0h	WCLK counter modification: Sets the running WCLK counter equal to the written value.

25.9.20 STMPWADD Register (Offset = 54h) [Reset = 00000000h]

STMPWADD is shown in Table 25-23.

Return to the Summary Table.

WCLK Counter Add Operation

Table 25-23. STMPWADD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE_INC	R/W	0h	WCLK counter modification: Adds the written value to the running WCLK counter. If a positive edge of WCLK occurs at the same time as the operation, this will be taken into account. To add a negative value, write "STMPWPER.VALUE - value".

25.9.21 STMPXPERMIN Register (Offset = 58h) [Reset = 0000FFFFh]

STMPXPERMIN is shown in Table 25-24.

Return to the Summary Table.

XOSC Minimum Period Value
Minimum Value of STMPXPER

Table 25-24. STMPXPERMIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALUE	R/W	FFFFh	Each time STMPXPER is updated, the value is also loaded into this register, provided that the value is smaller than the current value in this register. When written, the register is reset to 0xFFFF (65535), regardless of the value written. The minimum value can be used to detect extra WCLK pulses (this registers value will be significantly smaller than STMPXPER.VALUE).

25.9.22 STMPWCNT Register (Offset = 5Ch) [Reset = 00000000h]

STMPWCNT is shown in Table 25-25.

Return to the Summary Table.

Current Value of WCNT

Table 25-25. STMPWCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CURR_VALUE	R	0h	Current value of the WCLK counter

25.9.23 STMPXCNT Register (Offset = 60h) [Reset = 00000000h]

STMPXCNT is shown in Table 25-26.

Return to the Summary Table.

Current Value of XCNT

Table 25-26. STMPXCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CURR_VALUE	R	0h	Current value of the XOSC counter, latched when reading STMPWCNT.

25.9.24 STMPXCNTCAPT1 Register (Offset = 64h) [Reset = 00000000h]

STMPXCNTCAPT1 is shown in Table 25-27.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 25-27. STMPXCNTCAPT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CAPT_VALUE	R	0h	Internal. Only to be used through TI provided API.

25.9.25 STMPWCNTCAPT1 Register (Offset = 68h) [Reset = 00000000h]

STMPWCNTCAPT1 is shown in Table 25-28.

Return to the Summary Table.

Internal. Only to be used through TI provided API.

Table 25-28. STMPWCNTCAPT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CAPT_VALUE	R	0h	Internal. Only to be used through TI provided API.

25.9.26 IRQMASK Register (Offset = 70h) [Reset = 00000000h]

IRQMASK is shown in Table 25-29.

Return to the Summary Table.

Interrupt Mask Register
Selects mask states of the flags in IRQFLAGS that contribute to the I2S_IRQ event.

Table 25-29. IRQMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	AIF_DMA_IN	R/W	0h	IRQFLAGS.AIF_DMA_IN interrupt mask 0: Disable 1: Enable
4	AIF_DMA_OUT	R/W	0h	IRQFLAGS.AIF_DMA_OUT interrupt mask 0: Disable 1: Enable
3	WCLK_TIMEOUT	R/W	0h	IRQFLAGS.WCLK_TIMEOUT interrupt mask 0: Disable 1: Enable
2	BUS_ERR	R/W	0h	IRQFLAGS.BUS_ERR interrupt mask 0: Disable 1: Enable
1	WCLK_ERR	R/W	0h	IRQFLAGS.WCLK_ERR interrupt mask 0: Disable 1: Enable
0	PTR_ERR	R/W	0h	IRQFLAGS.PTR_ERR interrupt mask. 0: Disable 1: Enable

25.9.27 IRQFLAGS Register (Offset = 74h) [Reset = 00000000h]

IRQFLAGS is shown in Table 25-30.

Return to the Summary Table.

Raw Interrupt Status Register

Table 25-30. IRQFLAGS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	AIF_DMA_IN	R	0h	Set when condition for this bit field event occurs (auto cleared when input pointer is updated - AIFINPTRNEXT), see description of AIFINPTRNEXT register for details.
4	AIF_DMA_OUT	R	0h	Set when condition for this bit field event occurs (auto cleared when output pointer is updated - AIFOUTPTRNEXT), see description of AIFOUTPTRNEXT register for details
3	WCLK_TIMEOUT	R	0h	Set when the sample stamp generator does not detect a positive WCLK edge for 65535 clk periods. This signalizes that the internal or external BCLK and WCLK generator source has been disabled. The bit is sticky and may only be cleared by software (by writing '1' to IRQCLR.WCLK_TIMEOUT).
2	BUS_ERR	R	0h	Set when a DMA operation is not completed in time (that is audio output buffer underflow, or audio input buffer overflow). This error requires a complete restart since word synchronization has been lost. The bit is sticky and may only be cleared by software (by writing '1' to IRQCLR.BUS_ERR). Note that DMA initiated transactions to illegal addresses will not trigger an interrupt. The response to such transactions is undefined.
1	WCLK_ERR	R	0h	Set when: - An unexpected WCLK edge occurs during the data delay period of a phase. Note unexpected WCLK edges during the word and idle periods of the phase are not detected. - In dual-phase mode, when two WCLK edges are less than 4 BCLK cycles apart. - In single-phase mode, when a WCLK pulse occurs before the last channel. This error requires a complete restart since word synchronization has been lost. The bit is sticky and may only be cleared by software (by writing '1' to IRQCLR.WCLK_ERR).
0	PTR_ERR	R	0h	Set when AIFINPTRNEXT or AIFOUTPTRNEXT has not been loaded with the next block address in time. This error requires a complete restart since word synchronization has been lost. The bit is sticky and may only be cleared by software (by writing '1' to IRQCLR.PTR_ERR).

25.9.28 IRQSET Register (Offset = 78h) [Reset = 00000000h]

IRQSET is shown in Table 25-31.

Return to the Summary Table.

Interrupt Set Register

Table 25-31. IRQSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	AIF_DMA_IN	W	0h	1: Sets the interrupt of IRQFLAGS.AIF_DMA_IN (unless a auto clear criteria was given at the same time, in which the set will be ignored)
4	AIF_DMA_OUT	W	0h	1: Sets the interrupt of IRQFLAGS.AIF_DMA_OUT (unless a auto clear criteria was given at the same time, in which the set will be ignored)
3	WCLK_TIMEOUT	W	0h	1: Sets the interrupt of IRQFLAGS.WCLK_TIMEOUT
2	BUS_ERR	W	0h	1: Sets the interrupt of IRQFLAGS.BUS_ERR
1	WCLK_ERR	W	0h	1: Sets the interrupt of IRQFLAGS.WCLK_ERR
0	PTR_ERR	W	0h	1: Sets the interrupt of IRQFLAGS.PTR_ERR

25.9.29 IRQCLR Register (Offset = 7Ch) [Reset = 00000000h]

IRQCLR is shown in Table 25-32.

Return to the Summary Table.

Interrupt Clear Register

Table 25-32. IRQCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	AIF_DMA_IN	W	0h	1: Clears the interrupt of IRQFLAGS.AIF_DMA_IN (unless a set criteria was given at the same time in which the clear will be ignored)
4	AIF_DMA_OUT	W	0h	1: Clears the interrupt of IRQFLAGS.AIF_DMA_OUT (unless a set criteria was given at the same time in which the clear will be ignored)
3	WCLK_TIMEOUT	W	0h	1: Clears the interrupt of IRQFLAGS.WCLK_TIMEOUT (unless a set criteria was given at the same time in which the clear will be ignored)
2	BUS_ERR	W	0h	1: Clears the interrupt of IRQFLAGS.BUS_ERR (unless a set criteria was given at the same time in which the clear will be ignored)
1	WCLK_ERR	W	0h	1: Clears the interrupt of IRQFLAGS.WCLK_ERR (unless a set criteria was given at the same time in which the clear will be ignored)
0	PTR_ERR	W	0h	1: Clears the interrupt of IRQFLAGS.PTR_ERR (unless a set criteria was given at the same time in which the clear will be ignored)

Chapter 26

Radio

The radio in the CC13x4x10 and CC26x4x10 device platform offers a wide variety of different operational modes, covering many different packet formats. The radio firmware executes from the CC13x4x10 and CC26x4x10 platform radio domain on an Arm® Cortex®-M0 processor, which can provide extensive baseband automation. The application software interfaces and interoperates with the radio firmware using shared memory interface (system RAM or radio RAM) and specific handshake hardware (radio doorbell). The radio is a Non-secure peripheral and hence all shared memory used need to be in Non-secure regions.

26.1 RF Core 1986
26.2 Radio Doorbell 1988
26.3 RF Core HAL 1992
26.4 Data Queue Usage 2029
26.5 IEEE 802.15.4 2033
26.6 Bluetooth® Low Energy 2056
26.7 Data Handling 2077
26.8 Radio Operation Command Descriptions 2078
26.9 Immediate Commands 2117
26.10 Proprietary Radio 2118
26.11 Radio Registers 2140

26.1 RF Core

The RF core contains an Arm® Cortex®-M0 processor that interfaces the analog RF and baseband circuits, handles data to and from the system side, and assembles the information bits in a given packet structure. The RF core offers a high-level, command-based application programming interface (API) to the system CPU (Arm® Cortex®-M33 processor). The RF core can autonomously handle the time-critical aspects of the radio protocols (Wi-SUN™, Matter and Zigbee®, Bluetooth® Low Energy, and so on), thus offloading the system CPU and leaving more resources for the user's application.

The RF core has a dedicated 8 KB retention SRAM block and a 4 KB non-retention SRAM block and runs almost entirely from separate ROM. The contents of the non-retention SRAM block is lost every time the radio is powered down.

26.1.1 High-Level Description and Overview

The RF core receives high-level requests from the system CPU and performs all the necessary transactions to fulfill them. These requests are basically oriented to the transmission and reception of information through the radio channel, but can also include additional maintenance tasks such as calibration, test, or debug features.

As a general framework, the transactions between the system CPU and the RF core operate as follows:

- The RF core can access data and configuration parameters from the system RAM. This reduces the memory requirements of the RF core, avoids needless traffic between the different parts of the system, and reduces the total energy consumption.
- In a similar fashion, the RF core can decode and write back the contents of the received radio packet, together with status information, to the system RAM.
- For protocol confidentiality and authentication support purposes, the RF core can also access the security subsystem.
- In general, the RF core recognizes complex commands from the system CPU (CCA transmissions, RX with automatic acknowledge, and so forth) and divides them into subcommands without further intervention from the system CPU.

Figure 26-1 shows the external interfaces and dependencies of the RF core.

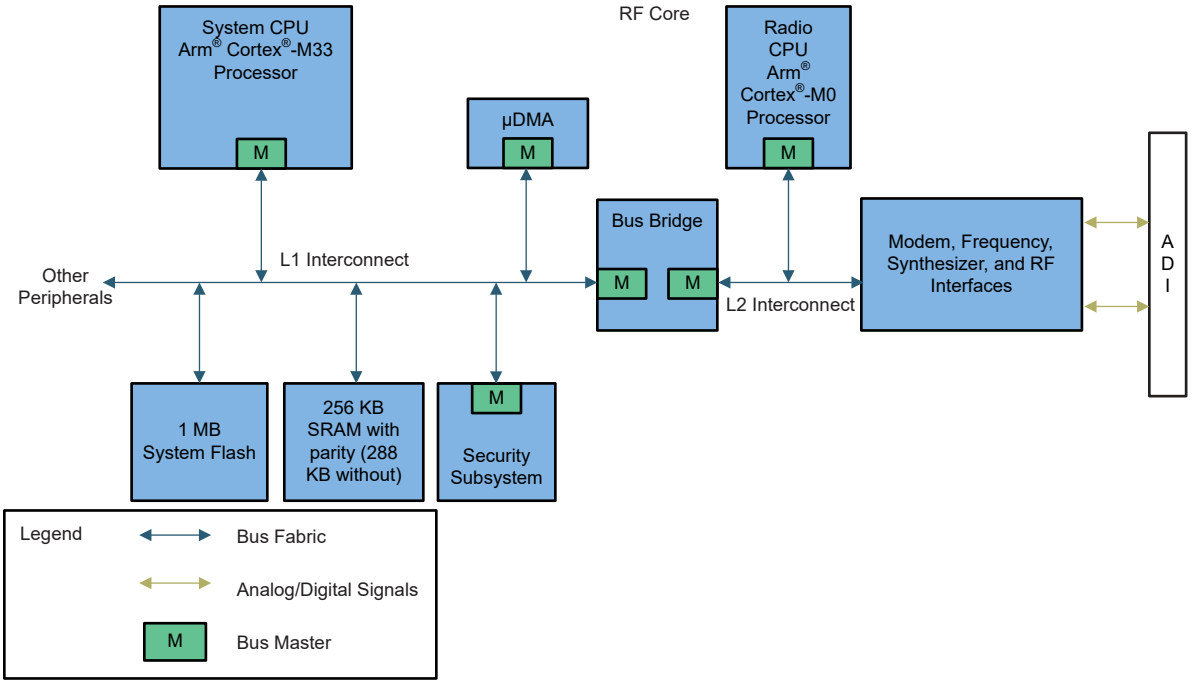


Figure 26-1. Limited RF Core Overview with External Dependencies

Each block shown in Figure 26-1 performs the following functions:

System Side

- System CPU: Main system processor that runs the user's application, together with the high-level protocol stack (for a number of supported configurations) and eventually some higher-level MAC features for some protocols. The system CPU runs code from the boot ROM and the system Flash.
- System RAM: Contains packet information (TX and RX payloads) and the different parameters or configuration options for a given transaction.
- Security Subsystem: Encompasses the different elements to provide protocol confidentiality and authentication.
- DMA: Optionally charged with the task of moving information from the radio RAM to the system RAM and vice versa, if direct CPU access is not used.

Radio Side

- Radio CPU: Main RF core processor. Receives high-level commands from the system CPU and schedules them into the different parts of the RF core.
- Modem, Frequency Synthesizer, RF Interfaces: This is the core of the radio, converting the bits into modulated signals and vice versa.

26.2 Radio Doorbell

The radio doorbell module (RFC_DBELL) is the primary means of communication between the system CPU and the radio CPU, also known as command and packet engine (CPE). The radio doorbell contains a set of dedicated registers, parameters in any of the RAMs of the device platform, and a set of interrupts to both the radio CPU and to the system CPU.

In addition, parameters and payload are transferred through the system RAM or the radio RAM. If any parameters or payload are in the system RAM, the system CPU must remain powered. However, if everything is in the radio RAM, the system CPU may go into power-down mode to save current.

During operation, the radio CPU updates parameters and payload in RAM and raises interrupts. The system CPU can mask out interrupts so that it remains in idle or power-down mode until the entire radio operation finishes.

Because the system CPU and the radio CPU share a common RAM area, ensure that no contention or race conditions can occur. This is achieved in software by rules set up in the radio hardware abstraction layer (HAL).

Figure 26-2 shows the relevant modules for information exchange between the CPUs.

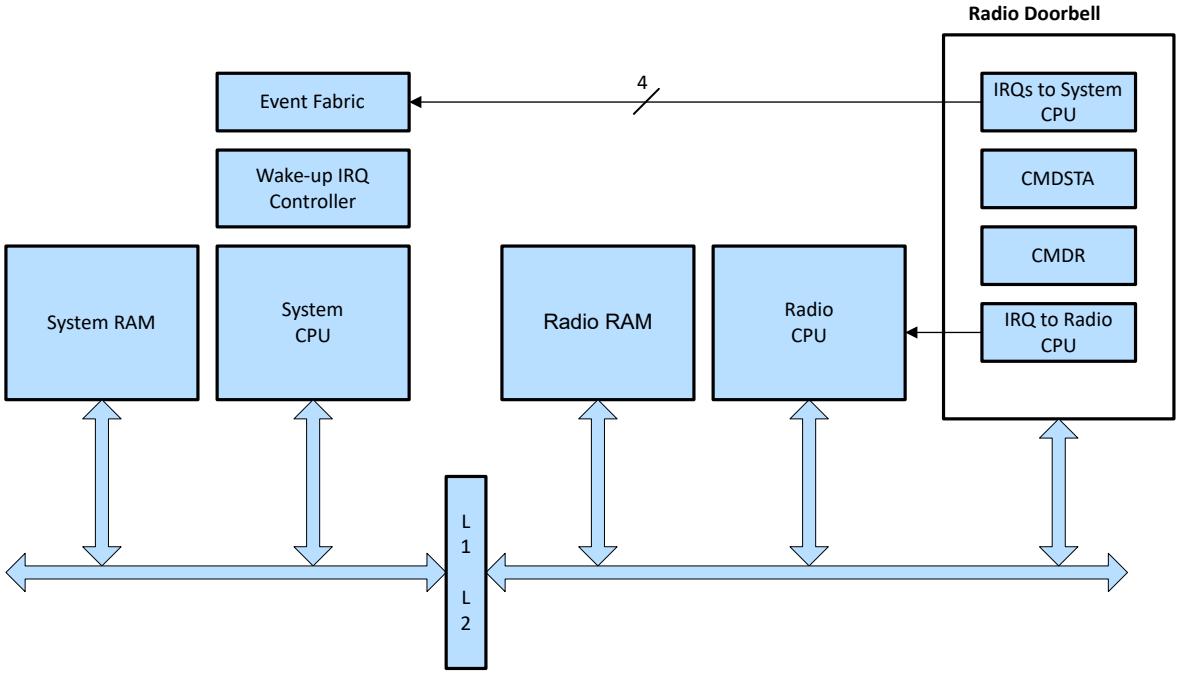


Figure 26-2. Hardware Support for the HAL

26.2.1 Special Boot Process

The PRCM:RFCBITS register aggregates control information to be evaluated by the CPE at boot. The features can be enabled/disabled individually by setting the corresponding bit, and help to automate the most commonly used functionalities, see Table 26-1.

Table 26-1. Format of PRCM:RFCBITS for Enabling Selected Features as Part of Boot Process

Bit Index	Value	Description
0	1	Special boot sequence
1..2	0	Boot options
3	x	If 1, force on all clocks that are enabled at boot time
4	x	If 1, request system bus immediately
5	x	If 1, start radio timer immediately
6	x	If 1, disable pointer checks immediately

Table 26-1. Format of PRCM:RFCBITS for Enabling Selected Features as Part of Boot Process (continued)

Bit Index	Value	Description
7	x	If 1, assume the RF core retention RAM is initialized
8..11	x	Reserved
12	x	If 1, enable interrupts Command_Started and FG_Command_Started to be generated
13..28	0	Reserved
29..31	7	Special boot option

26.2.2 Command and Status Register and Events

Sending commands to the radio is done through the RFC_DBELL:CMDR register, while the RFC_DBELL:CMDSTA read-only register provides status back from the radio. The CMDR register can only be written while it reads 0; otherwise, writes are ignored. When the CMDR register is 0 and a nonzero value is written to it, the radio CPU is notified and the CMDSTA register becomes 0. After this action, the value written is readable from the CMDR register until the radio CPU has processed the command, at which point it goes back to 0.

When the command is processed by the radio CPU, the CMDSTA register contains a nonzero status, which is provided at the same instant when the CMDR register goes back to 0. At this instant, an RFC_CMD_ACK interrupt occurs. This interrupt is also mapped to the RFC_DBELL:RFACKIFG register, which should be cleared when the interrupt is processed.

See Section 26.3.2 for the format of the command and status registers.

26.2.3 RF Core Interrupts

The RF core has four interrupt lines to the Arm® Cortex®-M33 processor (see Figure 26-2). The following interrupts are controlled by the radio doorbell module:

- RFC_CPE_0 (CPU interrupt 9, see EVENT:CPUIRQSEL9)
- RFC_CPE_1 (CPU interrupt 2, see EVENT:CPUIRQSEL2)
- RFC_HW_COMB (CPU interrupt 10, see EVENT:CPUIRQSEL10)
- RFC_CMD_ACK (CPU interrupt 11, see EVENT:CPUIRQSEL11)

26.2.3.1 RF Command and Packet Engine Interrupts

The two system-level interrupts RFC_CPE_0 and RFC_CPE_1 can be produced from a number of low-level interrupts produced by the CPE. Each of these low-level interrupts can be mapped to RFC_CPE_0 or RFC_CPE_1 using the RFC_DBELL:RFCPEISL register. In addition, interrupt generation at system level may be switched on and off using the RFC_DBELL:RFCPEIEN register.

In case of an event that triggers a low-level interrupt, the corresponding bit in the RFCPEIFG register is set to 1. Whenever a bit in RFC_DBELL:RFCPEIFG and the corresponding bit in RFCPEIEN are both 1, the system-level interrupt selected in RFCPEISL is raised. This means that the interrupt service routine (ISR) must clear the bits in RFCPEIFG that correspond to low-level interrupts that have been processed.

The register description for RFCPEIFG in Section 26.11.2 provides a list of the available interrupts.

Clear bits in RFCPEIFG by writing 0 to those bits, while any bits written to 1 remain unchanged.

Note

When clearing bits in the RFCPEIFG register, interrupts may be lost if a read-modify-write operation is done because interrupt flags that became active between the read and write operation might be lost. The Radio Software Bundle (rflib) in the SimpleLink™ CC13xx and CC26xx software development kit (SDK) shows how this is done correctly.

26.2.3.2 RF Core Hardware Interrupts

The system-level interrupt RFC_HW_COMB can be produced from a number of low-level interrupts produced by RF core hardware. Interrupt generation at system level may be switched on and off for each source by using the RFC_DBELL:RFHWIFG register.

In the case of an event that triggers a low-level interrupt, the corresponding bit in the RFC_DBELL:RFHWIFG register is set to 1. Whenever a bit in RFHWIFG and the corresponding bit in RFC_DBELL:RFHWIEN are both 1, the RFC_HW_COMB interrupt is raised. This means that the ISR should clear the bits in RFHWIFG that correspond to low-level interrupts that have been processed.

The register description for RFHWIFG in Section 26.11.2 provides a list of the available interrupts. In general, TI does not recommend servicing these interrupts in the System CPU, but the available radio timer channel interrupts may be served this way.

Clearing bits in RFHWIFG is done by writing 0 to those bits, while any bits written to 1 remain unchanged.

Note

When clearing bits in RFHWIFG, interrupts may be lost if a read-modify-write operation is done. Therefore, the same rule applies for the RFHWIFG register as for RFCPEIFG (see Section 26.2.3.1).

26.2.3.3 RF Core Command Acknowledge Interrupt

The system-level interrupt RFC_CMD_ACK is produced when an RF core command is acknowledged, that is, when the status becomes available in RFC_DBELL:CMDSTA (see Section 26.11.2). When the status becomes available, the RFC_DBELL:RFACKIFG.ACKFLAG register bit is set to 1. Whenever this bit is 1, the RFC_CMD_ACK interrupt is raised, which means that the ISR must clear RFACKIFG.ACKFLAG when processing the RFC_CMD_ACK interrupt.

26.2.4 Radio Timer

The radio has its own dedicated timer, the radio timer (RAT) module. The RAT is a free-running 32-bit timer that runs on 4 MHz. The RAT has eight channels with compare and capture functionality. Five of these channels are reserved for the radio CPU, while the remaining three are available for use by the System CPU. The available channels are numbered 5, 6, and 7.

The RAT can only run while the RF core is powered up. The RAT must be started by the command CMD_START_RAT or CMD_SYNC_START_RAT. The radio timer must be running to run a radio operation command with delayed start or any radio operation command that runs the receiver or transmitter.

When the RAT is running, the current value of the timer can be read from the RFC_RAT:RATCNT register (see Section 26.11.1).

26.2.4.1 Compare and Capture Events

The available channels may be set up in compare mode or capture mode.

Compare mode can be set up using the CMD_SET_RAT_CMP command (see Section 26.3.3.2.10). In this case, the timer generates an interrupt when the counter reaches the value given by compareTime. The interrupt is mapped to RFC_DBELL:RFHWIFG (see Section 26.2.3.2 and Section 26.11.2). For the available RAT channels, the interrupt flags in use are RATCH5, RATCH6, and RATCH7. Optionally, it is also possible to control an I/O pin when the counter reaches the value given by compareTime (see Section 26.2.4.2). When the CMD_SET_RAT_CMP command is sent, the value of compareTime is stored in the radio channel value register (RFC_RAT:RATCHnVAL) corresponding to the selected channel (see Section 26.11.1).

Capture mode can be used to capture a transition on an input pin and record the RAT counter value at the time when the transition occurred. Compare mode can be set up using the CMD_SET_RAT_CPT command (see Section 26.3.3.2.11). When the transition occurs, the current value of the RAT is stored in the RATCHnVAL register corresponding to the selected channel (see Section 26.11.1), and the timer generates an interrupt. As for compare mode, the interrupt is mapped to RFHWIFG. For the available RAT channels, the interrupt flags in

use are RATCH5, RATCH6, and RATCH7. If single-capture mode is configured in CMD_SET_CPT, only the first transition is captured, unless the channel is armed again, as explained in the following paragraph. If repeated mode is configured, every transition is captured.

Note

In this case, the captured value in RFC_RAT:RATCHnVAL register may be overwritten at any time if a new transition occurs.

A channel set up in compare mode or single capture mode may be armed or disarmed. When CMD_SET_RAT_CMP or CMD_SET_RAT_CPT is sent, the channel is armed automatically, and when the capture or compare event occurs, the channel is disarmed automatically. A disarmed channel does not produce any interrupt or cause any timer value to be captured. In addition, a channel may be armed or disarmed using CMD_ARM_RAT_CH or CMD_DISARM_RAT_CH (see Section 26.3.3.2.14 through Section 26.3.3.2.15). While disarmed, the channel keeps its configuration. To disable a channel that is not going to be re-armed with the same configuration, the CMD_DISABLE_RAT_CH command may be used (see Section 26.3.3.2.12).

26.2.4.2 Radio Timer Outputs

The RAT module has four controllable outputs, RAT_GPO0 through RAT_GPO3. These signals may be controlled by one of the RAT channels and mapped to signals available for the IOC using the RFC_DBELL:SYSGPOCTL register (see Section 26.11.2). The signal RAT_GPO0 is reserved for starting the transmitter and is controlled internally by the radio CPU (see Section 26.3.2.8). The other three signals may be configured using the CMD_SET_RAT_OUTPUT command (see Section 26.3.3.2.11). The different output modes indicate the transition of the output when an interrupt occurs on the chosen RAT channel except for the always-0 and always-1 configurations, which take effect immediately and may be used for initialization.

26.2.4.3 Synchronization with Real-Time Clock

When the radio is powered down, the RAT module is not counting. To keep a consistent time base over time for synchronized protocols, it is possible to synchronize the radio timer with the real-time clock (RTC) (see Chapter 18).

To allow synchronization after power up, the CMD_SYNC_STOP_RAT command (see Section 26.3.3.1.10) must be sent before RF core is powered down. This command (until the next RTC tick) stops the radio timer and returns a parameter rat0, should be stored and provided when the radio timer is restarted.

The next time the RF core is powered up and the RAT is started, this synchronization must be done using CMD_SYNC_START_RAT (see Section 26.3.3.1.11), where the rat0 parameter obtained from CMD_SYNC_STOP_RAT must be provided. This command starts the RAT, waits for an RTC tick, and adjusts the RAT. Depending on the application, it may not be necessary to run the CMD_SYNC_STOP_RAT command every time the radio is powered down; a previous value of rat0 may be reused. In some cases, however, this may cause issues if the radio is powered for a long time and the low-frequency and high-frequency crystal oscillators have a significant error relative to each other.

To get accurate synchronization, it is important that the system is running on the high-frequency crystal oscillator starting before the CMD_SYNC_START_RAT command is run and extending to after the CMD_SYNC_STOP_RAT command is finished.

Note

For the CMD_SYNC_START_RAT and CMD_SYNC_STOP_RAT commands, the AON_RTC:CTL RTC_UPD_EN register bit must be set to 1 (see Section 18.3.3). It is never necessary to reset this bit to 0; it may be set permanently to 1 when the RTC is started.

26.3 RF Core HAL

The RF core hides the complexity of the radio operations by providing a unified HAL to the system CPU.

Note

To ensure optimum radio performance, always use the latest radio patches provided by TI, available in theSimpleLink™ CC13xx and CC26xx software development kit (SDK).

26.3.1 Hardware Support

The radio HAL is supported by hardware, by means of the radio doorbell module in the RF core area and command descriptors in the system RAM.

26.3.2 Firmware Support

The RF core accepts a set of high-level primitives. The desired functionality is described at a high level in Section 26.3.2.1 through Section 26.3.2.8.

26.3.2.1 Commands

The radio CPU allows the user run a set of high-level primitives or commands from the system CPU. After a command is issued through the CMDR register, the radio CPU examines it and decides a course of action.

Three classes of commands are issued:

- Radio operation command
- Immediate command
- Direct command

For the first two classes of commands, CMDR contains a pointer to a command structure. This pointer must be a valid pointer with 32-bit word alignment, so the 2 least significant bits (LSBs) must be 0 0, as shown in Figure 26-3. A direct command is signaled by setting the 2 LSBs to 01 or 10 and placing the command ID number in bits 16 to 31 of CMDR. Bits 8 through 15, or alternatively 2 through 15, may be used as an optional byte parameter. Figure 26-4 shows the format for a direct command.

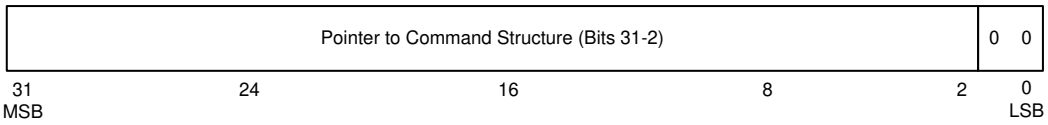


Figure 26-3. CMDR Register for Radio Operation Commands and Immediate Commands

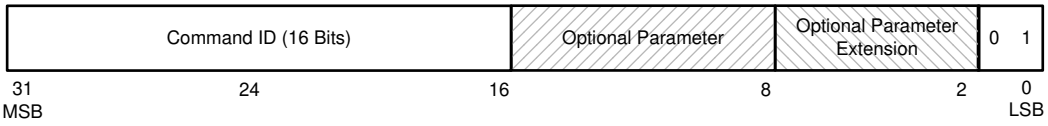


Figure 26-4. CMDR Register for Direct Commands

The data structure pointed to by the CMDR register for radio operation and immediate commands may be in the system RAM or the radio RAM. The system CPU must ensure that the memory area in use is free for access, in particular when using the radio RAM, where a part of the memory is reserved for use by the radio CPU. This information may be obtained with the CMD_GET_FW_INFO command (see Section 26.3.3.2.6). The format of the command follows the structure given in Section 26.3.2.6 and Section 26.3.2.6.1, and they are defined in more detail specifically for each command.

When deciding in which memory area to place data, consider which modules may be powered down:

- The radio RAM is accessible for the radio CPU at any time, but does not have retention when the radio is powered down. Data that must be retained must therefore be copied into or out of the radio RAM whenever the radio is powered up or down, respectively.

- The system RAM has retention in most low-power modes. If the system side is powered down, the radio CPU requests that it is powered up again to access the RAM. The active current consumption from the radio CPU accessing the system RAM is higher than the current consumption from accessing the radio RAM, especially if the system side could otherwise have been powered down.
- The lowest peak-power consumption is obtained by putting all data structures in the radio RAM and powering down the system side while the radio CPU is running. In some cases, the average power consumption may be lower by putting data structures in the system RAM, as less copying is then needed, and the system side can still be powered down for long periods (for instance, while the receiver is in sync search).

A radio operation command causes the radio hardware to be accessed. Radio operation commands can do operations such as transmitting or receiving a packet, setting up radio hardware registers, or doing more complex, protocol-dependent operations. A radio operation command can normally be issued only while the radio is idle.

An immediate command is a command to change or request status of the radio, or to manipulate TX or RX data queues. An intermediate command can monitor status such as received signal strength. An immediate command can be issued at any time, but the response is, in many cases, only of interest while a radio operation is ongoing.

A direct command is an immediate command with no parameters, or in some cases, a direct command has 1- or 2-byte parameters. A direct command is issued by sending a value to the CMDR register with the format of Figure 26-4. The 16 most significant bits (MSBs) contain the command ID of the immediate command to run. Bits 8 through 15 or bits 2 through 15 may contain an optional parameter if specified for the command.

26.3.2.2 Command Status

After a command is issued, the RFC_DBELL:CMDSTA register is updated by the radio CPU, causing an RFC_CMD_ACK interrupt to be sent back to the system CPU. This update occurs after the command finishes for immediate and direct commands and after the command is scheduled for radio operation commands. No new command may be issued until this interrupt is received. The CMDSTA register consists of 32 bits; the 8 LSBs give the result, while the upper 24 bits may be used for specific signaling in each command. Figure 26-5 shows this format.

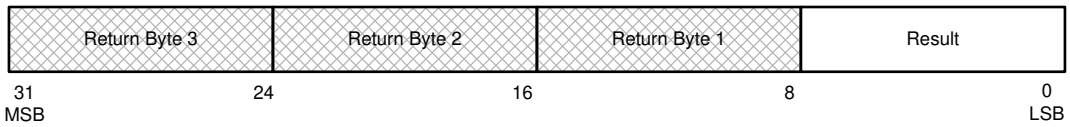


Figure 26-5. Format of RFC_DBELL:CMDSTA Register

In the result byte, bit 7 indicates whether an error occurred or not. The result byte of 0x00, meaning *pending*, is produced automatically by the radio doorbell hardware when a command is issued, and the other bits in the CMDSTA register also become 0, which is the value of CMDSTA before the RFC_CMD_ACK interrupt is raised.

Table 26-2 lists the values of the result byte in the CMDSTA register.

Table 26-2. Values of the Result Byte in the RFC_DBELL:CMDSTA Register

Value	Name	Description
No Error		
0x00	Pending	The command has not been parsed.
0x01	Done	Immediate command: The command finished successfully. Radio operation command: The command was successfully submitted for execution.
Error		
0x81	IllegalPointer	The pointer signaled in CMDR is not valid.
0x82	UnknownCommand	The command ID number in the command structure is unknown.
0x83	UnknownDirCommand	The command number for a direct command is unknown, or the command is not a direct command.
0x85	ContextError	An immediate or direct command was issued in a context where it is not supported.

Table 26-2. Values of the Result Byte in the RFC_DBELL:CMDSTA Register (continued)

Value	Name	Description
0x86	SchedulingError	A radio operation command was attempted to be scheduled while another operation was already running in the RF core. The new command is rejected, while the command already running is not impacted.
0x87	ParError	There were errors in the command parameters that are parsed on submission. For radio operation commands, errors in parameters parsed after start of the command are signaled by the command ending, and an error is indicated in the status field of that command structure.
0x88	QueueError	An operation on a data entry queue was attempted, but the operation was not supported by the queue in its current state.
0x89	QueueBusy	An operation on a data entry was attempted while that entry was busy.

In addition to the command status register, each radio operation command contains a status field (see Table 26-9). This field may have values in the following categories.

- **Idle**: The command has not started.
- **Pending**: The command is parsed, but the start trigger has not yet occurred.
- **Active**: The command is running.
- **Suspended**: The command was active, and may become active again. The command is supported only by certain IEEE 802.15.4 commands.
- **Finished**: The command is finished, and the system CPU is free to modify the command structure or free memory.
- **Skipped**: The command was skipped and never executed.

Table 26-3 lists the status codes for common commands and when parsing any command before starting.

Table 26-3. Common Radio Operation Status Codes

Number	Name	Description
Operation Not Finished		
0x0000	IDLE	Operation has not started.
0x0001	PENDING	Waiting for start trigger.
0x0002	ACTIVE	Running operation.
0x0003	SKIPPED	Operation skipped due to condition in another command.
Operation Finished Normally		
0x0400	DONE_OK	Operation ended normally.
0x0401	DONE_COUNTDOWN	Counter reached zero.
0x0402	DONE_RXERR	Operation ended with CRC error.
0x0403	DONE_TIMEOUT	Operation ended with time-out.
0x0404	DONE_STOPPED	Operation stopped after CMD_STOP command.
0x0405	DONE_ABORT	Operation aborted by CMD_ABORT command.
Operation Finished with Error		
0x0800	ERROR_PAST_START	The start trigger occurred in the past.
0x0801	ERROR_START_TRIG	Illegal start trigger parameter.
0x0802	ERROR_CONDITION	Illegal condition for next operation.
0x0803	ERROR_PAR	Error in a command specific parameter.
0x0804	ERROR_POINTER	Invalid pointer to next operation.
0x0805	ERROR_CMDID	Next operation has a command ID that is undefined or not a radio operation command.
0x0807	ERROR_NO_SETUP	Operation using RX, TX or synthesizer attempted without CMD_RADIO_SETUP.
0x0808	ERROR_NO_FS	Operation using RX or TX attempted without the synthesizer being programmed or powered on.

Table 26-3. Common Radio Operation Status Codes (continued)

Number	Name	Description
0x0809	ERROR_SYNTH_PROG	Synthesizer programming failed.
0x080A	ERROR_TXUNF	Modem TX underflow observed.
0x080B	ERROR_RXOVF	Modem RX overflow observed.
0x080C	ERROR_NO_RX	Data requested from last RX when no such data exists.

When the system CPU prepares a command structure, the CPU should initialize the status field to IDLE. Commands may be set up in a loop. If so, the system CPU must not modify command structures until the radio CPU becomes idle (the system CPU receives a LAST_COMMAND_DONE interrupt, even if the status is finished or skipped (see Section 26.11.2).

26.3.2.3 Interrupts

The radio CPU has 32 software interrupt sources that generate the RFC_CPE0 and RFC_CPE1 interrupts in the system CPU. An interrupt flag register can tell which software interrupt is raised, and the interrupts are enabled individually. In addition, the RFC_CMD_ACK interrupt is raised automatically when RFC_DBELL:CMDSTA is updated.

Some software-defined interrupts have a common meaning across all commands; the details of each of the other interrupts are defined for each protocol that uses a particular interrupt. Some interrupts are used in only one protocol, while others are used in several protocols. The interrupts are listed in the description of the RFC_DBELL:RFCPEIFG register (see Section 26.11.2).

26.3.2.4 Passing Data

There are two basic ways to pass data transmitted or received over the air; directly, or through a queue.

The most straight-forward way to pass data is to append it as part of the command parameters (directly or through a pointer). The exact format depends on the command being run; normally there is a length field and a data buffer for TX and a maximum length, received length (if variable length), and receive buffer for RX.

Queues are used to support operations where the number of packets received or transmitted cannot be known in advance. An operation can use one or more queues, for instance one RX and one TX queue for a combined RX/TX operation. Any operation using queues uses a common system for maintaining them, as explained in Section 26.3.2.7. For each radio command, it is stated whether they use a queue or a single buffer.

A radio operation command declares which data method is used.

26.3.2.5 Command Scheduling

The system CPU is responsible for scheduling the commands as required. When using low-power modes, the system CPU must wake up a short time before the start of the next operation, using the RTC.

A radio operation command can be scheduled with a delayed start (see Section 26.3.2.5.1). If a command is started with a delay, the radio CPU goes to idle mode until the command starts. The radio operation command is considered to be running during this delay, and no other radio operation command can be scheduled unless the pending command is aborted or stopped first.

The system CPU can schedule back-to-back radio operation commands by using the next operation pointer in any radio operation command. This pointer can point to the next command to perform in the chain, and by this method, complex operations can be made. Under some conditions (such as an error or the expiration of a timer), the next command is not started. Instead, the operation ends or a number of commands may be skipped (see Section 26.3.2.5.2). If a new command is scheduled while another command is running, the system CPU must wait for the previous command or chain of commands to finish. The IEEE 802.15.4 commands have exceptions for this rule.

When a radio operation command is finished, the radio CPU raises a COMMAND_DONE interrupt to the system CPU. If a number of commands are chained as explained previously, the COMMAND_DONE interrupt is raised after each command, while the LAST_COMMAND_DONE interrupt is raised after the last command in the chain. For one non-chained command, the LAST_COMMAND_DONE interrupt is also raised after the command. When LAST_COMMAND_DONE is raised, COMMAND_DONE is always raised at the same time. Before raising the COMMAND_DONE interrupt, the radio CPU updates the status field of the command structure to a status that indicates the command is finished. The radio CPU does not access the command structure after raising the COMMAND_DONE interrupt.

26.3.2.5.1 Triggers

Triggers can be used to set up a start time, or for other specific purposes in specific radio operation commands. A common trigger byte definition exists, as defined in Table 26-4.

Table 26-4. Format of Trigger Definition Byte

Bit Index	Field	Description
0–3	triggerType	The type of trigger
4	bEnaCmd	0: No alternative trigger command. 1: CMD_TRIGGER can be used as an alternative trigger.
5–6	triggerNo	The trigger number of the CMD_TRIGGER command that triggers this action.
7	pastTrig	0: A trigger in the past is never triggered, or for start of commands, give an error. 1: A trigger in the past is triggered as soon as possible.

The triggerType can take the values listed in Table 26-5. Other values are reserved.

Table 26-5. Supported Trigger Types

Number	Name	Description
0	TRIG_NOW	Now (not applicable to end triggers)
1	TRIG_NEVER	Never (except possibly by CMD_TRIGGER if bEnaCmd = 1)
2	TRIG_ABSTIME	At absolute time, given by timer parameter
3	TRIG_REL_SUBMIT	At a time relative to the time the command was submitted
4	TRIG_REL_START	At a time relative to start of this command (not allowed for start triggers)
5	TRIG_REL_PREVSTART	At a time relative to the start of the previous command
6	TRIG_REL_FIRSTSTART	At a time relative to the start of the first command of the chain
7	TRIG_REL_PREVEND	At a time relative to the end of the previous command
8	TRIG_REL_EVT1	At a time relative to event 1 of the previous command
9	TRIG_REL_EVT2	At a time relative to event 2 of the previous command
10	TRIG_EXTERNAL	On an external trigger input to the RAT

A 32-bit time parameter is used together with all triggers except for TRIG_NOW and TRIG_NEVER. Absolute timing uses the value of the 32-bit RAT. Relative timing uses the number of RAT ticks. The external trigger uses an identifier of source and edge, as defined in Table 26-6.

Table 26-6. Fields of Time Parameter for External Event Trigger

Bit Index	Field	Description
0–1		Reserved
2–3	inputMode	Input mode 00: Rising edge 01: Falling edge 10: Both edges 11: Reserved
4–7		Reserved
8–12	source	22: RFC_GPI0 23: RFC_GPI1 Others: Reserved
13–31		Reserved

Relative timing can either be relative to the time of submitting the command chain, to the start of the command, to the start of the previous or first command, or to certain observed events inside the command, to be defined for each command. The following rules apply:

- For the first command in a chain, if the start trigger is any of the types 5 through 9, the start is immediate. If another trigger referenced in the first command in a chain is any of the types 5 through 9, the trigger time is relative to the time the command was submitted.
- If the start trigger of a command is TRIG_REL_START, an error is produced.
- If the start trigger of a command is TRIG_NEVER and bEnaCmd is 0, an error is produced.
- Some radio operation commands define events 1 and 2. These are context-dependent events that can be observed by the radio CPU. See the description of each command for a definition in that context. If undefined, these events are the time of the start of the command.

If bEnaCmd is 1, the action may also be triggered with a command (CMD_TRIGGER command, see Section 26.3.3.2.5). The triggerNo parameter identifies the trigger number of this command.

If a trigger occurs in the past when evaluated, the behavior depends on the pastTrig bit. If this bit is 0, the trigger does not occur, or for start triggers, an error is produced. If this bit is 1, the trigger occurs as soon as possible. If the pastTrig bit is 1 for start triggers, timing relative to the start of the command is relative to the programmed start time, not the actual start time.

For an external trigger, the radio CPU sets the RAT to use the selected input event as a one-capture trigger; the CPU then uses this capture interrupt to trigger the action. If the event occurs before the setup occurs, the event is not captured, and the pastTrig bit is ignored.

26.3.2.5.2 Conditional Execution

The execution of a command may be conditional on the result of the previous command. For each command, three results are possible:

- TRUE
- FALSE
- ABORT

The criteria are defined for each command. If not defined, the result is TRUE unless the command ended with an error, in which case the result is ABORT.

Each command structure contains a condition for running the next command. The condition byte is as given in Table 26-7. If the rule is COND_SKIP_ON_FALSE or COND_SKIP_ON_TRUE, the number of commands to skip is signaled in the nSkip field. If the number of skips is zero, rerun the same command. If the number of skips is one, run the next command in the chain. If the number of skips is two, run the command after the next, and so forth. If the rule is COND_NEVER and no previous commands use skipping, the next command pointer is ignored and may be NULL.

Table 26-7. Format of Condition Byte

Bit Index	Field Name	Description
0–3	rule	Rule for how to proceed, as defined in Table 26-8
4–7	nSkip	Number of skips + 1 if the rule involves skipping 0: Same 1: Next 2: Skip ext ...

Table 26-8. Condition Rules

Number	Name	Description
0	COND_ALWAYS	Always run next command (except in case of ABORT).
1	COND_NEVER	Never run next command (next command pointer can still be used for skip).
2	COND_STOP_ON_FALSE	Run next command if this command returned TRUE, stop if it returned FALSE.
3	COND_STOP_ON_TRUE	Stop if this command returned TRUE, run next command if it returned FALSE.
4	COND_SKIP_ON_FALSE	Run next command if this command returned TRUE, skip a number of commands if it returned FALSE.
5	COND_SKIP_ON_TRUE	Skip a number of commands if this command returned TRUE, run next command if it returned FALSE.

If execution is stopped, the radio CPU goes back to idle and no further commands are run until a new command is entered through the CMDR register. The LAST_COMMAND_DONE interrupt is raised.

If a command ends with the ABORT result, the execution ends regardless of the condition. The LAST_COMMAND_DONE interrupt is raised. An example of criterion for the ABORT result is that a CMD_ABORT command is issued.

26.3.2.5.3 Handling Before Start of Command

For all radio operation commands, the start trigger and condition code are checked before parsing the rest of the command. If the start trigger has an illegal trigger type (including TRIG_REL_START, which is not allowed for start triggers, and TRIG_NEVER in combination with no command trigger), the radio CPU sets the status field to ERROR_START_TRIG. If the condition field has an illegal value, the radio CPU sets the status field to ERROR_CONDITION. If the start trigger occurs in the past and startTrigger.pastTrig is 0, the radio CPU sets the status field to ERROR_PAST_START.

26.3.2.6 Command Data Structures

The data structures are listed in tables throughout this chapter. The Byte Index is the offset from the pointer to that structure. Multibyte fields are little endian, and 16-bit halfword or 32-bit word alignment as given by the field size is required. For bit numbering, 0 is the LSB. The R/W column is used as follows:

R: The system CPU can read a result back; the radio CPU does not read the field.

W: The system CPU writes a value, the radio CPU reads it and does not modify it.

R/W: The system CPU writes an initial value, the radio CPU may modify it.

For data structures that are a specialization of another data structure, the fields from the parent structure are not repeated, but the Byte Index column reflects their presence.

The only mandatory field for all commands is the command ID number, which is a 16-bit number sent as the first 2 bytes of the command structure.

Some immediate commands have additional fields, which are defined for each command. The radio operation commands have additional mandatory fields as defined in Table 26-9.

All command fields marked as *Reserved* should be written to 0.

26.3.2.6.1 Radio Operation Command Structure

Table 26-9 lists the command structure for radio operation commands. Some commands have additional fields appended after this.

Table 26-9. Radio Operation Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	status			R/W	An integer telling the status of the command. This value is updated by the radio CPU during operation and may be read by the system CPU at any time.
4–7	pNextOp			W	Pointer to the next operation to run after this operation is done
8–11	startTime			W	Absolute or relative start time (depending on the value of the startTrigger field)
12	startTrigger				Identification of the trigger that starts the operation
13	condition			W	Condition for running next operation

26.3.2.7 Data Entry Structures

A data entry must belong to a queue. The queues are set up as part of the command structure of a radio operation command.

Operations on queues available as commands are described in Section 26.3.4.

26.3.2.7.1 Data Entry Queue

Any command that uses a queue contains a pointer to a data entry queue structure, as given in Table 26-10. The system CPU allocates and initializes this queue structure.

Table 26-10. Data Entry Queue Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pCurrEntry			R/W	Pointer to the data entry currently in use by the radio CPU (or next in line to be used if the radio is not using the queue). NULL means that no buffer is currently in the queue.
4–7	pLastEntry			R/W	Pointer to the last entry entered in this queue. If pCurrEntry is nonNULL and pLastEntry is NULL, further entries may not be appended.

26.3.2.7.2 Data Entry

A data entry queue contains data entries of the type listed in Table 26-11. These entries are organized in a linked list. The first entry of the queue is pointed to by the pCurrEntry field of the queue structure (see Table 26-10). Each pNextEntry field points to the next entry. The last entry in the queue is also pointed to by the pLastEntry field of the queue structure.

Table 26-11. General Data Entry Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pNextEntry			R/W	Pointer to next entry in the queue, NULL if this is the last entry
4	status			R/W	Indicates status of entry, including whether it is free to receive a write from the system CPU
5	config	0–1	type	W	Type of data entry structure 0: General data entry 1: Reserved 2: Pointer entry 3: Partial read RX entry
		2–3	lenSz	W	Size of length word in start of each RX entry element 0: No length indicator 1: 1-byte length indicator 2: 2-byte length indicator 3: Reserved
		4–7	irqIntv	W	For partial read RX entry only: The number of bytes between interrupt generated by the radio CPU (0000: 16 bytes)
6–7	length			W	Length of data field, or for pointer entries, of the data buffer. For TX entries, this corresponds to one entry element (packet). For RX entries, this gives the total available storage space.
8–(7+n)	data			R/W	Array of data to be received or transmitted (n = length)

The status field can take the following values:

- 0: Pending
  - The entry is not yet in use by the radio CPU. This is the status to write by the system CPU before submitting the entry.
- 1: Active
  - The entry is the entry in the queue currently open for writing (RX) or reading (TX) by the radio CPU.
- 2: Busy
  - An ongoing radio operation is writing or reading an unfinished packet. Certain operations are not allowed while an entry is in this state (see Section 26.3.4).
- 3: Finished
  - The radio CPU is finished writing data into this entry, and is free for the system CPU to reuse or free memory (if dynamically allocated).

For data entries, the system CPU sets up the required data structure, either in system RAM or in the available part of the radio RAM. If the data structure is dynamically allocated, the system CPU frees the memory after use.

In an entry is being used for received data, the radio CPU may start the entry element with a length indicator. If config.lenSz is 00, no such indicator is written. This option must only be used if the length of the received packet can be determined by other means. If config.lenSz is 01, 1 byte indicates the number of bytes following the length byte. This may only be used if no element of more than 255 bytes is written to the entry. If config.lenSz is 10, a 16-bit word indicates the number of bytes following the length word.

26.3.2.7.3 Pointer Entry

A pointer entry is an entry where the data are not contained in the entry itself, but the entry holds a pointer to the buffer. Such an entry is indicated by setting config.type to 2. The pointer replaces the data field, as shown in Table 26-12.

Table 26-12. Pointer Field in Pointer Entry Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
8–11	pData			W	Pointer to data buffer of size length bytes

The data is read from or stored in the buffer given by pData. The size of this buffer is given by length, just as for the size of the data field in a general data entry.

26.3.2.7.4 Partial Read RX Entry

Proprietary mode supports an RX entry where the data can be read before the entire packet is received over the air, which can be used for the following purposes:

- When data must be read before the entire packet is received
- When the length of the packet is not known in the beginning of the packet
- When the length of the packet is too long for the entire payload to be kept in memory simultaneously

To support this, a special variant of the structure in Table 26-11 is used. Several entry elements may be contained in the same entry. Each entry element corresponds to one packet received over the air, or part of it. The element may also contain additional fields. This type is selected by setting config.type to 3. In the entry, the data field is composed as shown in Table 26-13 (the indexes are relative to the entire entry structure).

Table 26-13. Fields in a Partial Read RX Entry

Byte Index	Byte Field Name	Bits	Bit Field Name	Type	Description
8–9	pktStatus	0–12	numElements	R	Number of entry elements committed in the entry
		13	bEntryOpen	R	The entry contains an element that is still open for appending data.
		14	bFirstCont	R	The first element is a continuation of the last packet from the previous entry.
		15	bLastCont	R	The packet in the last element continues in the next entry.
10–11	nextIndex			R	Index to the byte after the last byte of the last entry element committed by the radio CPU
12–(7+n)	rxData			R	Data received. Exact format depends on operation being run. Each entry element may start with a length byte or word.

The entry is updated as follows:

- The nextIndex field is updated as new bytes are written to the buffer.
- While a packet is being received, pktStatus.bEntryOpen is set to 1 by the radio CPU.

When an entry element is finished, either because the packet ended or because the element reached the end of the entry, pktStatus.bEntryOpen is set to 0 by the radio CPU, and pktStatus.numElements is incremented. If the packet continues in the next entry, pktStatus.bLastCont is set to 1 by the radio CPU. In this case, the pktStatus.bFirstCont bit of the next entry is also set to 1 by the radio CPU. If no next entry is available, the status is set to Unfinished, otherwise it is set to Finished.

The length field specified in the beginning of an entry element (depending on config.lenSz) gives the length of that entry element within the entry, not the entire packet. If the length is not known when the entry is opened,

the length field is written to the remaining length of the entry and updated by the radio CPU before the entry is finished.

For a partial read RX entry, the radio CPU generates an Rx_Data_Written interrupt to the system CPU whenever one or more bytes are written to the entry. In addition, it generates an Rx_N_Data_Written interrupt when k bytes have been written since the last interrupt or the start of the entry element, where k is given by config.irqIntv.

26.3.2.8 External Signaling

The radio CPU controls the signals CPEGPO0 and CPEGPO1. For control of an external front end, CPEGPO0 is high when the LNA must be enabled and low otherwise. CPEGPO1 is high when the PA must be enabled and low otherwise. The radio CPU also controls the signal CPEGPO2 to go high when synthesizer calibration starts, and low when the calibration is done. This control can be used for debugging.

Two of the output signals from the radio timer have automatic configuration that may be used for observation. The signal RATGPO0 goes high when transmission of a packet is initiated and low when transmission is done. The RATGPO0 signal may be observed for accurate timing of packet transmission, as the same signal is used internally. The signal is very similar to CPEGPO1, but it goes high some microseconds earlier, and the timing is more accurate compared to the first transmitted symbol out of the modem. However, CPEGPO1 is recommended for control of external PA to avoid turning it on too early. The signal RATGPO1 may be configured to go high when sync is found in the receiver, and low when the packet is received or reception aborted (this does not work for the IEEE 802.15.4 receiver command). To map RATGPO1 to a RFC_GPO signal, an additional override is needed. The other radio timer outputs may be configured to generate events as required, using CMD_SET_RAT_OUTPUT.

By default, the radio CPU maps CPEGPO0 to the signal RFC_GPO0, CPEGPO1 to the signal RFC_GPO1, CPEGPO2 to the signal RFC_GPO2, and RATGPO0 to the signal RFC_GPO3 at boot time. This mapping can be modified by writing to the RFC_DBELL:SYSGPOCTL register.

The RFC_GPO signals can be mapped to output pins using the system I/O controller.

26.3.3 Command Definitions

There is a set of commands independent of the current RF protocol. These commands are related to the low-level operations of the radio.

26.3.3.1 Protocol-Independent Radio Operation Commands

For radio operation commands described in the subsections that follow (Section 26.3.3.1.1 through Section 26.3.3.1.14), the operation ends due to one of the causes listed in Table 26-14 or by additional statuses listed for each command. After the operation has ended, the status field of the command structure indicates the reason why the operation ended. In each case, it is indicated if the result is TRUE, FALSE, or ABORT (see Section 26.3.2.5.2). This result indicates whether to start the next command (if any) indicated in pNextOp, or to return to an IDLE state.

Table 26-14. End of Radio Operation Commands

Condition	Status Code	Result
Finished operation	DONE_OK	TRUE
Received CMD_STOP while waiting for start trigger	DONE_STOPPED	FALSE
Received CMD_ABORT	DONE_ABORT	ABORT
The start trigger occurred in the past with startTrigger.pastTrig = 0	ERROR_PAST_START	ABORT
Illegal start trigger parameter	ERROR_START_TRIG	ABORT
Illegal condition for next operation	ERROR_CONDITION	ABORT
Observed illegal parameter	ERROR_PAR	ABORT
Invalid pointer to next operation	ERROR_POINTER	ABORT
Next operation has a command ID that is undefined or not a radio operation command	ERROR_CMDID	ABORT
Operation using RX, TX or synthesizer attempted without CMD_RADIO_SETUP	ERROR_NO_SETUP	ABORT
Operation using RX or TX attempted without the synthesizer being programmed	ERROR_NO_FS	ABORT

26.3.3.1.1 CMD_NOP: No Operation Command

Command ID number: 0x0801

CMD_NOP is a radio operation command that only takes the mandatory arguments listed in Table 26-9. The command only waits for the start trigger, and then ends. The command can be used to test the communication between the system CPU and the radio CPU or to insert a wait.

26.3.3.1.2 CMD_RADIO_SETUP: Set Up Radio Settings Command

Command ID number: 0x0802

CMD_RADIO_SETUP is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-15.

Table 26-15. CMD_RADIO_SETUP Command Format

Byte Index	Field	Bit Index	Bit Field name	Type	Description
14	mode			W	This is the main mode to use. 0x00: Bluetooth® Low Energy 0x01: IEEE 802.15.4 0x02: 2-Mbps GFSK 0x05: 5-Mbps coded 8-FSK 0xFF: Keep existing mode; update overrides only.
15	loDivider			W	Divider setting to use. Refer to Smart RF™ Studio or SysConfig for the recommended settings per device and band. Range is limited for some chip versions.
16–17	config	0–2	frontEndMode		0x00: Differential mode 0x01: Single-ended mode RFP 0x02: Single-ended mode RFN Others: Reserved
		3	biasMode	W	0: Internal bias 1: External bias
		4-9	analogCfgMode	W	0x00: Write analog configuration. Required first time after boot and when changing frequency band or front-end configuration. 0x2D: Keep analog configuration. May be used after standby or when changing mode with the same frequency band and front-end configuration. Others: Reserved
		10	bNoFsPowerup	W	0: Power up frequency synthesizer. 1: Do not power up frequency synthesizer.
		11	lnaIbBoost		Use setting from Smart RF™ Studio or SysConfig
		12	bSynthNarrowBand		Use setting from Smart RF™ Studio or SysConfig
		13–15			Reserved
18–19	txPower			W	Output power setting; use value from Smart RF™ Studio or SysConfig. For more details, see Section 26.3.3.2.16. 0xFFFF: Use 20 dBm PA
20–23	pRegOverride			W	Pointer to a list of hardware and configuration registers to override. If NULL, no override is used.

On start, the radio CPU sets up parameters for the operational mode given by mode.radioMode, with the modifications given in pRegOverride, a pointer to a structure containing override values for certain hardware registers, radio configuration controlled by the radio CPU, and protocol-related variables. If pRegOverride is NULL, no registers are overridden. The override value structure is a string of 32-bit entries provided by TI or produced by Smart RF™ Studio or SysConfig.

Running CMD_RADIO_SETUP or another radio setup command is mandatory before using any command that uses the receiver, transmitter, or frequency synthesizer. If the RF core is reset, CMD_RADIO_SETUP must be re-run.

When CMD_RADIO_SETUP is executing, trim values are read from FCFG1 unless they have been provided elsewhere. If these values are read from FCFG1, the following limitations apply while CMD_RADIO_SETUP is executing:

The VIMS module must be powered, allowing flash reads.

If the previously defined limitation is violated, the internal system bus might end up in a nonresponsive state. A system reset is required to exit this state. To ensure that FCFG1 can be read while running CMD_RADIO_SETUP, the TI provided RF driver automatically enables VIMS while running a radio setup command.

Table 26-16 lists the format of a hardware register override entry. Table 26-17 lists the format of array initiator. Table 26-18 lists the format of an ADI register override entry. Table 26-19 lists the format of a firmware-defined parameter override entry. Table 26-20 lists the format of an MCE/RFE override mode entry. Table 26-21 lists the format of a center frequency entry.

The txPower parameter is stored and applied every time transmission of a packet starts to set an output power with temperature compensation. This setting can be changed later with the command CMD_SET_TX_POWER (see Section 26.3.3.2.16). If txPower is 0xFFFF, the 20 dBm PA is selected instead of the normal one. This setting is only allowed on a "P" device. If this setting is used, a 20 dBm PA power override as defined in Table 26-22 is mandatory; if missing the setup command will end with error.

Table 26-16. Format of a Hardware Register Override Entry

Bit Index	Bit Field Name	Description
0–1	entryType	00: Hardware register 01: Array initiator, see Table 26-17 10: ADI register, see Table 26-18, or MCE/RFE override 11: Firmware defined parameter, see Table 26-19
2–15	hwAddr	Bits 2–15 of the address to the hardware register. Bits 0–1 of the address are 0.
16–31	value	The value to write to the register

Table 26-17. Format of Array Initiator

Bit Index	Bit Field Name	Description
0–1	entryType	01: Array initiator
2–15	startAddr	First address or index to write to: Hardware registers: Bits 2–15 of the address (bits 0–1 are 0) ADI registers: ADI bus address, half-byte indicator in bit 6, ADI selector in bit 7 Firmware-defined parameters: Byte Index
16–29	length	Number of entries
30–31	arrayType	Type of array: 00: Hardware registers with 16-bit values 01: Hardware registers with 32-bit values 10: ADI registers 11: Firmware-defined parameters

Table 26-18. Format of an ADI Register Override Entry

Bit Index	Bit Field Name	Description
0–1	entryType	10: ADI register
2–9	adiValue2	Optional second value to write
10–15	adiAddr2	Optional second ADI bus address
16–23	adiValue	Value to write to register
24–29	adiAddr	ADI bus address
30	bHalfSize	0: Use full-size writes 1: Use half-size writes, causing read-modify-write functionality
31	adiNo	0: Write to ADI 0 (RF) 1: Write to ADI 1 (synthesizer)

Table 26-19. Format of a Firmware-Defined Parameter Override Entry

Bit Index	Bit Field Name	Description
0–1	entryType	11: Firmware-defined parameter
2–3	entrySubType	00: Firmware-defined parameter 01: MCE/RFE override mode (must be in first entry), see Table 26-20 10: Reserved 11: End of override list
4–14	fwAddr	Byte index into parameter structure
15	bByte	0: 16-bit value 1: 8-bit value
16–31	value	The value to write to the parameter

Table 26-20. Format of an MCE/RFE Override Mode Entry

Bit Index	Bit Field Name	Description
0–1	entryType	11: Firmware-defined parameter
2–3	entrySubType	01: MCE/RFE override mode
4	bMceCopyRam	If 1, copy the contents of the MDM ROM bank given by mceRomBank to RAM after MCE has completed setup.
5	bRfeCopyRam	If 1, copy the contents of the RFE ROM bank given by rfeRomBank to RAM after MCE has completed setup.
6	bMceUseRam	0: Run MCE from ROM 1: Run MCE from RAM
7–10	mceRomBank	MCE ROM bank to run from
11	bRfeUseRam	0: Run RFE from ROM 1: Run RFE from RAM
12–15	rfeRomBank	RFE ROM bank to run from
16–23	mceMode	Mode to send to MCE
24–31	rfeMode	Mode to send to RFE

Table 26-21. Format of a Center Frequency Entry

Bit Index	Bit Field Name	Description
0–1	entryType	11: Firmware-defined parameter
2–3	entrySubType	10: Special configuration
4–7	specialType	0100: New center frequency entry
8-17		Reserved
18	bAutoTxIf	If 1, set Tx IF to Rx IF
19	bApplyRx	If 1, use centerFreq to re-calculate Rx IF
20-31	centerFreq	Center frequency in MHz

Table 26-22. 20 dBm PA TX Power Entry

Bit Index	Bit Field Name	Description
0–1	entryType	11: Firmware defined parameter
2–3	entrySubType	10: Special configuration
4–7	specialType	0010: 20 dBm PA TX power entry
8–9		Reserved
10–15	IB	New TX power setting. TI recommends using values from the Smart RF™ Studio or SysConfig. Value to write to the PA power control field at 25 °C. See Equation 14 for details.
16–17	ibBoost	Value to write to the bias control field of the PA
18	boost	Driver strength into the PA 0: Low driver strength 1: High driver strength
19–25	tempCoeff	Temperature coefficient for IB 0: No temperature compensation
26–31	paLdoTrim	Value to write to the output voltage control of the 20 dBm PA LDO

Table 26-23. Format of an End of List Entry

Bit Index	Bit Field Name	Description
0–1	entryType	11: Firmware-defined parameter
2–3	entrySubType	11: End of list segment
4–7	nextEntryRegion	0x0: End of list 0x1: SRAM. Base = 0x2000 0000 0x2: RF core RAM. Base = 0x2100 0000 0x3: Flash. Base = 0xA000 0000 0xF: End of list Others: Reserved
8–31	addrOffset	Address offset for next list part. Next address is: Base + (addrOffset × 4)

Use the Code Export feature in Smart RF™ Studio or SysConfig to generate an override list for the different PHYs and settings.

If the pointer in pRegOverride is invalid, any override entry is invalid. If the length of an array is too large or zero, the operation ends with the status ERROR_PAR. If config.bNoFsPowerup = 0 and powering up the synthesizer fails, the command ends with ERROR_SYNTH_PROG as the status.

If CMD_ABORT or CMD_STOP is received while waiting for the start trigger, the operation ends without any setup. If CMD_STOP is received after the start trigger, setup proceeds until finished. If CMD_ABORT is received after the start trigger, the setup process is aborted. This leaves the registers in an incomplete state, so another CMD_RADIO_SETUP command must be issued before using the radio.

26.3.3.1.3 CMD_FS_POWERUP: Power Up Frequency Synthesizer

Command ID number: 0x080C

CMD_FS_POWERUP is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-24.

On start, the radio CPU powers up the frequency synthesizer and applies the register modifications given in pRegOverride. If pRegOverride is NULL, no registers are overridden. The format of the override structure is the same as the format for CMD_RADIO_SETUP (see Section 26.3.3.1.2). Only overrides applicable to the synthesizer hardware are applied.

Running CMD_FS_POWERUP is mandatory before using any command that uses the frequency synthesizer (and thus, the transmitter or receiver), unless the synthesizer is powered up as part of the radio setup. The radio must be set up using CMD_RADIO_SETUP or another setup command before CMD_FS_POWERUP.

If the pointer in pRegOverride is invalid, the address or index is invalid, the length of an array is zero or is too large. If another parameter in an entry is not permitted, the operation ends with the status ERROR_PAR. If powering up the synthesizer fails, the command ends with ERROR_SYNTH_PROG as the status. When otherwise finished, the command ends with DONE_OK as the status.

Table 26-24. CMD_FS_POWERUP Command Format

Byte Index	Field Name	Bit Index	Bit Field Name	Type	Description
14–15					Reserved
16–19	pRegOverride			W	Pointer to a list of hardware and configuration registers to override. If NULL, no override is used.

26.3.3.1.4 CMD_FS_POWERDOWN: Power Down Frequency Synthesizer

Command ID number: 0x080D

CMD_FS_POWERDOWN is a radio operation command that only takes the mandatory arguments listed in Table 26-9. The command waits for the start trigger and then powers down the synthesizer. The act of powering down not only stops the synthesizer, as is done with CMD_FS_OFF (see Section 26.3.3.1.6) or at the end of certain other radio operation commands, but it also switches off analog modules.

After running CMD_FS_POWERDOWN, the synthesizer must be powered up again using CMD_FS_POWERUP, or another command that powers up the synthesizer before it is used.

CMD_FS_POWERDOWN must always be run before the radio is powered down (for instance, when the device is going into low-power modes).

When finished, the CMD_FS_POWERDOWN command ends with a DONE_OK status.

26.3.3.1.5 CMD_FS: Frequency Synthesizer Controls Command

Command ID number: 0x0803

CMD_FS is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-25, and can program the synthesizer to a specific frequency.

The frequency to use is given by frequency and fractFreq, and must be as close as possible to (frequency + fractFreq / 65536) MHz.

The synthesizer is set up in RX mode or TX mode, depending on synthConf.bTxMode. This mode may be changed by radio operation commands when setting up RX or TX. If synthConf.refFreq is nonzero, a reference frequency of 48 MHz / synthConf.refFreq is used instead of the default one.

If the synthesizer is programmed and reports loss of lock after being in lock, the radio CPU raises the Synth_No_Lock interrupt. The synthesizer keeps running, but the system CPU may use this information to stop and restart the radio. The Synth_No_Lock is not raised more than once for each time the synthesizer is programmed. The interrupt may also occur for commands with implicit-frequency programming.

If the command is called with an illegal frequency or divider setting, the command ends with ERROR_PAR as the status. If the command is called without the radio being configured, it ends with ERROR_NO_SETUP as the status. If the command is called without the synthesizer being powered up, it ends with ERROR_NO_FS as status.

Table 26-25. CMD_FS Command Format

Byte Index	Field Name	Bit Index	Bit Field Name	Type	Description
14–15	frequency			W	The frequency in MHz to which the synthesizer should be tuned
16–17	fractFreq			W	Fractional part of the frequency to which the synthesizer should be tuned
18	synthConf	0	bTxMode	W	0: Start synthesizer in RX mode. 1: Start synthesizer in TX mode.
		1–6	refFreq	W	0: Use default reference frequency. Others: Use reference frequency 48 MHz/refFreq.
		7	Reserved	W	Reserved
19–23			Reserved	W	Reserved

26.3.3.1.6 CMD_FS_OFF: Turn Off Frequency Synthesizer

Command ID number: 0x0804

CMD_FS_OFF is a radio operation command that only takes the mandatory arguments listed in Table 26-9, and turns off the frequency synthesizer if it has been started by CMD_FS or left on by a radio operation command that does not turn off the synthesizer.

When the command is started, the synthesizer outputs are disabled and the state machine is reset. The analog parts are still powered; CMD_FS_POWERDOWN (see Section 26.3.3.1.4) can power down the synthesizer to further reduce the current consumption.

26.3.3.1.7 CMD_RX_TEST: Receiver Test Command

Command ID number: 0x0807

CMD_RX_TEST is a radio operation command used to set the receiver in infinite RX mode for test purposes.

The sync word programmed in the receiver is given in the LSBs of syncWord. If config.bNoSync is 1, the correlation thresholds for sync search are set to the maximum value to avoid getting sync. The thresholds are restored after the command ends.

If pktConfig.bFsOff is 1, the synthesizer is turned off (corresponding to CMD_FS_OFF; see Section 26.3.3.1.6) after the operation is done; otherwise the synthesizer is left on.

A trigger to end the operation is set up by endTrigger and endTime (see Section 26.3.2.5.1). If the trigger that is defined by this parameter occurs, the radio operation ends.

The operation ends by one of the causes listed in Table 26-14.

The command structure for CMD_RX_TEST contains the fields listed in Table 26-26.

Table 26-26. CMD_RX_TEST Command Format

Byte Index	Byte Field Name	Bits	Bit Field Name	Type	Description
14	config	0	Reserved	W	Set to 0
		1	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		2	bNoSync	W	0: Run sync search as normal for the configured mode. 1: Write correlation thresholds to the maximum value to avoid getting sync.
15	endTrigger			W	Trigger classifier for ending the operation
16–19	syncWord			W	Sync word to use for receiver
20–23	endTime			W	Time to end the operation

26.3.3.1.8 CMD_TX_TEST: Transmitter Test Command

Command ID number: 0x0808

CMD_TX_TEST is a radio operation command used to set the receiver in infinite TX mode and transmit either a CW or modulated data for test purposes.

When starting, the radio CPU starts the transmitter. The radio must be configured with the CMD_RADIO_SETUP and the radio synthesizer must be started and in TX mode with the CMD_FS radio command before CMD_TX_TEST is issued. The radio transmits a preamble and a sync word of the size given by the current radio configuration, specified using CMD_RADIO_SETUP. The sync word is given in the LSBs of syncWord. The payload after the sync word consists of the 16-bit word txWord, repeated indefinitely. This word may be run through whitening, with the options given in config.whitenMode, which can take the following values:

- 0: No whitening is used
  - This value is useful for testing with a repeated pattern, but gives spurs if used for spectral measurements.
- 1: Default whitening
  - This value means that the whitening is as configured for the mode in use (potentially with overrides). If the mode does not use whitening, no whitening is applied.
- 2: PRBS-15
  - The polynomial $x^{15} + x^{14} + 1$ is used. This value gives a pseudo-noise sequence with length 32767.
- 3: PRBS-32
  - The polynomial $x^{32} + x^{22} + x^2 + x + 1$ is used. This value gives a pseudo-noise sequence with length 4294967295.

For whitening modes 2 and 3, initialization is done by the radio CPU writing 0xAAAA0000 to the PRBS value register before transmission starts. For whitening mode 1, the default initialization is used.

The transmitter runs until the trigger set up by endTrigger and endTime (see Section 26.3.2.5.1) occurs, or until an abort command is issued.

If pktConfig.bFsOn is 1, the synthesizer is turned off (corresponding to CMD_FS_OFF; see Section 26.3.3.1.6) after the operation is done; otherwise it is left on.

The operation ends by one of the causes listed in Table 26-14.

The command structure for CMD_TX_TEST contains the fields listed in Table 26-27.

Table 26-27. CMD_TX_TEST Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	config	0	bUseCw	W	0: Send modulated signal. 1: Send continuous wave.
		1	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		2–3	whitenMode	W	0: No whitening 1: Default whitening 2: PRBS-15 3: PRBS-32
15					Reserved
16–17	txWord			W	Value to send to the modem before whitening
18					Reserved
19	endTrigger			W	Trigger classifier for ending the operation
20–23	syncWord			W	Sync word to use for transmitter
24–27	endTime			W	Time to end the operation

26.3.3.1.9 CMD_SYNC_STOP_RAT: Synchronize and Stop Radio Timer Command

Command ID number: 0x0809

CMD_SYNC_STOP_RAT is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-28.

For more details, see Chapter 18.

AON_RTC:CTL.RTC_UPD_EN must be 1.

Table 26-28. CMD_SYNC_STOP_RAT Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15					Unused
16–19	rat0			R	The returned RAT value corresponding to the value the RAT would have had when the RTC was zero.

When starting, the radio CPU sets up capture of an RTC tick and waits for this tick, then stops the RAT and calculates the value rat0. This value must be stored for use when the RAT restarts.

26.3.3.1.10 CMD_SYNC_START_RAT: Synchronously Start Radio Timer Command

Command ID number: 0x080A

CMD_SYNC_START_RAT is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-29.

For more details, see Chapter 18.

TOP:AON_RTC:CTL.RTC_UPD_EN must be 1.

Table 26-29. CMD_SYNC_START_RAT Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15					Unused
16–19	rat0			W	The desired RAT value corresponding to the value the RAT would have had when the RTC was zero. This parameter is returned by CMD_SYNC_STOP_RAT.

When starting, the radio CPU starts the RAT and sets up capture of an RTC tick and waits for this tick, then calculates the necessary timer adjustment based on the input parameter rat0 and performs this adjustment. The input parameter rat0 is the value previously returned by CMD_SYNC_STOP_RAT.

Because the radio timer is normally not running when this command is issued, the start trigger must be TRIG_NOW (see Section 26.3.2.5.1).

The first time the RAT is started after system boot, the command CMD_START_RAT must be used (see Section 26.3.3.2.7). As an alternative, CMD_SYNC_START_RAT may be issued with a fixed parameter such as 0; however, this gives an arbitrary start value for the RAT. Before powering down the radio, the system CPU must run the CMD_SYNC_STOP_RAT command. After powering it up the radio again, the system CPU must run the CMD_SYNC_START_RAT command with the same parameter as the one received when CMD_SYNC_STOP_RAT was issued.

26.3.3.1.11 CMD_COUNT: Counter Command

Command ID number: 0x080B

CMD_COUNT is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-30.

Table 26-30. CMD_COUNT Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	counter			R/W	Counter. When starting, the radio CPU decrements the value, and the end status of the operation differs if the result is zero.

When starting, the radio CPU decrements the counter field by 1 and writes the result back to this field. If the result of the decrement is zero, the operation ends with the status DONE_COUNTDOWN and the result FALSE. Otherwise, the operation ends with the status DONE_OK and the result TRUE, which can be used in conditional execution to create a loop.

If the operation is started with counter equal to zero, this is an illegal parameter, so the operation ends with the status ERROR_PAR.

The operation ends by one of the causes listed in Table 26-26 or Table 26-31.

Table 26-31. Additional End Causes for CMD_COUNT

Condition	Status Code	Result
Finished operation with counter > 0	DONE_OK	TRUE
Finished operation with counter = 0	DONE_COUNTDOWN	FALSE

26.3.3.1.12 CMD_SCH_IMM: Run Immediate Command as Radio Operation

Command ID number: 0x0810

CMD_SCH_IMM is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-32.

Table 26-32. CMD_SCH_IMM Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15					Reserved
16–19	cmdrVal			W	Value as would be written to CMDR
20–23	cmdstaVal			R	Value as would be returned in CMDSTA

When starting, the radio CPU takes the value in cmdrVal and processes it as if it had been written to the CMDR register. This command may be a pointer to an immediate command, or it may form a direct command as shown in Figure 26-4. A pointer to a radio operation command causes a scheduling error. The value that would normally have been returned in CMDSTA is written to cmdstaVal, which means that no command RF_CMD_ACK interrupt is raised. Instead, a COMMAND_DONE interrupt is raised, as for any other radio operation command.

Depending on the result of the immediate or direct command, the status and result of the radio operation command is as shown in Table 26-33.

CMD_SCH_IMM may run immediate commands as part of a chain of radio operation commands, or to schedule them in the future. If an immediate or direct command received in the CMDR register is being processed at the same time that a scheduled CMD_SCH_IMM command starts, the processing of the scheduled command starts after the other command has finished.

Table 26-33. End Statuses for CMD_SCH_IMM

Condition	Status Code	Result
Immediate command ended with the result Done	DONE_OK	TRUE
There was an error in the execution of the command, giving a RFC_DBELL:CMDSTA result that is not listed in the table row that follows.	DONE_FAILED	FALSE
There was an error in the command, giving one of the following results: • SchedulingError • UnknownCommand • UnknownDirCommand • IllegalPointer • ParError.	ERROR_PAR	ABORT

26.3.3.1.13 CMD_COUNT_BRANCH: Counter Command with Branch of Command Chain

Command ID number: 0x0812

CMD_COUNT_BRANCH is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-34.

Table 26-34. CMD_COUNT_BRANCH Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	counter			R/W	Counter When starting, the radio CPU decrements the value, and the end status of the operation differs if the result is zero.
16–19	pNextOpIfOk			W	Pointer to next operation if counter did not expire.

When starting, the radio CPU decrements the counter field by 1, unless it was already 0, and writes the result back to this field. If the result of the decrement is zero, the operation ends with the status DONE_COUNTDOWN and the result FALSE. Otherwise, the operation ends with the status DONE_OK and the result TRUE. In this case, the next radio operation command to run is given by pNextOpIfOk instead of pNextOp (see Table 26-9), which can be used in conditional execution to create a loop.

If the operation is started with the counter equal to zero, the operation ends with the status DONE_OK and the next operation is taken from pNextOpIfOk. This operation can be used if the previous command is set up to skip optionally, as skipping from a previous command in the chain follows pNextOp.

The operation ends by one of the causes listed in Table 26-14 or Table 26-35.

Table 26-35. Additional End Causes for CMD_COUNT_BRANCH

Condition	Status Code	Result
Finished operation with counter = 0 when being started	DONE_OK	TRUE
Finished operation with counter > 0 after decrementing	DONE_OK	TRUE
Finished operation with counter = 0 after decrementing	DONE_COUNTDOWN	FALSE

26.3.3.1.14 CMD_PATTERN_CHECK: Check a Value in Memory Against a Pattern

Command ID number: 0x0813

CMD_PATTERN_CHECK is a radio operation command. In addition to the parameters listed in Table 26-9, the command structure contains the fields listed in Table 26-36.

Table 26-36. CMD_PATTERN_CHECK Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	patternOpt	0–1	operation	W	Operation to perform: 0: TRUE if value == compareVal 1: TRUE if value < compareVal 2: TRUE if value > compareVal 3: Reserved
		2	bByteRev	W	If 1, interchange the 4 bytes of the value, so that they are read MSB first.
		3	bBitRev	W	If 1, perform bit reversal of the value.
		4–8	signExtend	W	0: Treat value and compareVal as unsigned. 1–31: Treat value and compareVal as signed, where the value gives the number of the MSB in the signed number.
		9	bRxVal	W	0: Use pValue as a pointer. 1: Use pValue as a signed offset to the start of the last committed RX entry element.
16–19	pNextOpIfOk			W	Pointer to next operation if comparison result was TRUE
20–23	pValue			W	Pointer from which to read, or offset from last RX entry if patternOpt.bRxVal == 1
24–27	mask			W	Bit mask to apply before comparison
28–31	compareVal			W	Value to which to compare

When starting, the radio CPU reads a 4-byte value from the location pointed to by pValue if patternOpt.bRxVal == 0. If patternOpt.bRxVal == 1, the location to read from is found by taking the pointer to the start of the last committed RX entry element and adding the signed number found in pValue as a byte offset. In either case, this pointer does not need to be 4-byte-aligned. If the pointer is not byte-aligned, the value is read byte by byte.

The value is then subject to the following operations in this order:

1. If patternOpt.bByteRev == 1, interchange byte 3 with byte 0, and byte 1 with byte 2, as if the bytes had been read MSB first.
2. If patternOpt.bBitRev == 1, reverse the bit order of the entire 32-bit word.
3. Perform a bitwise ‘AND’ operation between the value and mask.
4. If patternOpt.signExtend > 0, copy the value of bit number patternOpt.signExtend (where bit 0 is the LSB) into all the more significant bits.
5. Perform a compare operation between the resulting value and compareVal, depending on patternOpt.operation (see Table 26-36). The compare operation is unsigned if patternOpt.signExtend == 0; otherwise it is signed.

If patternOpt.operation or pValue have illegal values, the operation ends with a status of ERROR_PAR. Otherwise, the operation ends by one of the causes listed in Table 26-14 or Table 26-37, depending on the result of the comparison in Step 5 in the previous list. If the comparison result was TRUE, the next radio operation command to run is given by pNextOpIfOk instead of pNextOp.

Table 26-37. Additional End Causes for CMD_PATTERN_CHECK

Condition	Status Code	Result
Comparison result was TRUE.	DONE_OK	TRUE
Comparison result was FALSE.	DONE_FAILED	FALSE
Command run with patternOpt.bRxVal when no RX data is fully received	ERROR_NO_RX	ABORT

26.3.3.2 Protocol-Independent Direct and Immediate Commands

This section contains immediate commands that can be used across protocols. Section 26.3.4 describes the commands for manipulating data queues.

26.3.3.2.1 CMD_ABORT: ABORT Command

Command ID number: 0x0401

CMD_ABORT is a direct command.

On reception, the radio CPU ends ongoing radio operation commands as soon as possible. Analog circuitry for RX and TX is safely turned off, and data structures are updated so they are not left in an unfinished state.

If a radio operation command is running when the CMD_ABORT is issued, the radio CPU produces a COMMAND_DONE and LAST_COMMAND_DONE interrupt when the radio operation command finishes. The status of the command structure of that radio operation command reflects that the command was aborted.

If no radio operation command is running, no action is taken. The result signaled in the RFC_DBELL:CMDSTA register is Done in all cases. If a radio operation command is running, CMDSTA may be updated before the radio operation ends.

26.3.3.2.2 CMD_STOP: Stop Command

Command ID number: 0x0402

CMD_STOP is a direct command.

On reception, the radio CPU informs the radio operation command currently running that it is requested to stop. The CMD_STOP command is more graceful than the CMD_ABORT command, but might take more time to finish. Normally, a packet being received or transmitted is handled to completion. The exact behavior on reception of CMD_STOP is described for each radio operation command. Some commands always end in a known time and do not respond to CMD_STOP.

If no radio operation command is running, no action is taken. The result signaled in CMDSTA is done in all cases. If a radio operation command is running, CMDSTA may be updated before the radio operation ends.

26.3.3.2.3 CMD_GET_RSSI: Read RSSI Command

Command ID number: 0x0403

CMD_GET_RSSI is an immediate command that takes no parameters, and therefore, can be used as a direct command.

On reception, the radio CPU reads the RSSI from an underlying receiver. The RSSI is returned in result byte 2 (bit 23–16) of RFC_DBELL:CMDSTA (see Figure 26-5). The RSSI is given on signed form in dBm. If no RSSI is available, this is signaled with a special value of the RSSI (0x80).

If no radio operation command is running, the radio CPU returns the result ContextError in CMDSTA. Otherwise, the radio CPU returns Done along with the RSSI value.

26.3.3.2.4 CMD_UPDATE_RADIO_SETUP: Update Radio Settings Command

Command ID number: 0x0001

CMD_UPDATE_RADIO_SETUP is an immediate command that takes the parameters listed in Table 26-38.

Table 26-38. CMD_UPDATE_RADIO_SETUP Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	pRegOverride			W	Pointer to a list of hardware and configuration registers to override

On reception, the radio CPU updates the registers given in pRegOverride. This is a pointer to a structure containing an override value for certain hardware registers, a radio configuration controlled by the radio CPU, and protocol-related variables. The format is as for CMD_RADIO_SETUP (see Section 26.3.3.1). If done while the radio is running, the update must primarily be done on the radio and protocol configuration, as modifications to hardware registers may cause undesired behavior.

26.3.3.2.5 CMD_TRIGGER: Generate Command Trigger

Command ID number: 0x0404

CMD_TRIGGER is an immediate command that takes the parameters listed in Table 26-39.

Table 26-39. CMD_TRIGGER Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	triggerNo			W	Command trigger number

On reception, the radio CPU generates the command trigger specified with triggerNo, so that running radio operation commands respond accordingly (see Section 26.3.2.5.1).

If the trigger number is outside the valid range 0–3, the radio CPU returns the result ParError in CMDSTA. If no radio operation command running is pending on the trigger number sent, the radio CPU returns the result ContextError in CMDSTA. Otherwise, the radio CPU returns Done, which may be returned before the running radio operation command responds to the trigger.

CMD_TRIGGER may be sent as a direct command. If so, the trigger number is given by the parameter in bits 8–15 of CMDR.

26.3.3.2.6 CMD_GET_FW_INFO: Request Information on the Firmware Being Run

Command ID number: 0x0002

CMD_GET_FW_INFO is an immediate command that takes the parameters listed in Table 26-40.

Table 26-40. CMD_GET_FW_INFO Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	versionNo			R	Firmware version number
4–5	startOffset			R	The start of free RAM
6–7	freeRamSz			R	The size of free RAM
8–9	availRatCh			R	Bitmap of available RAT channels

On reception, the radio CPU reports information on the running radio firmware. A version number is returned in versionNo. The startOffset and freeRamSz fields contain information on the area in the non-retention part of the radio RAM that is not used by the radio CPU for data (including stack and heap). This area is free to use by the system CPU for data exchange, radio CPU patching, or other purposes. The start and end address of the free RAM is given as offset from the start of the radio RAM. The RF core retention RAM is free to use except for the first 36 bytes.

Note

Some of this free RAM is used for patches provided by TI.

The availRatCh field is a bitmap where bit position n indicates whether RAT channel n may be used by the system CPU. A bit value of 1 indicates that the corresponding channel may be used by the system CPU, while a bit value of 0 means that the channel is reserved for the radio CPU or is nonexistent.

26.3.3.2.7 CMD_START_RAT: Asynchronously Start Radio Timer Command

Command ID number: 0x0405

CMD_ START_RAT is a direct command.

On reception, the radio CPU starts the RAT if it has not already started.

If the RAT is already running, the radio CPU returns the result ContextError in RFC_DBELL:CMDSTA. Otherwise, the radio CPU returns Done.

26.3.3.2.8 CMD_PING: Respond with Interrupt

Command ID number: 0x0406

CMD_PING is a direct command.

On reception, the radio CPU returns Done in RFC_DBELL:CMDSTA. This command can test the communication between the two CPUs, or to check when the radio CPU is ready after boot.

26.3.3.2.9 CMD_READ_RFREG: Read RF Core Register

Command ID number: 0x0601

CMD_READ_RFREG is an immediate command that takes the parameters listed in Table 26-41.

Table 26-41. CMD_READ_RFREG Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	address			W	The offset from the start of the RF core hardware register bank (0x4004 0000)
4–7	value			R	Returned value of the register

On reception, the radio CPU reads the RF core register with address 0x40040000 + address. The result is written to value. If the address is not divisible by 4, the radio CPU returns ParError in RFC_DBELL:CMDSTA.

CMD_READ_RFREG may be sent as a direct command. If so, the address is given by bits 2–15 of CMDR, with the 2 LSBs of the address set to 00.

When reading is performed, the result is returned in value. The 24 LSBs of the result are returned in CMDSTA bits 8–31. The result returned in CMDSTA is Done.

26.3.3.2.10 CMD_SET_RAT_CMP: Set RAT Channel to Compare Mode

Command ID number: 0x000A

CMD_SET_RAT_CMP is an immediate command that takes the parameters listed in Table 26-42.

Table 26-42. CMD_SET_RAT_CMP Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	ratCh			W	The radio timer channel number
3					Reserved
4–7	compareTime			W	The time at which the compare occurs

On reception, the radio CPU sets the RAT channel given by ratCh in compare mode, and sets the channel compare time to compareTime, which also arms the channel. A channel event occurs at the given time, and this can be enabled as an RF hardware interrupt to the system CPU through the RFC_DBELL module.

The channel number must indicate a channel that is not reserved for use by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the compare time is in the past when the command is evaluated, the radio CPU returns ContextError in CMDSTA and disables the RAT channel. If the compare event is successfully set up, the radio CPU returns Done in CMDSTA.

26.3.3.2.11 CMD_SET_RAT_CPT: Set RAT Channel to Capture Mode

Command ID number: 0x0603

CMD_SET_RAT_CPT is an immediate command that takes the parameters listed in Table 26-43.

Table 26-43. CMD_SET_RAT_CPT Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	config	0–2			Reserved
		3–7	inputSrc	W	Input source indicator: 22: RFC_GPI0 23: RFC_GPI1 Others: Reserved
		8–11	ratCh	W	The radio timer channel number
		12	bRepeated	W	0: Single capture mode 1: Repeated capture mode
		13–14	inputMode	W	Input mode: 00: Rising edge 01: Falling edge 10: Both edges 11: Reserved

On reception, the radio CPU sets the RAT channel given by config.ratCh in capture mode. If config.bRepeated is 0, the channel is set to single capture mode; otherwise, the channel is set to repeated capture mode. The radio CPU sets the input source to config.inputSrc and the input mode to config.inputMode. If the channel is set in single capture mode, it is also armed. This causes a channel event to occur when capture occurs, and can be enabled as an RF hardware interrupt to the system CPU through the RFC_DBELL module.

CMD_SET_RAT_CMP may be sent as a direct command. If so, bits 2–15 of the config word are given by bits 2–15 of CMDR (bits 0–1 of config are not used).

The channel number must indicate a channel that is not reserved for use by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the channel is successfully set up, the radio CPU returns Done in CMDSTA.

26.3.3.2.12 CMD_DISABLE_RAT_CH: Disable RAT Channel

Command ID number: 0x0408

CMD_DISABLE_RAT_CH is an immediate command that takes the parameters listed in Table 26-44.

Table 26-44. CMD_DISABLE_RAT_CH Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	ratCh			W	The RAT channel number

On reception, the radio CPU disables the RAT channel given by ratCh. This disables previous configurations of that channel done by the CMD_SET_RAT_CMP or CMD_SET_RAT_CPT command.

CMD_DISABLE_RAT_CH may be sent as a direct command. If so, ratCh is given by the parameter in bits 8–15 of CMDR.

The channel number must indicate a channel that is not reserved for use by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the channel number is valid, the CPU returns Done in CMDSTA after the channel is disabled.

26.3.3.2.13 CMD_SET_RAT_OUTPUT: Set RAT Output to a Specified Mode

Command ID number: 0x0604

CMD_SET_RAT_OUTPUT is an immediate command that takes the parameters listed in Table 26-45.

Table 26-45. CMD_SET_RAT_OUTPUT Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	config	0–1			Reserved
		2–4	outputSel	W	Output event indicator: 1: RAT_GPO1 2: RAT_GPO2 3: RAT_GPO3 Others: Reserved
		5–7	outputMode	W	Output mode: 000: Pulse 001: Set 010: Clear 011: Toggle 100: Always 0 101: Always 1 Others: Reserved
		8–11	ratCh	W	The RAT channel number

On reception, the radio CPU sets the RAT output event given by config.outputSel in the mode given by config.outputMode, and to be controlled by the RAT channel given by config.ratCh. This command must be combined with setting this channel in compare mode, using the CMD_SET_RAT_CMP command.

CMD_SET_RAT_OUTPUT may be sent as a direct command. If so, bits 2–15 of the config word are given by bits 2–15 of CMDR (bits 0–1 of config are not used).

The channel number, config.ratCh, must indicate a channel that is not reserved for use by the radio CPU, and the output number, config.outputSel, must not be an output used by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the output event is successfully set up, the radio CPU returns Done in CMDSTA.

26.3.3.2.14 CMD_ARM_RAT_CH: Arm RAT Channel

Command ID number: 0x0409

CMD_ARM_RAT_CH is an immediate command that takes the parameters listed in Table 26-46.

Table 26-46. CMD_ARM_RAT_CH Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	ratCh			W	The RAT channel number

On reception, the radio CPU arms the RAT channel given by ratCh.

The CMD_DISABLE_RAT_CH command may be sent as a direct command. If so, ratCh is given by the parameter in bits 8–15 of CMDR.

The channel number must indicate a channel not reserved for use by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the channel number is valid, the CPU returns Done in CMDSTA after the channel is armed.

26.3.3.2.15 CMD_DISARM_RAT_CH: Disarm RAT Channel

Command ID number: 0x040A

CMD_DISARM_RAT_CH is an immediate command that takes the parameters listed in Table 26-47.

Table 26-47. CMD_DISARM_RAT_CH Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	ratCh			W	The RAT channel number

On reception, the radio CPU disarms the RAT channel given by ratCh.

CMD_DISABLE_RAT_CH may be sent as a direct command. If so, ratCh is given by the parameter in bits 8–15 of CMDR.

The channel number must indicate a channel not reserved for use by the radio CPU. Otherwise, the radio CPU returns ParError in RFC_DBELL:CMDSTA. If the channel number is valid, the CPU returns Done in CMDSTA after the channel is armed.

26.3.3.2.16 CMD_SET_TX_POWER: Set Transmit Power

Command ID number: 0x0010

CMD_SET_TX_POWER is an immediate command that takes the parameters listed in Table 26-48.

Table 26-48. CMD_SET_TX_POWER Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	txPower			W	New TX power setting. TI recommends using values from Smart RF Studio or SysConfig. Bits 0-5: IB Value to write to the PA power control field at 25 °C. See Equation 14 for details. Bits 6-7: GC Value to write to the gain control of the first stage of the PA. Bit 8: boost Driver strength into the PA. 0: Low driver strength 1: High driver strength Bits 9-15: tempCoeff Temperature coefficient for IB. 0: No temperature compensation.

$$\mathrm{IB} = IB_{25^{\circ}\mathrm{C}} + \frac{[\mathrm{Temperature}(^{\circ}\mathrm{C}) - 25] \times \mathrm{tempCoeff}}{256} \quad (14)$$

On reception, the radio CPU sets the transmit power for use the next time transmission begins. If a packet is being transmitted, the transmit power is not updated until transmission begins for the next packet.

Each time transmission of a packet begins, temperature compensation of the transmit power is done.

It is not allowed to use CMD_SET_TX_POWER if the 20 dBm PA was configured in the radio setup command; in this case, CMD_SET_TX20_POWER (see Section 26.3.3.2.17) should be used.

On completion, the radio CPU returns Done in RFC_DBELL:CMDSTA.

26.3.3.2.17 CMD_SET_TX20_POWER: Set Transmit Power of the 20 dBm PA

Command ID number: 0x0014

CMD_SET_TX20_POWER is an immediate command that takes the parameters listed in Table 26-49.

Table 26-49. CMD_SET_TX20_POWER Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	txPower	0–5	IB	W	New TX power setting. TI recommends using values from Smart RF™ Studio or SysConfig. Value to write to the PA power control field at 25 °C. See Equation 14 for details.
		6–7	ibBoost	W	Value to write to the bias control field of the PA
		8	boost	W	Driver strength into the PA 0: Low driver strength 1: High driver strength
		9-15	tempCoeff	W	Temperature coefficient for IB 0: No temperature compensation
		16–21	paLdoTrim	W	Value to write to the output voltage control of the 20 dBm PA LDO
		22–31			Reserved

On reception, the radio CPU will set the transmit power and temperature coefficient for the 20 dBm PA for use the next time transmission is started. If a packet is being transmitted, the transmit power will not be updated until transmission starts for the next packet.

Each time transmission of a packet begins, temperature compensation of the transmit power is done.

It is not allowed to use CMD_SET_TX20_POWER if the 20 dBm PA was not configured in the radio setup command; in this case, CMD_SET_TX_POWER (see Section 26.3.3.2.16) should be used.

The radio CPU returns Done in RFC_DBELL:CMDSTA when finished.

26.3.3.2.18 CMD_MODIFY_FS: Set New Synthesizer Frequency Without Recalibration

Command ID number: 0x0013

CMD_MODIFY_FS is an immediate command that takes the parameters listed in Table 26-50.

Table 26-50. CMD_MODIFY_FS Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	frequency			W	The frequency in MHz to which to tune
4–5	fractFreq			W	Fractional part of the frequency to which to tune

On reception, the radio CPU will program a new frequency in the synthesizer without restarting calibration. This has to be a small change compared to the frequency used under calibration, otherwise the synthesizer will most likely be unable to relock. Extra distortion may occur if the command is done during RX or TX.

The frequency to use is given by frequency and fractFreq, and the frequency will be as close as possible to (frequency + fractFreq / 65536) MHz.

If the synthesizer is not running and the calibration is done, the radio CPU will return ContextError in CMDSTA. If frequency is invalid, the radio CPU will return ParError in RFC_DBELL:CMDSTA. Otherwise, the radio CPU will return Done in CMDSTA when the update is complete.

26.3.3.2.19 CMD_BUS_REQUEST: Request System BUS Available for RF Core

Command ID number: 0x040E

CMD_BUS_REQUEST is an immediate command that takes the parameters listed in Table 26-51.

Table 26-51. CMD_BUS_REQUEST Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2	bSysBusNeeded			W	0: System bus may sleep 1: System bus access needed

On reception, the radio CPU sets the bus request bit toward the PRCM to 1 if bSysBusNeeded is nonzero, or to 0 if bSysBusNeeded is zero. If bSysBusNeeded is nonzero, this indicates that the system bus stays awake even if the system goes to deep sleep, which must be done for the RF core to run and access the system side for one of the following reasons:

- Any command structure, data structure, and so on, pointed to by a pointer sent to the RF core is placed in system RAM or Flash.
- The RF core must read the temperature because the TX power has a nonzero temperature coefficient.
- The RF core must read the RTC to synchronize with the RAT during CMD_SYNC_STOP_RAT or CMD_SYNC_START_RAT.

CMD_BUS_REQUEST may be sent as a direct command. If so, bSysBusNeeded is given by the parameter in bits 8–15 of CMDR.

The radio CPU returns Done in RFC_DBELL:CMDSTA when the update finishes.

26.3.4 Immediate Commands for Data Queue Manipulation

The following commands are immediate commands used for data queue manipulation for all radio operations that use data queues.

26.3.4.1 CMD_ADD_DATA_ENTRY: Add Data Entry to Queue

Command ID number: 0x0005

CMD_ADD_DATA_ENTRY is an immediate command that takes the parameters listed in Table 26-52.

Table 26-52. CMD_ADD_DATA_ENTRY Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	pQueue			W	Pointer to the queue structure to which the entry is added
8–11	pEntry			W	Pointer to the entry

On reception, the radio CPU appends the provided data entry to the queue indicated. The radio CPU performs the following operations:

```
Set pQueue-> pLastEntry-> pNextEntry = pEntry
Set pQueue-> pLastEntry = pEntry
```

If either of the pointers pQueue or pEntry are invalid (that is, in an address range that is not memory or without 32-bit word alignment), the command fails, and the radio CPU sets the result byte of CMDSTA to ParError. If the queue specified in pQueue is set up not to allow entries to be appended (see Section 26.3.2.7.1), the command fails, and the radio CPU sets the result byte of CMDSTA to QueueError.

26.3.4.2 CMD_REMOVE_DATA_ENTRY: Remove First Data Entry from Queue

Command ID number: 0x0006

CMD_REMOVE_DATA_ENTRY is an immediate command that takes the parameters listed in Table 26-53.

Table 26-53. CMD_REMOVE_DATA_ENTRY Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	pQueue			W	Pointer to the queue structure from which the entry is removed
8–11	pEntry			R	Pointer to the entry that was removed

On reception, the radio CPU removes the first data entry from the queue indicated. The command returns a pointer to the entry that was removed. The radio CPU performs the following operations:

```
Set pEntry = pQueue->pCurrEntry
Set pQueue->pCurrEntry = pEntry->pNextEntry
Set pEntry->status = Finished
```

If the pointer pQueue is invalid, the command fails, and the radio CPU sets the result byte of CMDSTA to ParError. If the queue specified in pQueue is empty, the command fails, and the radio CPU sets the result byte of CMDSTA to QueueError. If the entry to be removed is in the BUSY state, the command fails, and the radio CPU sets the result byte of CMDSTA to QueueBusy.

26.3.4.3 CMD_FLUSH_QUEUE: Flush Queue

Command ID number: 0x0007

CMD_FLUSH_QUEUE is an immediate command that takes the parameters listed in Table 26-54.

Table 26-54. CMD_FLUSH_QUEUE Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	pQueue			W	Pointer to the queue structure to be flushed
8–11	pFirstEntry			R	Pointer to the first entry that was removed

On reception, the radio CPU flushes the queue indicated, and returns a pointer to the first entry that was removed. The radio CPU performs the following operations:

```
set pFirstEntry = pQueue->pCurrEntry
set pQueue->pCurrEntry = NULL
set pQueue->pLastEntry = NULL
```

If the pointer pQueue is invalid, the command fails, and the radio CPU sets the result byte of CMDSTA to ParError. If the first entry to be removed is in the BUSY state, the command fails, and the radio CPU sets the result byte of CMDSTA to QueueBusy. If the queue specified in pQueue is empty, the radio CPU does not need to do any operation, but this is still viewed as a success. The returned pFirstEntry is NULL in this case.

26.3.4.4 CMD_CLEAR_RX: Clear All RX Queue Entries

Command ID number: 0x0008

CMD_CLEAR_RX is an immediate command that takes the parameters listed in Table 26-55.

Table 26-55. CMD_CLEAR_RX Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–:3					Reserved
4–7	pQueue			W	Pointer to the queue structure be cleared

On reception, the radio CPU makes all RX entries indicate that they are empty. The radio CPU performs the following operations:

```
set pTemp = pQueue->pCurrEntry
do
    set pTemp->status = Pending
    if pTemp->type == 1 then
        Set pTemp->nextIndex = 0
        Set pTemp->numElements = 0
        Set pTemp = pTemp->nextIndex
    end if
while (pTemp != NULL and pTemp != pQueue->pCurrEntry)
```

If the pointer pQueue is invalid, the command fails, and the radio CPU sets the result byte of CMDSTA to ParError. If the queue specified in pQueue is empty, the command fails, and the radio CPU sets the result byte of CMDSTA to QueueError. If the first entry to be removed is in the BUSY state, the command fails, and the radio CPU sets the result byte of CMDSTA to QueueBusy.

26.3.4.5 CMD_REMOVE_PENDING_ENTRIES: Remove Pending Entries from Queue

Command ID number: 0x0009

CMD_REMOVE_PENDING_ENTRIES is an immediate command that takes the parameters listed in Table 26-56.

Table 26-56. CMD_REMOVE_PENDING_ENTRIES Command Format

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3					Reserved
4–7	pQueue			W	Pointer to the queue structure to be flushed
8–11	pFirstEntry			R	Pointer to the first entry that was removed

On reception, the radio CPU removes all entries that are in the Pending state from the queue indicated, and returns a pointer to the first entry that was removed. The radio CPU performs the following operations:

```
if pQueue->pCurrEntry->status = Pending, then
    set pFirstEntry = pQueue->pCurrEntry
    set pQueue->pCurrEntry = NULL
    set pQueue->pLastEntry = NULL
else
    set pFirstEntry = pQueue->pCurrEntry->pNextEntry
    set pQueue->pCurrEntry->pNextEntry = NULL
    set pQueue->pLastEntry = pQueue->pCurrEntry
end if
```

If the pointer pQueue is invalid, the command fails, and the radio CPU sets the result byte of CMDSTA to ParError. If the queue specified in pQueue is empty, the radio CPU does not need to do any operation, but this is still viewed as a success. The returned pFirstEntry is NULL in this case.

26.4 Data Queue Usage

This section describes how the radio CPU uses data queues.

26.4.1 Operations on Data Queues Available Only for Internal Radio CPU Operations

Section 26.3.4 lists commands used for data queue manipulation. For internal radio CPU operations described, additional operations are available. These operations are described in Section 26.4.1.1 through Section 26.4.1.5.

26.4.1.1 PROC_ALLOCATE_TX: Allocate TX Entry for Reading

The procedure takes the following input parameter:

- Pointer to queue, pQueue

The procedure returns the following:

- Pointer to allocated data entry, pEntry

The procedure returns with error if the specified queue is empty, or if the first entry of the queue is already busy. Otherwise, the following is done:

```
set pQueue->pCurrEntry->status = Busy
set pEntry = pQueue->pCurrEntry
```

26.4.1.2 PROC_FREE_DATA_ENTRY: Free Allocated Data Entry

The procedure takes the following input parameter:

- Pointer to queue, pQueue

The procedure returns the following:

- Pointer to allocated data entry, pEntry

The procedure returns with error if the specified queue is empty. Otherwise, the following is done:

```
set pQueue->pCurrEntry->status = Active
```

26.4.1.3 PROC_FINISH_DATA_ENTRY: Finish Use of First Data Entry From Queue

The procedure takes the following input parameter:

- Pointer to queue, pQueue

The procedure returns the following:

- Pointer to new entry, pEntry

The procedure returns with error if the specified queue is empty. Otherwise, the following is done:

```
set pTemp = pQueue->pCurrEntry
set pQueue->pCurrEntry = pTemp->pNextEntry
set pTemp->status = Finished
set pEntry = pQueue->pCurrEntry
```

26.4.1.4 PROC_ALLOCATE_RX: Allocate RX Buffer for Storing Data

The procedure takes the following input parameters:

- Pointer to queue, pQueue
- Size of entry element to store, size

The procedure returns the following:

- Pointer to data entry where data is stored, pEntry
- Pointer to a finished data entry, or NULL if not finished, pFinishedEntry

The procedure returns with error if the first entry of the queue is already busy. If there is not room for an entry element of the specified size, including if the queue is empty, a *no space* error is returned. The following procedure describes the operations:

```
set pFinishedEntry == NULL
if pQueue->pCurrEntry == NULL then
    return with no space error
end if
if pQueue->pCurrEntry->type != 1 then
    if pQueue->pCurrEntry->length < size then
        return with no space error
    else
        set pQueue->pCurrEntry->status = Busy
        set pEntry = pQueue->pCurrEntry
    end if
else
    set pTemp = pQueue->pCurrEntry
    if pTemp->nextIndex + 2 + size > pTemp->length then
        set pQueue->pCurrEntry = pTemp->pNextEntry
        set pTemp->status = Finished
        set pFinishedEntry = pTemp
        set pTemp = pTemp->pNextEntry
        if pTemp == NULL or pTemp->length < size + 2 then
            return with no space error
        end if
    end if
    set pTemp->status = Busy
    set pEntry = pTemp
end if
```

26.4.1.5 PROC_FINISH_RX: Commit Received Data to RX Data Entry

The procedure takes the following input parameters:

- Pointer to queue, pQueue
- Size of entry element that is stored, size

The procedure returns the following:

- Pointer to data entry where data is stored, pEntry
- Pointer to a finished data entry, or NULL if not finished, pFinishedEntry

The procedure returns with error if the queue is empty or if there is not room for an entry element of the specified size. Otherwise, the following is done:

```
if pQueue->pCurrEntry->type != 1 then
    set pTemp = pQueue->pCurrEntry
    set pQueue->pCurrEntry = pTemp->pNextEntry
    set pTemp->status = Finished
else
    increase pQueue->pCurrEntry->nextIndex by size
    increment pQueue->pCurrEntry->numElements by 1
    if pQueue->pCurrEntry->nextIndex + 2 == pQueue->pCurrEntry->length then
        set pTemp = pQueue->pCurrEntry
        set pQueue->pCurrEntry = pTemp->pNextEntry
        set pTemp->status = Finished
        set pFinishedEntry == pTemp
    else
        set pQueue->pCurrEntry->status = Active
        set pFinishedEntry == NULL
    end if
end if
```

This operation is done after doing PROC_ALLOCATE_RX and writing to the correct locations in the buffer; the size must be the same as with PROC_ALLOCATE_RX.

26.4.2 Radio CPU Usage Model

26.4.2.1 Receive Queues

When the radio CPU receives a packet, it prepares a buffer for reading by calling PROC_ALLOCATE_RX. If this is successful, the allocated buffer is used for storing the incoming packet as defined for each protocol. If a no space error occurs, the received data cannot be stored, and the handling is defined for each protocol.

After a packet is received, it may be kept or discarded depending on rules defined for each protocol. To keep the packet, the radio CPU calls PROC_FINISH_RX. This makes the received data available for the system CPU. To discard the packet, the radio CPU calls PROC_FREE_DATA_ENTRY, meaning that the next packet may overwrite the data received in the last packet.

26.4.2.2 Transmit Queues

When the radio CPU is about to transmit a packet from a TX queue, it calls PROC_ALLOCATE_TX to get a pointer to the data to transmit. When the packet transmits, the radio CPU calls PROC_FINISH_DATA_ENTRY or PROC_FREE_DATA_ENTRY. If PROC_FINISH_DATA_ENTRY is called, the system CPU is informed that the entry is finished and may be reused. This calling process must be used if retransmission of the packet is not an option. If PROC_FREE_DATA_ENTRY is called, the transmitted entry remains first in the queue so that it may be transmitted, which is used when an acknowledgment is expected.

If an acknowledgment is received on a packet that was transmitted, followed by the radio CPU calling PROC_FREE_DATA_ENTRY, the radio CPU calls PROC_ALLOCATE_TX followed by PROC_FINISH_DATA_ENTRY (this is equivalent to CMD_REMOVE_DATA_ENTRY, see Section 26.3.3.2). This calling process causes the next entry in the queue to be transmitted. If an acknowledgment is not received, the last transmitted packet is retransmitted.

26.5 IEEE 802.15.4

This section describes IEEE 802.15.4-specific command structure, interrupts, data handling, radio operation commands, and immediate commands.

26.5.1 IEEE 802.15.4 Commands

Table 26-57 and Table 26-58 define the IEEE 802.15.4-specific radio operation commands.

Table 26-57. IEEE 802.15.4 Radio Operation Commands on Background Level

ID	Command Name	Description
0x2801	CMD_IEEE_RX	Run receiver
0x2802	CMD_IEEE_ED_SCAN	Run energy detect scan

Table 26-58. IEEE 802.15.4 Radio Operation Commands on Foreground Level

ID	Command Name	Description
0x2C01	CMD_IEEE_TX	Transmit packet
0x2C02	CMD_IEEE_CSMA	Perform CSMA-CA
0x2C03	CMD_IEEE_RX_ACK	Receive acknowledgment
0x2C04	CMD_IEEE_ABORT_BG	ABORT background level operation

Table 26-59 defines immediate commands.

Table 26-59. IEEE 802.15.4 Immediate Commands

ID	Command Name	Description
0x2001	CMD_IEEE_MOD_CCA	Modify CCA parameters for running receiver
0x2002	CMD_IEEE_MOD_FILT	Modify frame filtering parameters for running receiver
0x2003	CMD_IEEE_MOD_SRC_MATCH	Modify source matching parameters for running receiver
0x2401	CMD_IEEE_ABORT_FG	ABORT foreground level operation
0x2402	CMD_IEEE_STOP_FG	Stop foreground level operation
0x2403	CMD_IEEE_CCA_REQ	Request CCA and RSSI information

26.5.1.1 IEEE 802.15.4 Radio Operation Command Structures

Table 26-9 defines the first 14 bytes for all radio operation commands. The CMD_IEEE_ABORT_BG command does not have any additional fields to those 14 bytes.

Table 26-60. IEEE 802.15.4 RX Command Structure

Byte Index	Field Name	Type	Description
14	channel	W	Channel to tune to in the start of the operation: 0: Use existing channel 11–26: Use as IEEE 802.15.4 channel, that is frequency is [2405 + 5 × (channel − 11)] MHz 60–207: Frequency is (2300 + channel) MHz Others: reserved
15	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-70 for details)
16–19	pRxQ	W	Receive queue
20–23	pOutput	W	Pointer to result structure (see Table 26-69) (NULL: Do not store results)
24–25	frameFiltOpt	R/W	Frame filtering options (see Table 26-72 for details)
26	frameTypes	R/W	Frame types to receive in frame filtering (see Table 26-73 for details)
27	ccaOpt	R/W	CCA options (see Table 26-71 for details)
28	ccaRssiThr	R/W	RSSI threshold for CCA
29			Reserved
30	numExtEntries	W	Number of extended address entries
31	numShortEntries	W	Number of short address entries
32–35	pExtEntryList	W	Pointer to list of extended address entries
36–39	pShortEntryList	W	Pointer to list of short address entries
40–47	localExtAddr	W	The extended address of the local device
48–49	localShortAddr	W	The short address of the local device
50–51	localPanID	W	The PAN ID of the local device
52–54			Reserved
55	endTrigger	W	Trigger that causes the device to end the RX operation
56–59	endTime	W	Time parameter for endTrigger

Table 26-61. IEEE 802.15.4 Energy Detect Scan Command Structure

Byte Index	Field Name	Type	Description
14	channel	W	Channel to tune to at the start of the operation: 0: Use existing channel 11–26: Use as IEEE 802.15.4 channel, that is frequency is [2405 + 5 × (channel – 11)] MHz 60–207: Frequency is (2300 + channel) MHz Others: reserved
15	ccaOpt	R/W	CCA options (see Table 26-71 for details)
16	ccaRssiThr	R/W	RSSI threshold for CCA
17			Reserved
18	maxRssi	R	The maximum RSSI recorded during the ED scan
19	endTrigger	W	Trigger that causes the device to end the RX operation
20–23	endTime	W	Time parameter for endTrigger

Table 26-62. IEEE 802.15.4 CSMA-CA Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	randomState			R/W	The state of the pseudo-random generator
16	macMaxBE			W	The IEEE 802.15.4 MAC parameter macMaxBE
17	macMaxCSMABackoffs			W	The IEEE 802.15.4 MAC parameter macMaxCSMABackoffs
18	csmaConfig	0–4	initCW	W	The initialization value for the CW parameter
		5	bSlotted	W	0 for nonslotted CSMA, 1 for slotted CSMA
		6–7	rxOffMode	W	0: RX stays on during CSMA backoffs 1: The CSMA-CA algorithm suspends the receiver if no frame is being received 2: The CSMA-CA algorithm suspends the receiver if no frame is being received, or after finishing it (including auto ACK) otherwise 3: The CSMA-CA algorithm suspends the receiver immediately during back-offs
19	NB			R/W	The NB parameter from the IEEE 802.15.4 CSMA-CA algorithm
20	BE			R/W	The BE parameter from the IEEE 802.15.4 CSMA-CA algorithm
21	remainingPeriods			R/W	The number of remaining periods from a paused backoff countdown
22	lastRssi			R	RSSI measured at the last CCA operation
23	endTrigger			W	Trigger that causes the device to end the CSMA-CA operation
24–27	lastTimeStamp			R	Time of the last CCA operation
28–31	endTime			W	Time parameter for endTrigger

Table 26-63. IEEE 802.15.4 TX Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	txOpt	0	bIncludePhyHdr	W	0: Find PHY header automatically. 1: Insert PHY header from the buffer.
		1	bIncludeCrc	W	0: Append automatically calculated CRC. 1: Insert FCS (CRC) from the buffer.
		2			Reserved
		3–7	payloadLenMsb	W	Most significant bits of payload length. Must only be nonzero to create long nonstandard packets for test purposes
15	payloadLen			W	Number of bytes in the payload
16–19	pPayload			W	Pointer to payload buffer of size payloadLen
20–23	timeStamp			R	Timestamp of transmitted frame

Table 26-64. IEEE 802.15.4 Receive ACK Command Structure

Byte Index	Field Name	Type	Description
14	seqNo	W	Sequence number to expect
15	endTrigger	W	Trigger that causes the device to give up acknowledgment reception
16–19	endTime	W	Time parameter for endTrigger

26.5.1.2 IEEE 802.15.4 Immediate Command Structures

Table 26-65. IEEE 802.15.4 Modify CCA Immediate Command Structure

Byte Index	Field Name	Type	Description
0–1	commandNo	W	The command number
2	newCcaOpt	W	New value of ccaOpt for the running background level operation (see Table 26-71 for details)
3	newCcaRssiThr	W	New value of ccaRssiThr for the running background level operation

Table 26-66. IEEE 802.15.4 Modify Frame Filtering Immediate Command Structure

Byte Index	Field Name	Type	Description
0–1	commandNo	W	The command number
2–3	newFrameFiltOpt	W	New value of frameFiltOpt for the running background level operation
4	newFrameTypes	W	New value of frameTypes for the running background level operation

Table 26-67. IEEE 802.15.4 Enable or Disable Source Matching Entry Immediate Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command number
2	options	0	bEnable	W	0: Disable entry 1: Enable entry
		1	srcPend	W	New value of the pending bit for the entry
		2	entryType	W	0: Extended address 1: Short address
		3–7			Reserved
3	entryNo			W	Index of entry to enable or disable

Table 26-68. IEEE 802.15.4 Request CCA State Immediate Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command number
2	currentRssi			R	The RSSI currently observed on the channel
3	maxRssi			R	The maximum RSSI observed on the channel because RX was started
4	ccaInfo	0–1	ccaState	R	Value of the current CCA state: 00: Idle 01: Busy 10: Invalid
		2–3	ccaEnergy	R	Value of the current energy detect CCA state: 00: Idle 01: Busy 10: Invalid
		4–5	ccaCorr	R	Value of the current correlator based carrier sense CCA state: 00: Idle 01: Busy 10: Invalid
		6	ccaSync	R	Value of the current sync found based carrier sense CCA state: 0: Idle 1: Busy
		7			Reserved

26.5.1.3 Output Structures

Table 26-69. RX Command

Byte Index	Field Name	Type	Description
0	nTxAck	R/W	Total number of transmitted ACK frames
1	nRxBeacon	R/W	Number of received beacon frames
2	nRxData	R/W	Number of received data frames
3	nRxAck	R/W	Number of received acknowledgment frames
4	nRxMacCmd	R/W	Number of received MAC command frames
5	nRxReserved	R/W	Number of received frames with reserved frame type
6	nRxOk	R/W	Number of received frames with CRC error
7	nRxIgnored	R/W	Number of frames received that are to be ignored
8	nRxBufFull	R/W	Number of received frames discarded because the RX buffer was full
9	lastRssi	R	RSSI of last received frame
10	maxRssi	R	Highest RSSI observed in the operation
11			Reserved
12–15	beaconTimeStamp	R	Timestamp of last received beacon frame

26.5.1.4 Other Structures and Bit Fields

Table 26-70. Receive Queue Entry Configuration Bit Field

Bits	Bit Field Name	Description
0	bAutoFlushCrc	If 1, automatically remove packets with CRC error from RX queue.
1	bAutoFlushIgn	If 1, automatically remove packets that can be ignored according to frame filtering from RX queue.
2	bIncludePhyHdr	If 1, include the received PHY header field in the stored packet; otherwise discard it.
3	bIncludeCrc	If 1, include the received CRC field in the stored packet; otherwise discard it. This requires pktConf.bUseCrc to be 1.
4	bAppendRssi	If 1, append an RSSI byte to the packet in the RX queue.
5	bAppendCorrCrc	If 1, append a correlation value and CRC result byte to the packet in the RX queue.
6	bAppendSrcInd	If 1, append an index from the source matching algorithm.
7	bAppendTimestamp	If 1, append a timestamp to the packet in the RX queue.

Table 26-71. CCA Configuration Bit Field

Bits	Bit Field Name	Description
0	ccaEnEnergy	Enable energy scan as CCA source.
1	ccaEnCorr	Enable correlator-based carrier sense as CCA source.
2	ccaEnSync	Enable sync found based carrier sense as CCA source.
3	ccaCorrOp	Operator to use between energy based and correlator-based CCA 0: Report busy channel if either ccaEnergy or ccaCorr are busy. 1: Report busy channel if both ccaEnergy and ccaCorr are busy.
4	ccaSyncOp	Operator to use between sync found based CCA and the others 0: Always report busy channel if ccaSync is busy. 1: Always report idle channel if ccaSync is idle.
5–6	ccaCorrThr	Threshold for number of correlation peaks in correlator-based carrier sense
7		Reserved

Table 26-72. Frame Filtering Configuration Bit Field

Bits	Bit Field Name	Description
0	frameFiltEn	0: Disable frame filtering. 1: Enable frame filtering.
1	frameFiltStop	0: Receive all packets to the end. 1: Stop receiving frame once frame filtering has caused the frame to be rejected.
2	autoAckEn	0: Disable auto ACK. 1: Enable auto ACK.
3	slottedAckEn	0: Non-slotted ACK. 1: Slotted ACK.
4	autoPendEn	0: Auto-pend disabled. 1: Auto-pend enabled.
5	defaultPend	The value of the pending data bit in auto ACK packets that are not subject to auto-pend.
6	bPendDataReqOnly	0: Use auto-pend for any packet. 1: Use auto-pend for data request packets only.
7	bPanCoord	0: Device is not the PAN coordinator. 1: Device is the PAN coordinator.
8–9	maxFrameVersion	Reject frames where the frame version field in the FCF is greater than this value.
10–12	fcfReservedMask	Value to be ANDed with the reserved part of the FCF; frame rejected if result is nonzero.
13–14	modifyFtFilter	Treatment of MSB of frame type field before frame-type filtering: 0: No modification. 1: Invert MSB. 2: Set MSB to 0. 3: Set MSB to 1.
15	bStrictLenFilter	0: Accept acknowledgment frames of any length ≥ 5. 1: Accept only acknowledgment frames of length 5.

Table 26-73. Frame Type Filtering Bit Field

Bits	Bit Field Name	Description
0	bAcceptFt0Beacon	Treatment of frames with frame type 000 (beacon): 0: Reject 1: Accept
1	bAcceptFt1Data	Treatment of frames with frame type 001 (data): 0: Reject 1: Accept
2	bAcceptFt2Ack	Treatment of frames with frame type 010 (ACK): 0: Reject, unless running ACK receive command 1: Always accept
3	bAcceptFt3MacCmd	Treatment of frames with frame type 011 (MAC command): 0: Reject 1: Accept
4	bAcceptFt4Reserved	Treatment of frames with frame type 100 (reserved): 0: Reject 1: Accept
5	bAcceptFt5Reserved	Treatment of frames with frame type 101 (reserved): 0: Reject 1: Accept
6	bAcceptFt6Reserved	Treatment of frames with frame type 110 (reserved): 0: Reject 1: Accept

Table 26-73. Frame Type Filtering Bit Field (continued)

Bits	Bit Field Name	Description
7	bAcceptFt7Reserved	Treatment of frames with frame type 111 (reserved): 0: Reject 1: Accept

Table 26-74. Short Address Entry Structure

Byte Index	Field Name	Description
0–1	shortAddr	Short address of the entry
2–3	panID	PAN ID of the entry

Table 26-75. Extended Address List Structure

Byte Index	Field Name	Type	Description
0–4K−1	srcMatchEn	R/W	Words with enable bits for each extAddrEntry; LSB of first word corresponds to entry 0. The array size K = ceil (N / 32), where N is the number of entries (given by numExtEntries, see Table 26-60) and ceil denotes rounding upward.
4K–8K−1	srcPendEn	R/W	Words with pending data bits for each extAddrEntry; LSB of first word corresponds to entry 0.
8K–8K+7	extAddrEntry[0]	W	Extended address number 0
...			
8K+8n–8K+8n+7	extAddrEntry[n]	W	Extended address number n
...			
8K+8(N−1)–8K+8N+7	extAddrEntry[N-1]	W	Extended address number N−1 (last entry)

Table 26-76. Short Address List Structure

Byte Index	Field Name	Type	Description
0:4K−1	srcMatchEn	R/W	Words with enable bits for each shortAddrEntry; LSB of first word corresponds to entry 0. The array size K = ceil (N / 32), where N is the number of entries (given by numShortEntries, see Table 26-60) and ceil denotes rounding upward.
4K:8K−1	srcPendEn	R/W	Words with pending data bits for each shortAddrEntry; LSB of first word corresponds to entry 0.
8K–8K+3	shortAddrEntry[0]	W	Short address number 0; the entry is an address/PAN ID pair as defined in Table 26-74.
...			
8K+4n–8K+4n+3	shortAddrEntry[n]	W	Short address number n; the entry is an address/PAN ID pair as defined in Table 26-74.
...			
8K+4(N−1)–8K+4N+3	shortAddrEntry[N-1]	W	Short address number N−1 (last entry); the entry is an address/PAN ID pair as defined in Table 26-74.

Table 26-77. Receive Correlation/CRC Result Bit Field

Bits	Bit Field Name	Description
0–5	corr	The correlation value
6	bIgnore	1 if the packet must be rejected by frame filtering; 0 otherwise.
7	bCrcErr	1 if the packet was received with CRC error; 0 otherwise.

26.5.2 Interrupts

The interrupts to be used by the IEEE 802.15.4 commands are listed in Table 26-78. Each interrupt may be enabled individually in the system CPU. Details for when the interrupts are generated are given in Section 26.5.4.

Table 26-78. Interrupt Definitions Applicable to IEEE 802.15.4

Interrupt Number	Interrupt Name	Description
0	COMMAND_DONE	A background level radio operation command has finished.
1	LAST_COMMAND_DONE	The last background level radio operation command in a chain of commands has finished.
2	FG_COMMAND_DONE	A foreground radio operation command has finished.
3	LAST_FG_COMMAND_DONE	The last foreground radio operation command in a chain of commands has finished.
4	TX_DONE	Transmitted frame
5	TX_ACK	Transmitted automatic ACK frame
16	RX_OK	Frame received with CRC OK
17	RX_NOK	Frame received with CRC error
18	RX_IGNORED	Frame received with ignore flag set
22	RX_BUF_FULL	Frame received that did not fit in the TX queue
23	RX_ENTRY_DONE	TX queue data entry changing state to Finished
29	MODULES_UNLOCKED	As part of the boot process, the Arm Cortex-M0 processor has opened access to RF core modules and memories.
30	BOOT_DONE	The RF core CPU boot is finished.
31	INTERNAL_ERROR	The radio CPU has observed an unexpected error.

26.5.3 Data Handling

For all the IEEE 802.15.4 commands, data received over the air is stored in a receive queue.

Data to be transmitted is fetched from a buffer given in the transmit command.

26.5.3.1 Receive Buffers

A frame being received is stored in the receive buffer. First, a length byte or word is stored, if configured in the RX entry, by config.lenSz, and calculated from the length received over the air and the configuration of appended status information.

The format of the entry elements in the receive queue pointed to by pRxQ is given by the configuration rxConfig defined in Section 26.6.1.4.

Following the length field, the received PHY header byte is stored if rxConfig.bIncludePhyHdr is 1. If a length field is present, this byte is redundant except for the reserved bit. The received MAC header and MAC payload is stored as received over the air. The MAC footer containing the 16-bit frame check sequence is stored if rxConfig.bIncludeCrc is 1.

If rxConfig.bAppendRssi is 1, a byte indicating the received RSSI value is appended. If rxConfig.bAppendCorrCrc is 1, a status byte of the type defined in Table 26-77, is appended. If rxConfig.bAppendSrcInd is 1, a byte giving the index of the first source matching entry that matches the header of the received packet is appended, or 0xFF if no match. If rxConfig.bAppendTimeStamp is 1, a timestamp indicating the start of the frame is appended. This timestamp is a four-byte number from the radio timer. Though the timestamp is multibyte, no word-address alignment is made, so the timestamp must be written and read bytewise. The timestamp is captured when SFD is found, but adjusted to reflect the start of the frame (assuming 4 preamble bytes as per the standard), defined so that it corresponds to the time of the start trigger used on

the transmit side. The adjustment is defined in the syncTimeAdjust firmware-defined parameter, and may be overridden.

Figure 26-6 shows the format of an entry element in the RX queues.

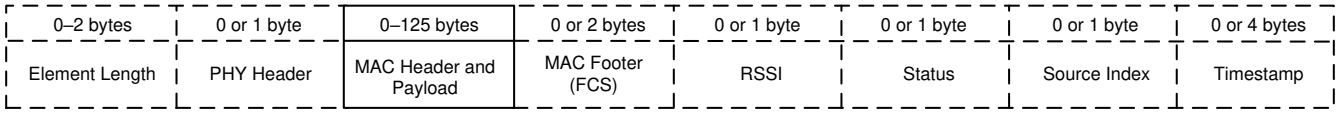


Figure 26-6. RX Queue Entry Element (Stapled Fields are Optional)

26.5.3.2 Transmit Buffers

In the transmit operation, a pointer to a buffer containing the payload is given by pPayload. The length of this buffer is given separately by payloadLen. The contents of the transmit buffer is given by the txOpt parameter. The transmit buffer always contains the MAC header and MAC payload. If txOpt.bIncludePhyHdr is 1, the buffer also includes the byte to be transmitted as a PHY header as the first byte in the buffer. If txOpt.bIncludeCrc is 1, the two last bytes of the buffer are transmitted as a CRC instead of the CRC being calculated automatically.

26.5.4 Radio Operation Commands

Before running any radio operation command described in this document, the radio must be set up in IEEE 802.15.4 mode using the command CMD_RADIO_SETUP. Otherwise, the operation ends with an error.

In IEEE 802.15.4 mode, the radio CPU accepts two levels of radio operation commands. Operations can run in the background level or in the foreground level. Each operation can only run in one of these levels. Operations in the foreground level normally require a background-level operation running at the same time.

The background-level operations are the receive and energy detect scan operations. Only one of these can run at a time. The foreground-level operations are the CSMA-CA operation, the receive ACK operation, the transmit operation, the abort background level operation, and the modify radio setup operation. These can be entered as one command or a command chain, even if a background-level operation is running. The CSMA-CA and receive ACK operations run simultaneously with the background-level operation. The transmit operation causes the background level operation to be suspended until the transmission is done. Table 26-79 shows the allowed combinations of background and foreground-level operations. Violation of these combinations causes an error when the foreground-level command is about to start, signaled by the ERROR_WRONG_BG status in the status field of the foreground-level command structure.

Table 26-79. Allowed Combinations of Foreground and Background Level Operations

Foreground Level Operation	Background Level Operation		
	None	CMD_IEEE_RX	CMD_IEEE_ED_SCAN
None	Allowed	Allowed	Allowed
CMD_IEEE_TX	Allowed1	Allowed	Allowed
CMD_IEEE_CSMA	Forbidden	Allowed	Allowed
CMD_IEEE_RX_ACK	Forbidden	Allowed	Forbidden
CMD_IEEE_ABORT_BG	Allowed2	Allowed	Allowed

A non-15.4 radio operation may not be run simultaneously with a 15.4 radio operation; if a non-15.4 radio operation is entered while a 15.4 operation is running on either level, scheduling error occurs. Chains of 15.4 and non-15.4 operations can be created, however.

When a foreground-level operation finishes, an FG_COMMAND_DONE interrupt is raised. If the command was the last one in a chain, a LAST_FG_COMMAND_DONE interrupt is raised as well (refer to Table 26-78). Background-level operations use the common interrupts, COMMAND_DONE and LAST_COMMAND_DONE (see Table 26-78).

The status field of the command structure is updated during the operation. When submitting the command, the system CPU writes this field with a state of IDLE. During the operation, the radio CPU updates the field to indicate the operation mode. When the operation is done, the radio CPU writes a status indicating that the command has finished. Table 26-80 lists the status codes for IEEE 802.15.4 radio operation.

Table 26-80. IEEE 802.15.4 Radio Operation Status Codes

Number	Name	Description
Operation Not Finished		
0x0000	IDLE	Operation not started
0x0001	PENDING	Waiting for start trigger
0x0002	ACTIVE	Running operation
0x2001	IEEE_SUSPENDED	Operation suspended
Normal Operation Ending		
0x2400	IEEE_DONE_OK	Operation ended normally
0x2401	IEEE_DONE_BUSY	CSMA-CA operation ended with failure
0x2402	IEEE_DONE_STOPPED	Operation stopped after stop command
0x2403	IEEE_DONE_ACK	ACK packet received with pending data bit cleared
0x2404	IEEE_DONE_ACKPEND	ACK packet received with pending data bit set
0x2405	IEEE_DONE_TIMEOUT	Operation ended due to timeout
0x2406	IEEE_DONE_BGEND	FG operation ended because necessary background-level operation ended
0x2407	IEEE_DONE_ABORT	Operation aborted by command
Operation Ending with Error		
0x0806	ERROR_WRONG_BG	Foreground level operation is not compatible with running background-level operation
0x2800	IEEE_ERROR_PAR	Illegal parameter
0x2801	IEEE_ERROR_NO_SETUP	Radio was not set up in IEEE 802.15.4 mode
0x2802	IEEE_ERROR_NO_FS	Synthesizer was not programmed when running RX or TX
0x2803	IEEE_ERROR_SYNTH_PROG	Synthesizer programming failed
0x2804	IEEE_ERROR_RXOVF	RX overflow observed during operation
0x2805	IEEE_ERROR_TXUNF	TX underflow observed during operation

The conditions for giving each status are listed for each operation. Some of the error causes listed in Table 26-80 are not repeated in these lists. In some cases, general error causes described in Section 26.3 may occur. In all of these cases, the result of the operation as defined in Section 26.3 is ABORT.

26.5.4.1 RX Operation

The receive radio operation is a background-level operation, started with the CMD_IEEE_RX command and using the command structure given in Table 26-60.

At the start of an RX operation, the radio CPU waits for the start trigger, then programs the frequency based on the channel parameter. If channel is 0xFF, the operation keeps running on an already configured channel. This requires that the operation follows another receive operation or a synthesizer programming operation. If the frequency synthesizer is not running, the operation ends with an error. After programming the frequency, the radio CPU configures the receiver to receive IEEE 802.15.4 packets.

When the demodulator obtains sync on a frame, the PHY header is read first. The 7 LSBs of this byte give the frame length. The further treatment depends on the setting of frameFiltOpt. If frameFiltOpt.frameFiltEn is 1, further frame filtering is done as explained below. If it is 0, no frame filtering is done.

The number of bytes given by the received PHY header are received and stored in the receive queue given by pRxQ. As explained in Section 26.7.1, the format depends on rxConfig. The last two bytes of the PHY payload are the FCS, or CRC, for the packet. These bytes are checked according to the FCS specification, and the further treatment depends on the CRC result.

If there is a CRC error and rxConfig.bAutoFlushCrc is 1, the packet is discarded from the RX buffer. If there is no available RX buffer with enough available space to hold the received packet, the received data is discarded. If frameFiltOpt.frameFiltStop is 1, the reception stops, otherwise the packet is received so that the CRC can be checked.

26.5.4.1.1 Frame Filtering and Source Matching

If frameFiltOpt.frameFiltEn is 1, frame filtering and source matching are performed as described in this section. The frame filtering can have several purposes:

- Distinction between different packet types
- Rejection of packets with a nonmatching destination address
- Rejection of packets with unknown version or illegal fields
- Automatic identification of source address
- Automatic acknowledgment transmission
- Automatic insertion of pending data bit based on source address

26.5.4.1.1.1 Frame Filtering

When frame filtering is enabled, the MAC header of the packet is investigated by the radio CPU. The frame control field (FCF) is checked first. The frame type subfield is the first subfield of the FCF to be checked, and determines the further processing. The MSB of the frame type is processed according to frameFiltOpt.modifyFtFilter before the check is made. The result of this modification is only used when checking, not when storing the FCF in the RX queue entry. For each of the eight possible values of the frame type field (including four reserved values), the frame can be setup to be accepted or rejected. This is controlled by the bits of frameTypes. If the frame type is Acknowledgment (010b) and a CMD_RX_ACK operation is running in the foreground, the packet is processed further even if frameTypes.bAcceptFt2Ack is 0. Section 26.5.4.5 gives more details on the processing in that case.

Filtering is performed on the Frame Version and Reserved subfields. If the frame version is greater than frameFiltOpt.maxFrameVersion, the frame is rejected.

If the Reserved subfield ANDed with frameFiltOpt.fcfReservedMask is nonzero, the frame is rejected. The addressing fields are checked to see if the frame must be accepted or not. This filtering follows the rules for third-level filtering (refer to the IEEE 802.15.4 standard). When checking against the local address, the localExtAddr or localShortAddr field is used, and when checking against the local PAN ID, the localPanID field is used.

If frameFiltOpt.bStrictLenFilter is 1 and the frame type indicates that the frame is an acknowledgment frame, the frame is rejected if the length of the PHY payload is not 5, which is the length of a correctly-formulated ACK frame.

If frameFiltOpt.frameFiltStop is 1 and the frame filtering gives the conclusion that the frame is to be rejected, reception stops and the radio returns to sync search. Otherwise, the frame is received to the end.

The radio CPU checks the header to see if an acknowledgment is to be transmitted. This gives a preliminary result; the actual transmission of the ACK depends on the status at the end of the frame. The condition for transmitting an acknowledgment frame is given in Section 26.5.4.1.3.

26.5.4.1.1.2 Source Matching

Source matching is performed on frames accepted by the frame filtering with a source address present. If the source address was an extended address, the received address is compared against the entries in the list pExtEntryList. If the source address was a short address, the received address and source pan ID are compared against the entries in the list pShortEntryList.

The number of entries that the lists can hold is given by numExtEntries and numShortEntries. If either of these values are 0, no source matching is performed on addresses of the corresponding type, and the corresponding pointer is NULL. The lists start with source mapping enable bits, srcMatchEn, and continue with pending enable bits, srcPendEn, followed by the list entries, see Table 26-74 and Table 26-75. The enable bits consist of the number of 32-bit words needed to hold an enable bit for each entry in the list. For each entry where the corresponding srcMatchEn bit is 1, the entry is compared against the received source address for extended addresses, or against the received source address and PAN ID for short addresses. If a match is found, the index is stored, and reported back in the message footer if configured (see Section 26.7.1). If no match is found, the index reported back is 0xFF.

The source matching procedure may also be used for finding the pending data bit to be transmitted in an auto-acknowledgment frame (see Section 26.5.4.1.3). If frameFiltOpt.autoPendEn is 1 and a source match was found, the pending data bit is set to the value of the bit in srcPendEn corresponding to the index of the match. If no match was found or if frameFiltOpt.autoPendEn is 0, the pending data bit is set equal to frameFiltOpt.defaultPend. If frameFiltOpt.bPendDataReqOnly is 1, the radio CPU investigates the frame to determine if it is a MAC command frame with the command frame identifier set to a Data Request. If not, the pending data bit of an auto ACK is set equal to 0, regardless of the source matching result and the value of frameFiltOpt.defaultPend.

26.5.4.1.2 Frame Reception

After the frame filtering is done, the rest of the packet is received and stored in the receive queue. The last two bytes of the PHY packet is the MAC footer, or FCS, which is a checked CRC. The CRC is only stored in the queue if rxConfig.bIncludeCrc is 1.

The status of the received frame depends on the frame filtering result and the CRC check result. Two status bits bCrcErr and bIgnore must be maintained. If configured, these bits are present in the Status byte of the RX queue entry. The bCrcErr bit is 1 if the frame had a CRC error, and 0 otherwise. The bIgnore bit is 1 if frame filtering is enabled and the frame was rejected by frame filtering, and 0 otherwise.

Note

If frameFiltOpt.frameFiltStop is 1, frames with bIgnore equal to 1 are never observed, as the reception is stopped and the received bytes are not stored in the queue. If rxConfig.bAutoFlushCrc is 1, packets with bCrcErr equal to 1 are removed from the queue after reception, and if rxConfig.bAutoFlushIgn is 1, packets with bIgnore equal to 1 are removed from the queue after reception.

After a packet is received, an interrupt is raised and one of the counters in pOutput is incremented. Table 26-81 lists these conditions.

Table 26-81. Conditions for Incrementing Counters and Raising Interrupts for RX Operation

Condition	Counter Incremented	Interrupt Generated
Frame received with CRC OK and frame filtering disabled	nRxData	RX_OK
Frame received with CRC error	nRxNok	RX_NOK
Frame received that did not fit in the RX queue	nRxBufFull	RX_BUF_FULL

Table 26-81. Conditions for Incrementing Counters and Raising Interrupts for RX Operation (continued)

Condition	Counter Incremented	Interrupt Generated
Beacon frame received with CRC OK and bIgnore = 0	nRxBeacon	RX_OK
ACK frame received with CRC OK and bIgnore = 0	nRxAck	RX_OK
Data frame received with CRC OK and bIgnore = 0	nRxData	RX_OK
MAC command frame received with CRC OK and bIgnore = 0	nRxMacCmd	RX_OK
Frame with reserved frame type received with CRC OK and bIgnore = 0	nRxReserved	RX_OK
Frame received with CRC OK and bIgnore = 1	nRxIgnored	RX_IGNORED
The first RX data entry in the RX queue changed state to finished	—	RX_ENTRY_DONE

When a frame is received, the RSSI observed while receiving the frame is written to pOutput->lastRssi. If the frame was a beacon frame accepted by the frame filtering and with CRC OK, the timestamp at the beginning of the frame is written to pOutput ->beaconTimeStamp. If the timestamp is appended to the RX entry element (see Section 26.7.1), these two timestamps are the same for a beacon frame.

After a packet is received, the radio CPU either restarts sync search or sends an acknowledgment frame. The conditions for the latter are as given in Section 26.5.4.1.3.

26.5.4.1.3 ACK Transmission

After a packet is received, the radio CPU initiates transmission of an acknowledgment frame, given that all of the following conditions are satisfied:

- Auto ACK is enabled by frameFiltOpt.autoAckEn = 1
- The frame is accepted by frame filtering (bIgnore = 0)
- The frame is a data frame or a MAC command frame
- The destination address is not the broadcast address
- The ACK request bit of the FCF is set
- The CRC check is passed (bCrcErr = 0)
- The frame fits in the receive queue

The transmit time of the ACK packet is timed by the radio CPU, depending on frameFiltOpt.slottedAckEn. If this bit is 0, the ACK packet is transmitted 192 µs after the end of the received packet. Otherwise, slotted ACK is used. Assume that the received packet started on a backoff-slot boundary. The ACK frame then starts a whole number of backoff periods later than the start of the received frame, at the first backoff boundary following at least one TurnaroundTime-symbol period after the end of the received frame.

The contents of the automatically transmitted ACK frame are as follows:

- The PHY header is 0x05
- The PHY payload consists of a 3-byte MAC header and a 2-byte MAC footer
- The MAC header starts with the 2-byte FCF with the following fields:
  - The Frame Type subfield is 010b
  - The Frame Pending subfield is set as described in Section 26.5.4.1.1.2
  - The remaining subfields are set to all zeros
- The next byte in the MAC header is the sequence number, which is set equal to the sequence number of the received frame
- The MAC footer is the FCS, which is calculated automatically

After the ACK frame is transmitted, a TX_ACK interrupt is raised. The radio CPU then enables the receiver again.

26.5.4.1.4 End of Receive Operation

The receive operation can end as a result of the end trigger given by endTrigger and endTime, or by a command. The commands that can end the receive operation are the immediate commands CMD_ABORT and CMD_STOP, and the foreground-level radio operation command CMD_IEEE_ABORT_BG. The end-trigger and the CMD_STOP command cause the receiver to keep running until the end of the frame, or until the reception would otherwise be stopped if observed while a packet was being received. The CMD_ABORT and CMD_IEEE_ABORT_BG commands cause the receiver to stop as quickly as the implementation allows.

A receive operation ends through one of the causes listed in Table 26-82. The status field of the command structure after the command has ended indicates the reason why the operation ended. In all cases, a COMMAND_DONE interrupt is raised. In each case, the result is indicated as TRUE, FALSE, or ABORT. The receiver decides whether to start the next command (if any) indicated in pNextOp, or to return to an IDLE state depending on the result. Before the receive operation ends, the radio CPU writes the maximum observed RSSI during the receive operation to pOutput->maxRssi.

If a transmit operation is started in the foreground, the receive operation is suspended. The receiver stops as when aborted, but the synthesizer is left on to the extent possible when switching to transmit mode. When the receiver has stopped, the status field of the command structure is set to IEEE_SUSPENDED. When the transmit command is done, the receiver restarts and the status field of the command structure is set back to RUNNING.

Table 26-82. End of Receive Operation

Condition	Status Code	Result
Observed end trigger and finished any ongoing reception	IEEE_DONE_OK	TRUE
Received CMD_STOP	IEEE_DONE_STOPPED	FALSE
Received CMD_ABORT or CMD_IEEE_ABORT_BG	IEEE_DONE_ABORT	ABORT
Observed illegal parameter	IEEE_ERROR_PAR	ABORT

26.5.4.1.5 CCA Monitoring

While the receiver is running, the radio CPU monitors some signals for use in clear-channel assessment. This monitoring is controlled by ccaOpt. There are three sources for CCA: RSSI above level (ccaEnergy), carrier sense based on the correlation value (ccaCorr), and carrier sense based on sync found (ccaSync). Each of these may have the state BUSY, IDLE, or INVALID.

The RSSI above-level is maintained by monitoring the RSSI. If the RSSI is greater than or equal to ccaRssiThr, ccaEnergy is busy. If the RSSI is smaller than ccaRssiThr, ccaEnergy is IDLE. When an RSSI calculation has not yet been completed because the receiver started, ccaEnergy is INVALID.

The carrier-sense monitoring based on correlation value uses correlation peaks as defined for use in the SFD search algorithm in the receiver. If the number of correlation peaks observed in the last 8 symbol periods (32 µs) is greater than ccaOpt.corrThr, ccaCorr is BUSY; otherwise ccaCorr is IDLE. The value of ccaOpt.corrThr can be from 0 to 3. While the receiver is receiving a frame, ccaCorr is BUSY regardless of the observed correlation peaks. If the time since the receiver started is less than 8 symbol periods and the number of correlation peaks observed since the receiver started is less than or equal to ccaOpt.corrThr, ccaCorr is INVALID.

The carrier-sense monitoring based on sync found is maintained by the radio CPU as follows. If sync is obtained on the receiver, the radio CPU checks the PHY header to find the frame length. The radio CPU considers the channel to be busy for the duration of this frame. This check is done even if reception of the frame is stopped due to the frame filtering and sync search is restarted. If sync is found again while the channel is viewed as BUSY, the channel is viewed as Busy until both these frames have ended according to the observed frame lengths. The INVALID state is not used for ccaSync.

If the radio is transmitting an ACK or is suspended for running a TX operation, ccaEnergy, ccaCorr, and ccaSync are all BUSY.

The overall CCA state ccaState depends on the ccaEnEnergy, ccaEnCorr, and ccaEnSync bits of ccaOpt together with the ccaCorrOp and ccaSyncOp bits. The following rules apply for finding the ccaState (ccaTmp is a helper state in the description):

- If ccaEnEnergy = 0 and ccaEnCorr = 0 and ccaEnSync = 0, then ccaState = IDLE
- If ccaEnEnergy = 1 and ccaEnCorr = 0, then ccaTmp = ccaEnergy
- If ccaEnEnergy = 0 and ccaEnCorr = 1, then ccaTmp = ccaCorr
- If ccaEnEnergy = 1 and ccaEnCorr = 1 and ccaCorrOp = 0, then:
  - If either ccaEnergy or ccaCorr is BUSY, then ccaTmp = BUSY;
  - Otherwise, if either ccaEnergy or ccaCorr is INVALID, then ccaTmp = INVALID;
  - Otherwise, ccaTmp = IDLE
- If ccaEnEnergy = 1 and ccaEnCorr = 1 and ccaCorrOp = 1, then:
  - If either ccaEnergy or ccaCorr is IDLE, then ccaTmp = IDLE;
  - Otherwise, if either ccaEnergy or ccaCorr is Invalid, then ccaTmp = INVALID;
  - Otherwise, ccaTmp = BUSY
- If ccaEnEnergy = 0 and ccaEnCorr = 0 and ccaEnSync = 1, then ccaState = ccaSync
- Otherwise, if ccaEnSync = 1 and ccaSyncOp = 0, then:
  - If either ccaTmp or ccaSync is BUSY, then ccaState = BUSY;
  - Otherwise, if ccaTmp is Invalid, then ccaState = INVALID;
  - Otherwise, ccaState = IDLE
- Otherwise, if ccaEnSync = 1 and ccaSyncOp = 1, then:
  - If either ccaTmp or ccaSync is IDLE, then ccaState = IDLE;
  - Otherwise, if ccaTmp is INVALID, then ccaState = INVALID;
  - Otherwise, ccaState = BUSY

The ccaSync CCA state is required to be Idle for the overall CCA state to be IDLE, according to the IEEE 802.15.4 standard. Thus, to comply, ccaEnSync is 1 and cceSyncOp is 0.

CCA mode 1, as defined in the IEEE 802.15.4 standard, is implemented by setting ccaEnEnergy = 1 and ccaEnCorr = 0. CCA mode 2 is implemented by setting ccaEnEnergy = 0 and ccaEnCorr = 1. CCA mode 3 is implemented by setting ccaEnEnergy = 1 and ccaEnCorr = 1. With CCA mode 3, ccaCorrOp is allowed to be either 0 or 1; this distinguishes between the logical operator AND (1) and OR (0), as described in the IEEE 802.15.4 standard.

The CCA states and the current RSSI can be read by the system CPU by issuing the immediate command CMD_IEEE_CCA_REQ. If a CMD_IEEE_CSMA operation is running in the foreground, the radio CPU also monitors the CCA autonomously.

26.5.4.2 Energy Detect Scan Operation

The energy detect scan radio operation is a background-level operation that starts with the CMD_IEEE_ED_SCAN command and uses a command structure, as given in Table 26-61.

At the start of an RX operation, the radio CPU waits for the start trigger, then programs the frequency based on the channel parameter. If channel is 0xFF, the operation keeps running on an already-configured channel. This requires that the operation follows another receive operation or a synthesizer programming operation. If the frequency synthesizer is not running, the operation ends with an error. After programming the frequency, the radio CPU configures the receiver to receive IEEE 802.15.4 packets, but it does not store any received data.

While the receiver is running, CCA is updated as described in Section 26.5.4.1.5. When the demodulator obtains sync on a frame, the PHY header is read. This is used only for determining the carrier sense based on sync found, and sync search restarts immediately afterwards.

The energy detect scan operation ends under the same conditions as the RX operation, as described in Section 26.5.4.1.4. Before the operation ends, the radio CPU writes the maximum-observed RSSI during the energy detect scan operation to maxRssi.

26.5.4.3 CSMA-CA Operation

The CSMA-CA operation is a foreground-level operation that runs on top of a receive or energy-detect scan operation. If run on top of an energy-detect scan operation, this does not perform the energy-detect scan procedure, but starts a receiver without having to receive packets. This operation starts with the CMD_IEEE_CSMA command, and uses the command structure given in Table 26-62.

At the start of a CSMA-CA operation, the radio CPU waits for the start trigger.

The radio CPU maintains a variable *CW*, which initializes to csmaConfig.initCW.

If remainingPeriods is nonzero at the start of the command, the radio CPU delays for that number of backoff periods (default 320 µs) measured from the start trigger before proceeding. Otherwise, the radio CPU draws a pseudo-random number in the range 0 to $2^{(BE)-1}$, where BE is listed in Table 26-62. The radio CPU then waits that number of backoff periods from the start trigger before proceeding.

After this wait time, the radio CPU checks the CCA state from the background-level operation, as described in Section 26.5.4.1.5. If the CCA state was INVALID, the radio CPU waits before trying again. If csmaConfig.bSlotted = 1, the wait is for one backoff period, otherwise it waits until an RSSI result is available. If the CCA state was IDLE, the radio CPU decrements CW by 1, and if this results in a value of zero, the CSMA-CA operation ends with success. If this results in a nonzero value, the radio CPU waits one backoff period timed from the end of the wait time, and then checks the CCA state again as described previously.

If the channel was BUSY when the CCA state was checked, the radio CPU updates the variables as follows:

CW = csmaConfig.initCW; NB + = 1; BE + = min (BE + 1, macMaxBE);

If NB after this update is greater than macMaxCSMABackoffs, the CSMA-CA operation ends with failure. Otherwise, the radio CPU draws a random number of backoff periods to wait as described previously, and proceeds as before. If csmaConfig.bSlotted = 1, the wait is from the next backoff period after the end of the previous wait time; otherwise, the wait is from a configurable time after the end of the previous wait time.

Figure 26-7 shows the flow chart for the CSMA-CA operation.

In addition to the CSMA-CA operation ending with success or failure as previously described, the operation can end as a result of the end trigger given by endTrigger and endTime, or by a command. The commands that can end the CSMA-CA operation are the immediate commands CMD_ABORT, CMD_STOP, CMD_IEEE_ABORT_FG, and CMD_IEEE_STOP_FG. When the CSMA-CA operation ends, the radio CPU writes lastTimeStamp with the timer value at the end of the most recent wait period before a CCA check was done, and lastRssi with the RSSI value at that time. If the operation ended because of a timeout or stop command, the radio CPU writes remainingPeriods with the number of backoff periods remaining of the wait time. Otherwise, the radio CPU writes remainingPeriods to 0.

The pseudo-random algorithm is based on a maximum-length 16-bit linear-feedback shift register (LFSR). The seed is as provided in randomState. When the operation ends, the radio CPU writes the current state back to this field. If randomState is 0, the radio CPU self-seeds by initializing the LFSR to the 16 LSBs of the radio timer. There is some randomness to this value, but this is limited, especially for slotted CSMA-CA, and seeding with a true-random number (or a pseudo-random number based on a true-random seed) by the system CPU is therefore recommended. If the 16 LSBs of the radio timer are all 0, another fixed value is substituted.

Depending on csmaConfig.rxOffMode, the underlying RX operation may be suspended during the backoff before another CCA check, if there is enough time for it. The different values have the following meaning:

- rxOffMode = 0: The radio stays on during CSMA backoffs.
- rxOffMode = 1: If a frame is being received, an ACK being transmitted, or in the transition between those, the radio stays on. Otherwise, the radio switches off until the end of the backoff period.
- rxOffMode = 2: If a frame is being received, an ACK being transmitted, or in the transition between those, the radio stays on until the packet is fully received and the ACK is transmitted if applicable. After that, the radio switches off until the end of the backoff period.
- rxOffMode = 3: The radio switches off immediately at the beginning of a backoff period. This aborts a frame being received or an ACK being transmitted. The radio remains switched off until the end of the backoff period.

If the radio switches off this way, the receiver restarts sufficiently early for the next CCA operation to be done, and the radio only switches off it there is sufficient time. This feature can be used for power saving in systems that do not always need to be in RX. All modes except mode 0 may cause frames to be lost, at increasing probability.

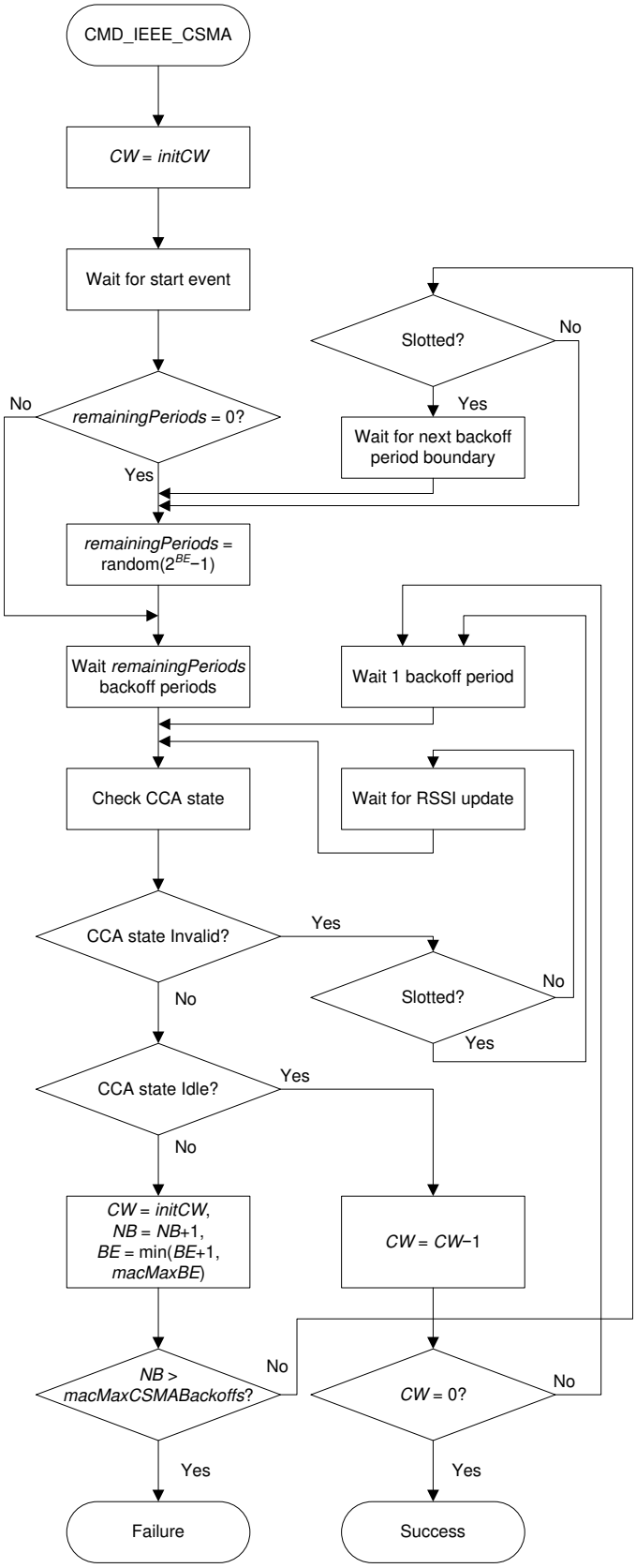


Figure 26-7. CSMA-CA Operation

For operation according to IEEE 802.15.4, the parameters must be initialized as follows before starting a new CSMA-CA operation:

1. randomState must be set to a random value
2. csmaConfig.initCW must be set to 2 for slotted CSMA-CA and 1 for unslotted CSMA-CA
3. csmaConfig.bSlotted must be set to 1 for slotted CSMA-CA and 0 for unslotted CSMA-CA
4. NB must be set to 0
5. BE must be set to macMinBE, except for slotted CSMA-CA with battery-life extension, where BE must be set to min (2, macMinBE)
6. remainingPeriods must be set to 0
7. macMaxBE and macMaxCSMABackoffs must be set to their corresponding MAC PIB attribute

For slotted CSMA-CS, startTrigger must be set up to occur on a backoff-slot boundary. For slotted CSMA-CA, the endTrigger must be set up to occur at the latest time that the transaction can be completed within the superframe, as specified in the IEEE 802.15.4 standard. If the CSMA-CA ends due to timeout, the CSMA can be restarted without modifying the parameters (except possibly the end time) at the next superframe.

Table 26-83 lists the causes of a CSMA-CA operation end. The status field of the command structure after the command has ended indicates the reason why the operation ended. In all cases, an FG_COMMAND_DONE interrupt is raised. In each case, it is indicated if the result is TRUE, FALSE, or ABORT. This indicates whether to start the next command (if any) in pNextOp, or to return to an IDLE state.

Table 26-83. End of CSMA-CA Operation

Condition	Status Code	Result
CSMA-CA operation finished with success	IEEE_DONE_OK	TRUE
CSMA-CA operation finished with failure	IEEE_DONE_BUSY	FALSE
End trigger occurred	IEEE_DONE_TIMEOUT	FALSE
Received CMD_STOP or CMD_IEEE_STOP_FG	IEEE_DONE_STOPPED	FALSE
Received CMD_ABORT or CMD_IEEE_ABORT_FG	IEEE_DONE_ABORT	ABORT
Background operation ended	IEEE_DONE_BGEND	ABORT
Observed illegal parameter	IEEE_ERROR_PAR	ABORT

When the operation ends, the time of the last CCA check (that is, the time written into lastTimeStamp) is defined as event 1, and may be used for timing subsequent chained operations.

26.5.4.4 Transmit Operation

The transmit operation is a foreground-level operation that transmits one packet. The operation is started with the CMD_IEEE_TX command, and uses the command structure given in Table 26-63.

When the radio CPU receives the command, it waits for the start trigger. Any background-level operation keeps running during this wait time. At the start trigger, the radio CPU suspends the receiver and configures the transmitter. It is assumed that the synthesizer is powered and calibrated, so if no background-level operation is running, the TX operation must be preceded with a calibrate synthesizer command. If the frequency synthesizer is not running, the operation ends with an error.

The transmitter transmits the payload found in the buffer pointer to pPayload, which consists of payloadLen bytes. If txOpt.payloadLenMsb is nonzero, this field is multiplied by 256 and added to payloadLen to create (for test purposes) a long frame that is not compliant with IEEE 802.15.4. If txOpt.bIncludePhyHdr is 0, the radio CPU inserts a PHY header automatically, calculated from the payload length. Otherwise, no PHY header is inserted by the radio CPU, so for IEEE 802.15.4 compliance, the first byte in the payload buffer must be the PHY header.

The payload is then transmitted as found in the payload buffer. If txOpt.bIncludeCrc is 0, the radio CPU appends two CRC bytes, calculated according to the IEEE 802.15.4 standard. Otherwise, no CRC is appended, so for IEEE 802.15.4 MAC compliance, the last two bytes in the payload buffer must be the MAC footer.
The transmit operation can be ended by one of the immediate commands CMD_ABORT, CMD_STOP, CMD_IEEE_ABORT_FG, and CMD_IEEE_STOP_FG. If CMD_ABORT or CMD_IEEE_ABORT_FG is received,

the transmission ends as soon as possible in the middle of the packet. If CMD_STOP or CMD_IEEE_STOP_BG is received while the radio CPU is waiting for the start trigger, the operation ends without any transmission; otherwise, the transmission is finished, but the end status and result differ as explained in the following.

When transmission of the packet starts, the trigger RAT time used for starting the modem is written to the timeStamp field by the radio CPU. This timestamp is delayed by the firmware-defined parameter startToTXRatOffset, compared to the configured start time of the CMD_IEEE_TX command. If the transmitter and receiver have synchronized RAT timers, this timestamp is the same as the timestamp appended to the RX entry element, as in Section 26.7.1, although with estimation uncertainty on the receiver side.

When the operation ends, the end time of the transmitted frame is defined as event 1, and may be used for timing subsequent chained operations.

Table 26-84 lists the causes of a transmit operation end. The status field of the command structure after the command has ended indicates the reason why the operation ended. In all cases, an FG_COMMAND_DONE interrupt is raised. In each case, it is indicated if the result is TRUE, FALSE, or ABORT. This indicates whether to start the next command (if any) in pNextOp, or to return to an IDLE state.

Table 26-84. End of Transmit Operation

Condition	Status Code	Result
Packet transmitted	IEEE_DONE_OK	TRUE
Received CMD_STOP or CMD_IEEE_STOP_FG, then finished transmitting if started	IEEE_DONE_STOPPED	FALSE
Received CMD_ABORT or CMD_IEEE_ABORT_FG	IEEE_DONE_ABORT	ABORT
Observed illegal parameter	IEEE_ERROR_PAR	ABORT

26.5.4.5 Receive Acknowledgment Operation

The receive-ACK operation is a foreground-level operation that runs on top of a receive operation. The operation starts with the CMD_IEEE_RX_ACK command, and uses the command structure listed in Table 26-64.

At the start of a receive-ACK operation, the radio CPU waits for the start trigger. If the receiver was suspended due to a TX operation before the receive-ACK operation, the background-level RX operation is not resumed until the start trigger occurs.

While the receive-ACK operation is running, the background-level RX operation runs normally. However, in addition to looking for the packets, the operation looks for ACK packets with the sequence number given in seqNo. The packet is stored in the receive queue only if configured to in the background-level receive operation (frameTypes.bAcceptFt2Ack = 1). If ACK packets are filtered out in the background RX operation, for an ACK packet the sequence number is received, and if it matches, also the FCS.

If the ACK packet with the requested sequence number is received, the FCS is checked. If the CRC is OK, the receive-ACK operation ends, otherwise it continues. If the ACK is received OK, the pending-data bit of the header is checked.

In addition to the receive-ACK operation ending after receiving the ACK as described previously, the operation can end as a result of the end trigger given by endTrigger and endTime, or by a command. The commands that can end the receive-ACK operation are the immediate commands CMD_ABORT, CMD_STOP, CMD_IEEE_ABORT_FG, and CMD_IEEE_STOP_FG.

A receive-ACK operation ends due to one of the causes listed in Table 26-85. The status field of the command structure after the command has ended indicates the reason why the operation ended. In all cases, an FG_COMMAND_DONE interrupt is raised. In each case, it is indicated if the result is TRUE, FALSE, or ABORT. This indicates whether to start the next command (if any) in pNextOp, or to return to an IDLE state.

Table 26-85. End of Receive ACK Operation

Condition	Status Code	Result
Requested ACK successfully received with pending data bit cleared	IEEE_DONE_ACK	FALSE
Requested ACK successfully received with pending data bit set	IEEE_DONE_ACKPEND	TRUE

Table 26-85. End of Receive ACK Operation (continued)

Condition	Status Code	Result
End trigger occurred	IEEE_DONE_TIMEOUT	FALSE
Received CMD_STOP or CMD_IEEE_STOP_FG	IEEE_DONE_STOPPED	FALSE
Received CMD_ABORT or CMD_IEEE_ABORT_FG	IEEE_DONE_ABORT	ABORT
Background operation ended	IEEE_DONE_BGEND	ABORT
Observed illegal parameter	IEEE_ERROR_PAR	ABORT

26.5.4.6 Abort Background-Level Operation Command

The abort background-level operation command is a foreground-level command that stops the command running in the background. The abort background-level operation command is defined as a foreground-operation command so that it has a start time, and so that it can be chained with other foreground-operation commands. The command is executed with the CMD_IEEE_ABORT_BG command and uses a command structure with only the minimum set of parameters.

At the start of an abort background-level operation, the radio CPU waits for the start trigger, then aborts the ongoing background-level receive or energy-detect scan operation.

The operation may be stopped by a command while waiting for the start trigger. The commands that can stop the operation are CMD_ABORT, CMD_STOP, CMD_IEEE_ABORT_FG, and CMD_IEEE_STOP_FG. The first two cause the background-level operation to stop regardless.

An abort background-level operation ends due to one of the causes listed in Table 26-86. The status field of the command structure after the command has ended indicates the reason why the operation ended. In all cases, an FG_COMMAND_DONE interrupt is raised. In each case, it is indicated if the result is TRUE, FALSE, or ABORT. This indicates whether to start the next command (if any) in pNextOp, or to return to an IDLE state.

Table 26-86. End of ABORT Background-Level Operation

Condition	Status Code	Result
Background level aborted	IEEE_DONE_OK	TRUE
Received CMD_STOP or CMD_IEEE_STOP_FG	IEEE_DONE_STOPPED	FALSE
Received CMD_ABORT or CMD_IEEEE_ABORT_FG	IEEE_DONE_ABORT	ABORT

26.5.5 Immediate Commands

26.5.5.1 Modify CCA Parameter Command

The CMD_IEEE_MOD_CCA command takes a command structure as defined in Table 26-65.

CMD_IEEE_MOD_CCA must only be sent while an RX or energy-detect scan operation is running. On reception, the radio CPU modifies the values of ccaRssiThr and ccaOpt for the running process into the values given by newCcaRssiThr and newCcaOpt, respectively. The radio CPU updates the command structure. The new settings are used for future CCA requests.

If the command is issued without an active or suspended background-level operation, the radio CPU returns the result ContextError in CMDSTA. If any of the parameters entered are illegal, the radio CPU returns the result ParError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.5.5.2 Modify Frame-Filtering Parameter Command

The CMD_IEEE_MOD_FILT command takes a command structure as defined in Table 26-66.

CMD_IEEE_MOD_FILT must only be sent while an RX operation is running. On reception, the radio CPU modifies the values of frameFiltOpt and frameTypes for the running process into the values given by newFrameFiltOpt and newFrameTypes, respectively. The radio CPU updates the command structure.

The new values of the frame-filtering options are used from the next time frame filtering is started. If autoAckEn or slottedAckEn are changed, the change applies from the next time reception of a packet ends.

If the command is issued without an active or suspended background-level RX operation, the radio CPU returns the result ContextError in CMDSTA. If any of the parameters entered are illegal, the radio CPU returns the result ParError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.5.5.3 Enable or Disable Source Matching Entry Command

The CMD_IEEE_MOD_SRC_MATCH command takes a command structure as defined in Table 26-66. CMD_IEEE_MOD_SRC_MATCH must only be sent while an RX operation is running. On reception, the radio CPU enables or disables the source-matching entry signaled in the command structure. If options.entryType is 0, the entry is extended-address entry in the structure pointed to by pExtEntryList, and if options.entryType is 1, the entry is short-address entry in the structure pointed to by pShortEntryList. The index of the entry is signaled in entryNo. If options.bEnable is 0, the entry is disabled, and if it is 1, the entry is enabled. The corresponding source pending bit is set to the value of options.srcMatch. The new values of the enable values are used from the next time source-matching is performed. The system CPU may modify the address of a disabled entry, but not an enabled one. If the command is issued without an active or suspended background-level RX operation, the radio CPU returns the result ContextError in CMDSTA. If any of the parameters entered are illegal, for example, pointing to a nonexistent entry, the radio CPU returns the result ParError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.5.5.4 Abort Foreground-Level Operation Command

CMD_IEEE_ABORT_FG is an immediate command that takes no parameters, and can thus be used as a direct command.

The CMD_IEEE_ABORT_FG command aborts the foreground-level operation while the background-level operation continues to run. For more detail, see the description of the foreground-level operations in Table 26-58.

If no foreground-level radio operation command is running, no action is taken. The result signaled in CMDSTA is DONE in all cases. If a foreground-level radio operation command was running, CMDSTA may be updated before the radio operation has ended.

26.5.5.5 Stop Foreground-Level Operation Command

CMD_IEEE_STOP_FG is an immediate command that takes no parameters, and can thus be used as a direct command.

The CMD_IEEE_STOP_FG command causes the foreground-level operation to stop gracefully, while the background-level operation continues to run. For more detail, see the description of the foreground-level operations in Table 26-58.

If no foreground-level radio operation command is running, no action is taken. The result signaled in CMDSTA is DONE in all cases. If a foreground-level radio operation command was running, CMDSTA may be updated before the radio operation has ended.

26.5.5.6 Request CCA and RSSI Information Command

The CMD_IEEE_CCA_REQ command takes a command structure as defined in Table 26-68.

CMD_IEEE_CCA_REQ must only be sent while an RX or energy-detect scan operation is running. On reception, the radio CPU writes the following figures back into the command structure:

- currentRssi is set to the RSSI number currently available from the demodulator
- maxRssi is set to the maximum RSSI observed because the background-level operation was started
- ccaState is set to the CCA state according to the current CCA options, refer to Section 26.5.4.1.5
- ccaEnergy is set to the energy-detect CCA state according to Section 26.5.4.1.5
- ccaCorr is set to the correlator-based carrier-sense CCA state according to Section 26.5.4.1.5
- ccaSync is set to the sync found-based carrier-sense CCA state according to Section 26.5.4.1.5

If no valid RSSI is found when the request is sent, the currentRssi and maxRssi returned indicate this by using a special value (0x80).

If the command is issued without an active or suspended background-level RX operation, the radio CPU returns the result ContextError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.6 Bluetooth® Low Energy

This section describes Bluetooth® Low Energy specific command structures, data handling, radio operation commands, and immediate commands.

26.6.1 Bluetooth® Low Energy Commands

The Bluetooth® Low Energy specific radio operation commands for legacy mode are defined in Table 26-87. These commands should be used for running Bluetooth® versions 4.0, 4.1, 4.2, and for legacy features of Bluetooth® 5.0.

Table 26-87. Legacy Bluetooth® Low Energy Radio Operation Commands

ID	Command Name	Description
0x1801	CMD_BLE_SLAVE	Start slave operation.
0x1802	CMD_BLE_MASTER	Start master operation.
0x1803	CMD_BLE_ADV	Start connectable undirected advertiser operation.
0x1804	CMD_BLE_ADV_DIR	Start connectable directed advertiser operation.
0x1805	CMD_BLE_ADV_NC	Start the nonconnectable advertiser operation.
0x1806	CMD_BLE_ADV_SCAN	Start scannable undirected advertiser operation.
0x1807	CMD_BLE_SCANNER	Start scanner operation.
0x1808	CMD_BLE_INITIATOR	Start initiator operation.
0x1809	CMD_BLE_GENERIC_RX	Receive generic packets (used for PHY test or packet sniffing).
0x180A	CMD_BLE_TX_TEST	Transmit PHY test packets.

Table 26-88 defines the Bluetooth® Low Energy specific radio operation commands for Bluetooth® 5 mode.

Table 26-88. Bluetooth® Low Energy 5 Radio Operation Commands

ID	Command name	Description
0x1820	CMD_BLE5_RADIO_SETUP	Set up radio in Bluetooth® 5 mode with the ability to switch between PHYs.
0x1821	CMD_BLE5_SLAVE	Start slave operation.
0x1822	CMD_BLE5_MASTER	Start master operation.
0x1823	CMD_BLE5_ADV_EXT	Start extended advertiser operation on primary channel.
0x1824	CMD_BLE5_ADV_AUX	Start extended advertiser operation on secondary channel.
0x1827	CMD_BLE5_SCANNER	Start scanner operation.
0x1828	CMD_BLE5_INITIATOR	Start initiator operation.
0x1829	CMD_BLE5_GENERIC_RX	Receive generic packets (used for PHY test or packet sniffing).
0x182A	CMD_BLE5_TX_TEST	Transmit PHY test packets.

Table 26-89 defines the Bluetooth®Low Energy specific immediate command.

Table 26-89. Bluetooth® Low Energy Immediate Command

ID	Command Name	Description
0x1001	CMD_BLE_ADV_PAYLOAD	Modify the payload used in advertiser operations.

26.6.1.1 Command Data Definitions

This section defines data types used in describing the data structures used for communication between the system CPU and the radio CPU. The data structures are listed with tables. The byte index is the offset from the pointer to that structure. Multibyte fields are little endian, and half-word or word alignment is required. For bit numbering, 0 is the least significant bit. The R/W column is used as follows:

- R: The system CPU can read a result back; the radio CPU will not read the field.
- W: The system CPU shall write a value, the radio CPU will read it and will not modify the value.
- R/W: The system CPU shall write an initial value, the radio CPU may modify the initial value.

26.6.1.1.1 Bluetooth® Low Energy Command Structures

Table 26-90. Legacy Bluetooth® Low Energy Radio Operation Command Structure(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number
2–3	status			R/W	An integer telling the status of the command. This value is updated by the radio CPU during operation and may be read by the system CPU at any time.
4–7	pNextOp			W	Pointer to the next operation to run after this operation is done
8–11	startTime			W	Absolute or relative start time (depending on the value of startTrigger)
12	startTrigger				Identification of the trigger that starts the operation.
13	condition	0–3	rule	W	Condition for running next command: Rule for how to proceed.
		4–7	nSkip	W	Number of skips + 1 if the rule involves skipping. 0: same 1: next 2: skip next ...

Table 26-90. Legacy Bluetooth® Low Energy Radio Operation Command Structure(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	channel			W	Channel to use 0–39: Bluetooth® Low Energy advertising/data channel number 60–207: Custom frequency; (2300 + channel) MHz 255: Use existing frequency Others: Reserved
15	whitening	0–6	init	W	If bOverride = 1 or custom frequency is used: 0: Do not use whitening Other value: Initialization for 7-bit LFSR whitener
		7	bOverride	W	0: Use default whitening for Bluetooth® Low Energy advertising/data channels 1: Override whitening initialization with value of init
16–19	pParams			W	Pointer to command specific parameter structure
20–23	pOutput			W	Pointer to command specific output structure

(1) This command structure shall be used for all the radio operation commands for legacy Bluetooth® Low Energy support that are listed in Table 26-87.

Table 26-91. Bluetooth® Low Energy 5 Radio Operation Command Structure(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number 0x1821
2–3	status			R/W	An integer telling the status of the command. This value is updated by the radio CPU during operation and may be read by the system CPU at any time.
4–7	pNextOp			W	Pointer to the next operation to run after this operation is done
8–11	startTime			W	Absolute or relative start time (depending on the value of startTrigger)
12	startTrigger				Identification of the trigger that starts the operation
13	condition	0–3	rule	W	Condition for running next command: Rule for how to proceed
		4–7	nSkip	W	Number of skips + 1 if the rule involves skipping. 0: same 1: next 2: skip next ...
14	channel			W	Channel to use 0–39: Bluetooth® Low Energyy advertising/data channel number 60–207: Custom frequency; (2300 + channel) MHz 255: Use existing frequency Others: Reserved
15	whitening	0–6	init	W	If bOverride = 1 or custom frequency is used: 0: Do not use whitening Other value: Initialization for 7-bit LFSR whitener
		7	bOverride	W	0: Use default whitening for Bluetooth® Low Energy advertising/data channels 1: Override whitening initialization with value of init
16	phyMode	0–1	mainMode	W	PHY to use: 0: 1 Mbps 1: 2 Mbps 2: Coded 3: Reserved
		2–7	coding	W	Coding to use for TX if coded PHY is selected. See Section 26.8.3
17	rangeDelay			W	Number of RAT ticks to add to the listening time after T_IFS

Table 26-91. Bluetooth® Low Energy 5 Radio Operation Command Structure(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
18–19	txPower			W	Transmit power to use (overrides the one given in radio setup). 0x0000: Use default TX power.
20–23	pParams			W	Pointer to command specific parameter structure
24–27	pOutput			W	Pointer to command specific output structure

(1) This command structure shall be used for all the radio operation commands for Bluetooth® Low Energy 5 support, listed in Table 26-88, except CMD_BLE5_RADIO_SETUP.

Table 26-92. Bluetooth® Low Energy 5 Radio Setup Command Structure(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	commandNo			W	The command ID number 0x1820
2–3	status			R/W	An integer telling the status of the command. This value is updated by the radio CPU during operation and may be read by the system CPU at any time.
4–7	pNextOp			W	Pointer to the next operation to run after this operation is done
8–11	startTime			W	Absolute or relative start time (depending on the value of startTrigger)
12	startTrigger				Identification of the trigger that starts the operation
13	condition	0–3	rule	W	Condition for running next command: Rule for how to proceed
		4–7	nSkip	W	Number of skips + 1 if the rule involves skipping. 0: same 1: next 2: skip next ...
14	defaultPhy	0–1	mainMode	W	PHY to use for non-BLE commands: 0: 1 Mbps 1: 2 Mbps 2: Coded 3: Reserved
		2	coding	W	Coding to use for TX if coded PHY is selected for non-BLE commands 0: S = 8 (125 kbps) 1: S = 2 (500 kbps)
		3–7			Reserved
15					Reserved
16–17	config				Configuration options
18–19	txPower				Default transmit power
20–23	pRegOverrideCommon			W	Pointer to a list of hardware and configuration registers to override during common initialization. If NULL, no override is used.
24–27	pRegOverride1Mbps			W	Pointer to a list of hardware and configuration registers to override when selecting 1 Mbps PHY mode. If NULL, no override is used.
28–31	pRegOverride2Mbps			W	Pointer to a list of hardware and configuration registers to override when selecting 2-Mbps PHY mode. If NULL, no override is used.
32–35	pRegOverrideCoded			W	Pointer to a list of hardware and configuration registers to override when selecting coded PHY mode. If NULL, no override is used.

(1) This command structure shall be used for CMD_BLE5_RADIO_SETUP.

Table 26-93. Update Advertising Payload Command(1)

Byte Index	Field Name	Type	Description
0–1	commandNo	W	The command ID number 0x1001
2	payloadType	W	0: Advertising data 1: Scan response data
3	newLen	W	Length of the new payload
4–7	pNewData	W	Pointer to the buffer containing the new data
8–11	pParams	W	Pointer to the parameter structure to update

(1) This command structure shall be used for CMD_BLE_ADV_PAYLOAD.

26.6.1.2 Parameter Structures

Table 26-94. Legacy Slave Command(1)

Byte Index	Field Name	Type	Description
0–3	pRxQ	W	Pointer to receive queue
4–7	pTxQ	W	Pointer to transmit queue
8	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-114)
9	seqStat	R/W	Sequence number status (see Table 26-115)
10	maxNack	W	Maximum number of NACKs received before operation ends. 0: No limit
11	maxPkt	W	Maximum number of packets transmitted in the operation before it ends. 0: No limit
12–15	accessAddress	W	Access address used on the connection
16	crcInit0	W	CRC initialization value used on the connection – least significant byte
17	crcInit1	W	CRC initialization value used on the connection – middle byte
18	crcInit2	W	CRC initialization value used on the connection – most significant byte
19	timeoutTrigger	W	Trigger that defines timeout of the first receive operation
20–23	timeoutTime	W	Time used together with timeoutTrigger that defines timeout of the first receive operation
24–26			Reserved
27	endTrigger	W	Trigger that causes the device to end the connection event as soon as allowed
28–31	endTime	W	Time used together with endTrigger that causes the device to end the connection event as soon as allowed

(1) This parameter structure is used for legacy slave commands, CMD_BLE_SLAVE.

Table 26-95. Legacy Master Command(1)

Byte Index	Field Name	Type	Description
0–3	pRxQ	W	Pointer to receive queue
4–7	pTxQ	W	Pointer to transmit queue
8	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-114)
9	seqStat	R/W	Sequence number status (see Table 26-115)
10	maxNack	W	Maximum number of NACKs received before operation ends. 0: No limit
11	maxPkt	W	Maximum number of packets transmitted in the operation before it ends. 0: No limit
12–15	accessAddress	W	Access address used on the connection
16	crcInit0	W	CRC initialization value used on the connection – least significant byte
17	crcInit1	W	CRC initialization value used on the connection – middle byte
18	crcInit2	W	CRC initialization value used on the connection – most significant byte
19	endTrigger	W	Trigger that causes the device to end the connection event as soon as allowed

Table 26-95. Legacy Master Command(1) (continued)

Byte Index	Field Name	Type	Description
20–23	endTime	W	Time used together with endTrigger that causes the device to end the connection event as soon as allowed

(1) This parameter structure is used for legacy master commands, CMD_BLE_MASTER.

Table 26-96. Legacy Advertiser Commands(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries (see Table 26-114)
5	advConfig	0–1	advFilterPolicy	W	Advertiser filter policy 0: Process scan and connect requests from all devices 1: Process connect requests from all devices and only scan requests from devices that are in the whitelist 2: Process scan requests from all devices and only connect requests from devices that are in the whitelist 3: Process scan and connect requests only from devices in the whitelist
		2	deviceAddrType	W	The type of the device address: public (0) or random (1)
		3	peerAddrType	W	Directed advertiser: The type of the peer address: public (0) or random (1)
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length 1: Discard messages with illegal length for the given packet type
		5	chSel	W	0: Do not report support of Channel Selection Algorithm 2 1: Report support of Channel Selection Algorithm 2
		6			Reserved
		7	rpaMode	W	Resolvable private address mode 0: Normal operation 1: Use whitelist for a received RPA regardless of filter policy
6	advLen			W	Size of advertiser data
7	scanRspLen			W	Size of scan response data
8–11	pAdvData			W	Pointer to buffer containing ADV*_IND data
12–15	pScanRspData			W	Pointer to buffer containing SCAN_RSP data
16–19	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by advConfig.deviceAddrType is inverted.
20–23	pWhiteList			W	Pointer (with least significant bit set to 0) to whitelist or peer address (directed advertiser). If least significant bit is 1, the address type given by advConfig.peerAddrType is inverted.
24–26					Reserved
27	endTrigger			W	Trigger that causes the device to end the advertiser event as soon as allowed
28–31	endTime			W	Time used together with endTrigger that causes the device to end the advertiser event as soon as allowed

(1) This parameter structure is used for all the legacy advertiser commands, CMD_BLE_ADV, CMD_BLE_ADV_DIR, CMD_BLE_ADV_NC, and CMD_BLE_ADV_SCAN.

Table 26-97. Legacy Scanner Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries
5	scanConfig	0	scanFilterPolicy	W	Scanning filter policy regarding advertiser address 0: Accept all advertisement packets. 1: Accept only advertisement packets from devices where the address of the advertiser is in the whitelist.
		1	bActiveScan	W	0: Passive scan 1: Active scan
		2	deviceAddrType	W	The type of the device address: public (0) or random (1)
		3	rpaFilterPolicy	W	Filter policy for TargetA for ADV_DIRECT_IND messages 0: Accept only TargetA that matches own address. 1: Also accept all resolvable private addresses.
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length 1: Discard messages with illegal length for the given packet type
		5	bAutoWlIgnore	W	0: Do not set ignore bit in whitelist from radio CPU 1: Automatically set ignore bit in whitelist
		6	bEndOnRpt	W	0: Continue scanner operation after each reporting ADV*_IND or sending SCAN_RSP 1: End scanner operation after each reported ADV*_IND and potentially SCAN_RSP
		7	rpaMode	W	Resolvable private address mode 0: Normal operation 1: Use whitelist for a received RPA regardless of filter policy
6–7	randomState			R/W	State for pseudo-random number generation used in backoff procedure
8–9	backoffCount			R/W	Parameter backoffCount used in backoff procedure (see the Bluetooth® Specification documents in Related Documentation).
10	backoffPar	0–3	logUpperLimit	R/W	Binary logarithm of parameter upperLimit used in scanner backoff procedure
		4	bLastSucceeded	R/W	1 if the last SCAN_RSP was successfully received and upperLimit not changed
		5	bLastFailed	R/W	1 if reception of the last SCAN_RSP failed and upperLimit was not changed
		6–7			Reserved
11	scanReqLen			W	Size of scan request data
12–15	pScanReqData			W	Pointer to buffer containing SCAN_REQ data
16–19	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by scanConfig.deviceAddrType is inverted.
20–23	pWhiteList			W	Pointer to whitelist
24–25					Reserved
26	timeoutTrigger			W	Trigger that causes the device to stop receiving as soon as allowed
27	endTrigger			W	Trigger that causes the device to stop receiving as soon as allowed

Table 26-97. Legacy Scanner Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
28–31	timeoutTime			W	Time used together with timeoutTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_RXTIMEOUT
32–35	endTime			W	Time used together with endTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_ENDED

(1) This parameter structure is used for the legacy scanner command, CMD_BLE_SCANNER.

Table 26-98. Legacy Initiator Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries (see Table 26-114)
5	initConfig	0	bUseWhiteList	W	Initiator filter policy 0: Use specific peer address 1: Use whitelist
		1	bDynamicWinOffset		0: No dynamic WinOffset insertion 1: Use dynamic WinOffset insertion
		2	deviceAddrType	W	The type of the device address: public (0) or random (1)
		3	peerAddrType	W	The type of the peer address: public (0) or random (1)
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length 1: Discard messages with illegal length for the given packet type
		5	chSel	W	0: Do not report support of Channel Selection Algorithm 2 1: Report support of Channel Selection Algorithm 2
		6			Reserved
		7			Reserved
6					Reserved
7	connectReqLen			W	Size of connect request data
8–11	pConnectReqData			W	Pointer to buffer containing LLData to go in the CONNECT_IND (CONNECT_REQ)
12–15	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by initConfig.deviceAddrType is inverted.
16–19	pWhiteList			W	Pointer (with least significant bit set to 0) to whitelist or peer address. If least significant bit is 1, the address type given by initConfig.peerAddrType is inverted.
20–23	connectTime			R/W	Indication of timer value of the first possible start time of the first connection event. Set to the calculated value if a connection is made and to the next possible connection time (see Table 26-110) if not.
24–25					Reserved
26	timeoutTrigger			W	Trigger that causes the device to stop receiving as soon as allowed
27	endTrigger			W	Trigger that causes the device to stop receiving as soon as allowed

Table 26-98. Legacy Initiator Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
28–31	timeoutTime			W	Time used together with timeoutTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_RXTIMEOUT
32–35	endTime			W	Time used together with endTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_ENDED

(1) This parameter structure is used for the legacy initiator command, CMD_BLE_INITIATOR.

Table 26-99. Generic RX Command(1)

Byte Index	Field Name	Type	Description
0–3	pRxQ	W	Pointer to receive queue. May be NULL; if so, received packets are not stored.
4	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-114)
5	bRepeat	W	0: End operation after receiving a packet 1: Restart receiver after receiving a packet
6–7			Reserved
8–11	accessAddress	W	Access address used on the connection
12	crcInit0	W	CRC initialization value used on the connection – least significant byte
13	crcInit1	W	CRC initialization value used on the connection – middle byte
14	crcInit2	W	CRC initialization value used on the connection – most significant byte
15	endTrigger	W	Trigger that causes the device to end the RX operation
16–19	endTime	W	Time used together with endTrigger that causes the device to end the RX operation.

(1) This parameter structure is used both for legacy and Bluetooth® Low Energy 5 generic receiver commands, CMD_BLE_GENERIC_RX and CMD_BLE5_GENERIC_RX.

Table 26-100. TX Test Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	numPackets			W	Number of packets to transmit 0: Transmit unlimited number of packets
2	payloadLength			W	The number of payload bytes in each packet.
3	packetType			W	The packet type to be used, encoded according to the Bluetooth® 5.0 Specification (see Related Documentation).
4–7	period			W	Number of radio timer cycles between the start of each packet
8	config	0	bOverrideDefault	W	0: Use default packet encoding 1: Override packet contents
		1	bUsePrbs9	W	If bOverride is 1: 0: No PRBS15 encoding of packet 1: Use PRBS9 encoding of packet
		2	bUsePrbs15	W	If bOverride is 1: 0: No PRBS15 encoding of packet 1: Use PRBS15 encoding of packet
		3–7			Reserved
9	byteVal			W	If config.bOverride is 1, value of each byte to be sent
10					Reserved
11	endTrigger			W	Trigger that causes the device to end the Test TX operation

Table 26-100. TX Test Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
12–15	endTime			W	Time used together with endTrigger that causes the device to end the Test TX operation

(1) This parameter structure is used both for legacy and Bluetooth® Low Energy 5 transmitter test commands, CMD_BLE_TX_TEST and CMD_BLE5_TX_TEST.

Table 26-101. Bluetooth® 5 Slave Command(1)

Byte Index	Field Name	Type	Description
0–3	pRxQ	W	Pointer to receive queue
4–7	pTxQ	W	Pointer to transmit queue
8	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-114)
9	seqStat	RW	Sequence number status (see Table 26-115)
10	maxNack	W	Maximum number of NACKs received before operation ends. 0: No limit
11	maxPkt	W	Maximum number of packets transmitted in the operation before it ends. 0: No limit
12–15	accessAddress	W	Access address used on the connection
16	crcInit0	W	CRC initialization value used on the connection – least significant byte
17	crcInit1	W	CRC initialization value used on the connection – middle byte
18	crcInit2	W	CRC initialization value used on the connection – most significant byte
19	timeoutTrigger	W	Trigger that defines timeout of the first receive operation
20–23	timeoutTime	W	Time used together with timeoutTrigger that defines timeout of the first receive operation
24	maxRxPktLen	W	Maximum packet length currently allowed for received packets on the connection
25	maxLenLowRate	W	Maximum packet length for which using S = 8 (125 kbps) is allowed when transmitting. 0: No limit
26			Reserved
27	endTrigger	W	Trigger that causes the device to end the connection event as soon as allowed
28–31	endTime	W	Time used together with endTrigger that causes the device to end the connection event as soon as allowed

(1) This parameter structure is used for the Bluetooth® 5 slave, CMD_BLE5_SLAVE.

Table 26-102. Bluetooth® 5 Master Command(1)

Byte Index	Field Name	Type	Description
0–3	pRxQ	W	Pointer to receive queue
4–7	pTxQ	W	Pointer to transmit queue
8	rxConfig	W	Configuration bits for the receive queue entries (see Table 26-114)
9	seqStat	RW	Sequence number status (see Table 26-115)
10	maxNack	W	Maximum number of NACKs received before operation ends. 0: No limit
11	maxPkt	W	Maximum number of packets transmitted in the operation before it ends. 0: No limit
12–15	accessAddress	W	Access address used on the connection
16	crcInit0	W	CRC initialization value used on the connection – least significant byte
17	crcInit1	W	CRC initialization value used on the connection – middle byte
18	crcInit2	W	CRC initialization value used on the connection – most significant byte
19	endTrigger	W	Trigger that causes the device to end the connection event as soon as allowed
20–23	endTime	W	Time used together with endTrigger that causes the device to end the connection event as soon as allowed
24	maxRxPktLen	W	Maximum packet length currently allowed for received packets on the connection

Table 26-102. Bluetooth® 5 Master Command(1) (continued)

Byte Index	Field Name	Type	Description
25	maxLenLowRate	W	Maximum packet length for which using S = 8 (125 kbps) is allowed when transmitting. 0: No limit

(1) This parameter structure is used for the Bluetooth® 5 master, CMD_BLE5_MASTER.

Table 26-103. Extended Advertiser Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0	advConfig	0–1			Reserved
		2	deviceAddrType	W	The type of the device address public (0) or random (1)
		3–7			Reserved
1–2					Reserved
3	auxPtrTargetType			W	Number indicating reference for auxPtrTargetTime. Takes same values as trigger types, but only TRIG_ABSTIME and TRIG_REL_* are allowed.
4–7	auxPtrTargetTime			W	Time of start of packet to which auxPtr points
8–11	pAdvPkt			W	Pointer to extended advertising packet for the ADV_EXT_IND packet
12–15	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by advConfig.deviceAddrType is inverted.

(1) This parameter structure is used for the Bluetooth® Low Energy 5 extended advertiser command, CMD_BLE5_ADV_EXT. For definition of the structure pointed to by pAdvPkt, see Table 26-121.

Table 26-104. Secondary Channel Advertiser Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries (see Table 26-114)

Table 26-104. Secondary Channel Advertiser Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
5	advConfig	0–1	advFilterPolicy	W	Advertiser filter policy 0: Process scan and connect requests from all devices. 1: Process connect requests from all devices and only scan requests from devices that are in the whitelist. 2: Process scan requests from all devices and only connect requests from devices that are in the whitelist. 3: Process scan and connect requests only from devices in the whitelist.
		2	deviceAddrType	W	The type of the device address public (0) or random (1)
		3	targetAddrType	W	Directed secondary advertiser: The type of the target address public (0) or random (1)
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length. 1: Discard messages with illegal length for the given packet type.
		5	bDirected	W	0: Advertiser is undirected: pWhiteList points to a whitelist 1: Advertiser is directed: pWhiteList points to a single device address.
		6			Reserved
		7	rpaMode	W	Resolvable private address mode 0: Normal operation 1: Use whitelist for a received RPA regardless of filter policy.
6					Reserved
7	auxPtrTargetType			W	Number indicating reference for auxPtrTargetTime. Takes same values as trigger types, but only TRIG_ABSTIME and TRIG_REL_* are allowed.
8–11	auxPtrTargetTime			W	Time of start of packet to which auxPtr points
12–15	pAdvPkt			W	Pointer to extended advertising packet for the ADV_AUX_IND packet
16–19	pRspPkt			W	Pointer to extended advertising packet for the AUX_SCAN_RSP or AUX_CONNECT_RSP packet (may be NULL if not applicable)
20–23	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by advConfig.deviceAddrType is inverted.
24–27	pWhiteList			W	Pointer (with least significant bit set to 0) to whitelist or peer address (directed advertiser). If least significant bit is 1, the address type given by advConfig.peerAddrType is inverted.

(1) This parameter structure is used for the Bluetooth® Low Energy 5 secondary channel advertiser command, CMD_BLE5_ADV_AUX. For definition of the structure pointed to by pAdvPkt and pRspPkt, see Table 26-121.

Table 26-105. Bluetooth® 5 Scanner Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries (see Table 26-114)

Table 26-105. Bluetooth® 5 Scanner Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
5	scanConfig	0	scanFilterPolicy	W	Scanning filter policy regarding advertiser address 0: Accept all advertisement packets. 1: Accept only advertisement packets from devices where the address of the advertiser is in the whitelist.
		1	bActiveScan	W	0: Passive scan 1: Active scan
		2	deviceAddrType	W	The type of the device address public (0) or random (1)
		3	rpaFilterPolicy	W	Filter policy for initA of ADV_DIRECT_IND messages 0: Accept only initA that matches own address. 1: Also accept all resolvable private addresses.
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length. 1: Discard messages with illegal length for the given packet type.
		5	bAutoWlIgnore	W	0: Do not set ignore bit in whitelist from radio CPU for legacy packets. 1: Automatically set ignore bit in whitelist for legacy packets.
		6	bEndOnRpt	W	0: Continue scanner operation after each reporting ADV*_IND or sending SCAN_RSP. 1: End scanner operation after each reported ADV*_IND and potentially SCAN_RSP.
		7	rpaMode	W	Resolvable private address mode 0: Normal operation 1: Use whitelist for a received RPA regardless of filter policy.
6–7	randomState			R/W	State for pseudo-random number generation used in backoff procedure
8–9	backoffCount			R/W	Parameter backoffCount used in backoff procedure, see the Bluetooth® Specification listed in Related Documentation.
10	backoffPar	0–3	logUpperLimit	R/W	Binary logarithm of parameter upperLimit used in scanner backoff procedure
		4	bLastSucceeded	R/W	1 if the last SCAN_RSP was successfully received and upperLimit not changed
		5	bLastFailed	R/W	1 if reception of the last SCAN_RSP failed and upperLimit was not changed
		6–7			Reserved
11	extFilterConfig	0	bCheckAdi	W	0: Do not perform ADI filtering. 1: Perform ADI filtering on packets where ADI is present.
		1	bAutoAdiUpdate	W	0: Do not update ADI entries in radio CPU. 1: Automatically update ADI entry for received packets with AdvDataIndex.
		2	bApplyDuplicateFiltering	W	0: Do not apply duplicate filtering based on device address for extended advertiser packets. 1: Apply duplicate filtering based on device address for extended advertiser packets with no ADI field.
		3	bAutoWlIgnore	W	0: Do not set ignore bit in whitelist from radio CPU for extended advertising packets. 1: Automatically set ignore bit in whitelist for extended advertising packets.
		4–7			Reserved
12–15					Reserved

Table 26-105. Bluetooth® 5 Scanner Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
16–19	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by scanConfig.deviceAddrType is inverted.
20–23	pWhiteList			W	Pointer to whitelist
24–27	pAdiList			W	Pointer to advDataInfo list
28–29	maxWaitTimeForAuxCh			W	Maximum wait time for switching to secondary scanning within the command. If the time to the start of the event is greater than this, the command will end with BLE_DONE_AUX. If it is smaller, the radio will automatically switch to the correct channel and PHY.
30	timeoutTrigger			W	Trigger that causes the device to stop receiving as soon as allowed
31	endTrigger			W	Trigger that causes the device to stop receiving as soon as allowed
32–35	timeoutTime			W	Time used together with timeoutTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_RXTIMEOUT
36–39	endTime			W	Time used together with endTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_ENDED
40–43	rxStartTime			R	The time needed to start RX in order to receive the packet
44–45	rxListenTime			R	The time needed to listen in order to receive the packet. 0: No AUX packet
46	channelNo			R	The channel number used for secondary advertising
47	phyMode			R	PHY to use on secondary channel: 0: 1 Mbps 1: 2 Mbps 2: Coded Others: Reserved

(1) This parameter structure is used for the Bluetooth® 5 scanner command, CMD_BLE5_SCANNER.

Table 26-106. Bluetooth® 5 Initiator Command(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–3	pRxQ			W	Pointer to receive queue
4	rxConfig			W	Configuration bits for the receive queue entries (see Table 26-114)

Table 26-106. Bluetooth® 5 Initiator Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
5	initConfig	0	bUseWhiteList	W	Initiator filter policy 0: Use specific peer address. 1: Use whitelist.
		1	bDynamicWinOffset		1: Use dynamic WinOffset insertion.
		2	deviceAddrType	W	The type of the device address public (0) or random (1)
		3	peerAddrType	W	The type of the peer address public (0) or random (1)
		4	bStrictLenFilter	W	0: Accept any packet with a valid advertising packet length. 1: Discard messages with illegal length for the given packet type.
		5	chSel	W	0: Do not report support of Channel Selection Algorithm 2 in CONNECT_IND. 1: Report support of Channel Selection Algorithm 2 in CONNECT_IND.
		6			Reserved
		7			Reserved
6–7	randomState			R/W	State for pseudo-random number generation used in backoff procedure
8–9	backoffCount			R/W	Parameter backoffCount used in backoff procedure, see the Bluetooth® specification listed in Related Documentation.
10	backoffPar	0–3	logUpperLimit	R/W	Binary logarithm of parameter upperLimit used in scanner backoff procedure
		4	bLastSucceeded	R/W	1 if the last SCAN_RSP was successfully received and upperLimit not changed
		5	bLastFailed	R/W	1 if reception of the last SCAN_RSP failed and upperLimit was not changed
		6–7			Reserved
11	connectReqLen			W	Size of connect request data
12–15	pConnectReqData			W	Pointer to buffer containing LLData to go in the CONNECT_IND or AUX_CONNECT_REQ packet
16–19	pDeviceAddress			W	Pointer (with least significant bit set to 0) to device address used for this device. If least significant bit is 1, the address type given by initConfig.deviceAddrType is inverted.
20–23	pWhiteList			W	Pointer (with least significant bit set to 0) to whitelist or peer address. If least significant bit is 1, the address type given by initConfig.peerAddrType is inverted.
24–27	connectTime			R/W	Indication of timer value of the first possible start time of the first connection event. Set to the calculated value if a connection is made and to the next possible connection time if not.
28–29	maxWaitTimeForAuxCh			W	Maximum wait time for switching to secondary scanning within the command. If the time to the start of the event is greater than this, the command will end with BLE_DONE_AUX. If it is smaller, the radio will automatically switch to the correct channel and PHY.
30	timeoutTrigger			W	Trigger that causes the device to stop receiving as soon as allowed
31	endTrigger			W	Trigger that causes the device to stop receiving as soon as allowed

Table 26-106. Bluetooth® 5 Initiator Command(1) (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
32–35	timeoutTime			W	Time used together with timeoutTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_RXTIMEOUT
36–39	endTime			W	Time used together with endTrigger that causes the device to stop receiving as soon as allowed, ending with BLE_DONE_ENDED
40–43	rxStartTime			R	The time needed to start RX in order to receive the packet
44–45	rxListenTime			R	The time needed to listen in order to receive the packet. 0: No AUX packet
46	channelNo			R	The channel number used for secondary advertising
47	phyMode			R	PHY to use on secondary channel: 0: 1 Mbps 1: 2 Mbps 2: Coded Others: Reserved
40	auxChRes				

(1) This parameter structure is used for the Bluetooth® 5 initiator command, CMD_BLE5_INITIATOR.

26.6.1.3 Output Structures

Table 26-107. Master and Slave Commands

Byte Index	Field Name	Type	Description
0	nTx	R/W	Total number of packets (including auto-empty and retransmissions) that have been transmitted
1	nTxAck	R/W	Total number of transmitted packets (including auto-empty) that have been ACKed
2	nTxCtrl	R/W	Number of unique LL control packets from the TX queue that have been transmitted
3	nTxCtrlAck	R/W	Number of LL control packets from the TX queue that have been finished (ACKed)
4	nTxCtrlAckAck	R/W	Number of LL control packets that have been ACKed and where an ACK is sent in response
5	nTxRetrans	R/W	Number of retransmissions that is done
6	nTxEntryDone	R/W	Number of packets from the TX queue that have been finished (ACKed)
7	nRxOk	R/W	Number of packets that have been received with payload, CRC OK and not ignored
8	nRxCtrl	R/W	Number of LL control packets that have been received with CRC OK and not ignored
9	nRxCtrlAck	R/W	Number of LL control packets that have been received with CRC OK and not ignored, and then ACKed
10	nRxNok	R/W	Number of packets that have been received with CRC error
11	nRxIgnored	R/W	Number of packets that have been received with CRC OK and ignored due to repeated sequence number
12	nRxEmpty	R/W	Number of packets that have been received with CRC OK and no payload
13	nRxBufFull	R/W	Number of packets that have been received and discarded due to lack of buffer space
14	lastRssi	R	RSSI of last received packet (signed)
15	pktStatus	R/W	Status of received packets (see Table 26-120)
16–19	timeStamp	R	Slave operation: Timestamp of first received packet

Table 26-108. Advertiser Commands

Byte Index	Field Name	Type	Description
0–1	nTxAdvInd	R/W	Number of ADV*_IND packets completely transmitted
2	nTxScanRsp	R/W	Number of AUX_SCAN_RSP or SCAN_RSP packets transmitted

Table 26-108. Advertiser Commands (continued)

Byte Index	Field Name	Type	Description
3	nRxScanReq	R/W	Number of AUX_SCAN_REQ or SCAN_REQ packets received OK and not ignored
4	nRxConnectReq	R/W	Number of AUX_CONNECT_REQ or CONNECT_IND (CONNECT_REQ) packets received OK and not ignored
5	nTxConnectRsp	R/W	Number of AUX_CONNECT_RSP packets transmitted
6–7	nRxNok	R/W	Number of packets received with CRC error
8–9	nRxIgnored	R/W	Number of packets received with CRC OK, but ignored
10	nRxBufFull	R/W	Number of packets received that did not fit in RX queue
11	lastRssi	R	The RSSI of the last received packet (signed)
12–15	timeStamp	R	Timestamp of the last received packet

Table 26-109. Legacy Scanner Command

Byte Index	Field Name	Type	Description
0–1	nTxScanReq	R/W	Number of transmitted SCAN_REQ packets
2–3	nBackedOffScanReq	R/W	Number of SCAN_REQ packets not sent due to backoff procedure
4–5	nRxAdvOk	R/W	Number of ADV*_IND packets received with CRC OK and not ignored
6–7	nRxAdvIgnored	R/W	Number of ADV*_IND packets received with CRC OK, but ignored
8–9	nRxAdvNok	R/W	Number of ADV*_IND packets received with CRC error
10–11	nRxScanRspOk	R/W	Number of SCAN_RSP packets received with CRC OK and not ignored
12–13	nRxScanRspIgnored	R/W	Number of SCAN_RSP packets received with CRC OK, but ignored
14–15	nRxScanRspNok	R/W	Number of SCAN_RSP packets received with CRC error
16	nRxAdvBufFull	R/W	Number of ADV*_IND packets received that did not fit in RX queue
17	nRxScanRspBufFull	R/W	Number of SCAN_RSP packets received that did not fit in RX queue
18	lastRssi	R	The RSSI of the last received packet (signed)
19			Reserved
20–23	timeStamp	R	Timestamp of the last successfully received ADV*_IND packet that was not ignored

Table 26-110. Legacy Initiator Command

Byte Index	Field Name	Type	Description
0	nTxConnectReq	R/W	Number of transmitted CONNECT_IND (CONNECT_REQ) packets
1	nRxAdvOk	R/W	Number of ADV*_IND packets received with CRC OK and not ignored
2–3	nRxAdvIgnored	R/W	Number of ADV*_IND packets received with CRC OK, but ignored
4–5	nRxAdvNok	R/W	Number of ADV*_IND packets received with CRC error
6	nRxAdvBufFull	R/W	Number of ADV*_IND packets received that did not fit in RX queue
7	lastRssi	R	The RSSI of the last received packet (signed)
8–11	timeStamp	R	Timestamp of the received ADV*_IND packet that caused transmission of CONNECT_IND (CONNECT_REQ)

Table 26-111. Bluetooth® Low Energy 5 Scanner and Initiator Command

Byte Index	Field Name	Type	Description
0–1	nTxReq	R/W	Number of transmitted AUX_SCAN_REQ, SCAN_REQ, AUX_CONNECT_REQ, or CONNECT_IND packets
2–3	nBackedOffReq	R/W	Number of AUX_SCAN_REQ, SCAN_REQ, or AUX_CONNECT_REQ packets not sent due to backoff procedure
4–5	nRxAdvOk	R/W	Number of ADV*_IND packets received with CRC OK and not ignored
6–7	nRxAdvIgnored	R/W	Number of ADV*_IND packets received with CRC OK, but ignored

Table 26-111. Bluetooth® Low Energy 5 Scanner and Initiator Command (continued)

Byte Index	Field Name	Type	Description
8–9	nRxAdvNok	R/W	Number of ADV*_IND packets received with CRC error
10–11	nRxRspOk	R/W	Number of AUX_SCAN_RSP, SCAN_RSP, or AUX_CONNECT_RSP packets received with CRC OK and not ignored
12–13	nRxRspIgnored	R/W	Number of AUX_SCAN_RSP, SCAN_RSP, or AUX_CONNECT_RSP packets received with CRC OK, but ignored
14–15	nRxRspNok	R/W	Number of AUX_SCAN_RSP, SCAN_RSP, or AUX_CONNECT_RSP packets received with CRC error
16	nRxAdvBufFull	R/W	Number of ADV*_IND packets received that did not fit in RX queue
17	nRxRspBufFull	R/W	Number of AUX_SCAN_RSP, SCAN_RSP, or AUX_CONNECT_RSP packets received that did not fit in RX queue
18	lastRssi	R	The RSSI of the last received packet (signed)
19			Reserved
20–23	timeStamp	R	Timestamp of the last successfully received *ADV*_IND packet that was not ignored

Table 26-112. Generic RX Command

Byte Index	Field Name	Type	Description
0–1	nRxOk	R/W	Number of packets received with CRC OK
2–3	nRxNok	R/W	Number of packets received with CRC error
4–5	nRxBufFull	R/W	Number of packets that have been received and discarded due to lack of buffer space
6	lastRssi	R	The RSSI of the last received packet (signed)
7			Reserved
8–11	timeStamp	R	Timestamp of the last received packet

Table 26-113. Test TX Command

Byte Index	Field Name	Type	Description
0–1	nTx	R/W	Number of packets transmitted

26.6.1.4 Other Structures and Bit Fields

Table 26-114. Receive Queue Entry Configuration Bit Field(1)

Bit Index	Bit Field Name	Type	Description
0	bAutoFlushIgnored	W	If 1, automatically remove ignored packets from RX queue.
1	bAutoFlushCrcErr	W	If 1, automatically remove packets with CRC error from RX queue.
2	bAutoFlushEmpty	W	If 1, automatically remove empty packets from RX queue.
3	bIncludeLenByte	W	If 1, include the received length byte in the stored packet; otherwise discard it.
4	bIncludeCrc	W	If 1, include the received CRC field in the stored packet; otherwise discard it.
5	bAppendRssi	W	If 1, append an RSSI byte to the packet in the RX queue.
6	bAppendStatus	W	If 1, append a status word to the packet in the RX queue.
7	bAppendTimestamp	W	If 1, append a timestamp to the packet in the RX queue.

(1) This bit field is used for the rxConfig byte of the parameter structures.

Table 26-115. Sequence Number Status Bit Field(1)

Bit Index	Bit Field Name	Type	Description
0	lastRxSn	R/W	The SN bit of the header of the last packet received with CRC OK
1	lastTxSn	R/W	The SN bit of the header of the last transmitted packet
2	nextTxSn	R/W	The SN bit of the header of the next packet to transmit
3	bFirstPkt	R/W	For slave: 0 if a packet is transmitted on the connection; 1 otherwise
4	bAutoEmpty	R/W	1 if the last transmitted packet was an auto-empty packet
5	bLlCtrlTx	R/W	1 if the last transmitted packet was an LL control packet (LLID = 11)
6	bLlCtrlAckRx	R/W	1 if the last received packet was the ACK of an LL control packet
7	bLlCtrlAckPending	R/W	1 if the last successfully received packet was an LL control packet, which has not yet been ACKed

(1) This bit field is used for the seqStat byte of the master and slave parameter structures.

The whitelist structure has the form of an array. Each element consists of 8 bytes, as depicted in Table 26-116. The first byte of the first element tells the number of entries, while this byte is reserved in the remaining entries. The second byte contains some configuration bits, and the remaining 6 bytes contain the address.

Table 26-116. Whitelist Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0	size			W	Number of while list entries. Used in the first entry of the list only.
1	conf	0	bEnable	W	1 if the entry is in use, 0 if the entry is not in use
		1	addrType	W	The type address in the entry public (0) or random (1)
		2	bWlIgn	R/W	1 if the entry is to be ignored by a scanner if the AdvDataInfo field is not present, 0 otherwise. Used to mask out entries that have already been scanned and reported.
		3			Reserved
		4	bIrkValid	R/W	1 if a valid IRK exists, so that the entry is to be ignored by an initiator; 0 otherwise
		5–7			Reserved
2–3	address			W	Least significant 16 bits of the address contained in the entry
4–7	addressHi			W	Most significant 32 bits of the address contained in the entry

The ADI list has the form of an array that consists of 16 entries of the type given in Table 26-117.

Table 26-117. Advertising Data ID Entry Structure

Bit Index	Bit Field Name	Type	Description
0–11	advDataId	R/W	If mode = 1: Last Advertising Data ID (DID) for the Advertising Set ID (SID) corresponding to the entry number in the array
12–13	mode	R/W	0: Entry is invalid (always receive packet with the given SID) 1: Entry is valid (ignore packets with the given SID where DID equals advDataId) 2: Entry is blocked (always ignore packet with the given SID) 3: Reserved
14–15			Reserved

Table 26-118. Receive Status Byte Bit Field for Legacy Commands(1)

Bit Index	Bit Field Name	Type	Description
0–5	channel	R	The channel on which the packet was received, provided channel is in the range 0–39; otherwise 0x3F
6	bIgnore	R	1 if the packet is marked as ignored; 0 otherwise
7	bCrcErr	R	1 if the packet was received with CRC error; 0 otherwise

(1) A byte of this bit field is appended to the received entries of legacy commands if configured.

Table 26-119. Receive Status Word Bit Field for Bluetooth® Low Energy 5 Commands(1)

Bit Index	Bit Field Name	Type	Description
0–5	Channel	R	The channel on which the packet was received, provided channel is in the range 0–39; otherwise 0x3F
6	bIgnore	R	1 if the packet is marked as ignored; 0 otherwise
7	bCrcErr	R	1 if the packet was received with CRC error; 0 otherwise
8–9	phyMode	R	The PHY on which the packet was received 0: 1 Mbps 1: 2 Mbps 2: Coded, S = 8 (125 kbps) 3: Coded, S = 2 (500 kbps)
10–15			Reserved

(1) A 16-bit word of this bit field is appended to the received entries of Bluetooth® Low Energy 5 commands if configured.

The master and slave output structure field pktStatus has the format listed in Table 26-120. The bTimeStampValid bit is set to 0 by the radio CPU at the start of the operation and to 1 if a timestamp is written to the output structure (this happens for slave operation only). The bLastCrcErr bit is set according to the CRC result when a packet is fully received; if no packet is received, then this bit remains unaffected. The remaining bits are set when a packet is received with CRC OK; if no packet is correctly received, then these bits remain unaffected.

Table 26-120. Master and Slave Packet Status Byte

Bit Index	Bit Field Name	Type	Description
0	bTimeStampValid	R/W	1 if a valid timestamp is written to timeStamp; 0 otherwise
1	bLastCrcErr	R/W	1 if the last received packet had CRC error; 0 otherwise
2	bLastIgnored	R/W	1 if the last received packet with CRC OK was ignored; 0 otherwise
3	bLastEmpty	R/W	1 if the last received packet with CRC OK was empty; 0 otherwise
4	bLastCtrl	R/W	1 if the last received packet with CRC OK was an LL control packet; 0 otherwise
5	bLastMd	R/W	1 if the last received packet with CRC OK had MD = 1; 0 otherwise
6	bLastAck	R/W	1 if the last received packet with CRC OK was an ACK of a transmitted packet; 0 otherwise
7			Reserved

Table 26-121. Common Extended Packet Entry Format(1)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0	extHdrInfo	0–5	length	W	Extended header length
		6–7	advMode	W	Advertiser mode as defined in Bluetooth® Low Energy: 0: Nonconnectable, nonscannable 1: Connectable, nonscannable 2: Nonconnectable, scannable 3: Reserved
1	extHdrFlags			W	Extended header flags as defined in Bluetooth® Low Energy
2	extHdrConfig	0	bSkipAdvA	W	0: AdvA is present in extended payload if configured in extHdrFlags. 1: AdvA is inserted automatically from command structure if configured in extHdrFlags and is omitted from extended header.
		1	bSkipTargetA	W	0: TargetA is present in extended payload if configured in extHdrFlags. For response messages, the value is replaced by the received address when sending. 1: TargetA is inserted automatically from command structure or received address if configured in extHdrFlags and is omitted from extended header.
		2	deviceAddrType	W	If bSkipAdvA = 0: The type of the device address in extended header buffer public (0) or random (1)
		3	targetAddrType	W	If bSkipAdvA = 0: The type of the target address in extended header buffer public (0) or random (1)
		4–7			Reserved
3	advDataLen			W	Size of payload buffer
4–7	pExtHeader			W	Pointer to buffer containing extended header. If no fields except extended header flags, automatic advertiser address, or automatic target address are present, pointer may be NULL.
8–11	pAdvData			W	Pointer to buffer containing advData. If advDataLen = 0, pointer may be NULL.

(1) This structure is used for the packet format of transmitted packets of the common extended advertising payload format (see the Specification of the Bluetooth System, Version 5.0 listed in Related Documentation), used with CMD_BLE5_ADV_EXT and CMD_BLE5_ADV_AUX.

26.6.2 Interrupts

The radio CPU uses firmware-defined interrupts to signal events back to the system CPU. Table 26-122 lists the interrupts used by the Bluetooth® Low Energy commands. Each interrupt may be enabled individually in the system CPU. Section 26.8 describes the details about when the interrupts are generated.

Table 26-122. Interrupt Definitions Applicable to Bluetooth® Low Energy

Interrupt Number	Interrupt Name	Description
0	Command_Done	A radio operation command has finished
1	Last_Command_Done	The last radio operation command in a chain of commands has finished
4	Tx_Done	A packet is transmitted
5	Tx_Ack	Acknowledgment received on a transmitted packet
6	Tx_Ctrl	Transmitted LL control packet
7	Tx_Ctrl_Ack	Acknowledgment received on a transmitted LL control packet
8	Tx_Ctrl_Ack_Ack	Acknowledgment received on a transmitted LL control packet, and acknowledgment transmitted for that packet
9	Tx_Retrans	Packet retransmitted

Table 26-122. Interrupt Definitions Applicable to Bluetooth® Low Energy (continued)

Interrupt Number	Interrupt Name	Description
10	Tx_Entry_Done	TX queue data entry state changed to Finished
11	Tx_Buffer_Changed	A buffer change is complete after CMD_BLE_ADV_PAYLOAD
16	Rx_Ok	Packet received with CRC OK, payload, and not to be ignored
17	Rx_Nok	Packet received with CRC error
18	Rx_Ignored	Packet received with CRC OK, but to be ignored
19	Rx_Empty	Packet received with CRC OK, not to be ignored, no payload
20	Rx_Ctrl	LL control packet received with CRC OK, not to be ignored
21	Rx_Ctrl_Ack	LL control packet received with CRC OK, not to be ignored, then acknowledgment sent
22	Rx_Buf_Full	Packet received that did not fit in the RX queue
23	Rx_Entry_Done	RX queue data entry changing state to FINISHED
29	Modules_Unlocked	As part of the boot process, the Arm Cortex-M0 processor has opened access to RF core modules and memories
30	Boot_Done	The RF core CPU boot is finished
31	Internal_Error	The radio CPU has observed an unexpected error

26.7 Data Handling

For all the Bluetooth® Low Energy commands, data received over the air is stored in a receive queue. Data to be transmitted is fetched from a transmit queue for master and slave operation, while for the nonconnected operations, the data is fetched from a specific buffer, or created entirely by the radio CPU based on other available information.

26.7.1 Receive Buffers

A packet being received is stored in a receive buffer. First, a length byte or word is stored if configured in the RX entry by config.lenSz, as explained in Section 26.3.2.7.2. This is calculated from the length received over the air and the configuration of appended information.

Following the optional length field, the received header and payload are stored as received over the air. If rxConfig.bIncludeLenByte is 1, the full 16-bit header (including the received length field) is stored, despite the length field being redundant information if a length byte or word is present. If rxConfig.bIncludeLenByte is 0, only the first byte of the header is stored so that the second byte, which only contains the redundant length field, and (depending on the Bluetooth® version) some RFU bits are discarded.

If rxConfig.bIncludeCrc is 1, the received CRC value is stored in the RX buffer. If rxConfig.bAppendRssi is 1, a signed byte indicating the received RSSI value is appended. If rxConfig.bAppendStatus is 1, a status word is appended. For the legacy Bluetooth® Low Energy commands, this is a 1-byte field as defined in Table 26-118, while for the Bluetooth® Low Energy 5 commands, this is a 2-byte field as defined in Table 26-119. If rxConfig.bAppendTimeStamp is 1, a timestamp indicating the start of the packet is appended. This timestamp corresponds to the ratmr_t data type, which is a 32-bit value in little-endian format. No padding shall be done to ensure a certain alignment in the multibyte fields (timestamp and Bluetooth® Low Energy 5 status). Therefore, the multibyte fields must be written and read bytewise.

Figure 26-8 shows the format of an entry element in the RX queue.

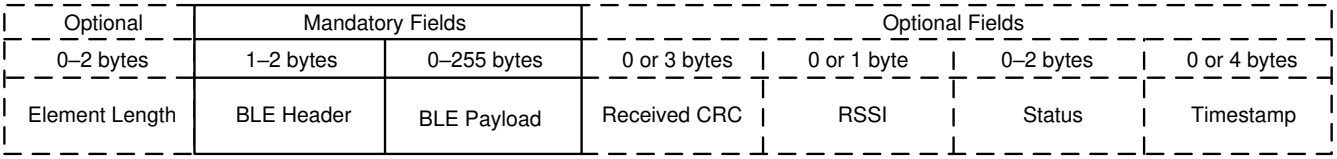


Figure 26-8. Receive Buffer Entry Element

26.7.2 Transmit Buffers

For master and slave operations, transmit buffers are set up in a buffer queue. The length of the packet is defined by the length field in the data entry. The first byte of the data entry gives the LLID that goes into the data channel packet header. The NESN, SN, and MD bits shall be inserted automatically by the radio CPU, the RFU bits shall be set to 0, and the length field shall be calculated from the length of the data entry.

For legacy advertising channel packets and packets transmitted by the Bluetooth® Low Energy scanner and initiator, the radio CPU shall automatically generate the header and the address fields of the payload. The data that comes after the address fields for each message type is given by a pointer to a data buffer. The number of bytes in this buffer is given in a separate parameter. If no data bytes are to be transmitted, this can be indicated by setting the length to 0. In this case, the pointer shall be ignored, and may be set to NULL. For BBluetooth® Low Energy compliance, the ADV_DIRECT_IND and SCAN_REQ messages shall have no payload (but for the possibility of overriding this, data buffers are still present in the legacy commands). With the Bluetooth® Low Energy 5 scanner command, SCAN_REQ and AUX_SCAN_REQ messages are always sent without payload, as per the Bluetooth® specification. For CONNECT_IND and AUX_CONNECT_REQ messages, the data is required to have length 22 for Bluetooth® Low Energy compliance, but the implementation shall allow any length.

For the Bluetooth® Low Energy 5 advertising commands, the transmitted packets use the common extended advertising format. Table 26-121 lists the structure used for describing such packets. The PDU header and parts of the extended header shall be automatically generated by the radio CPU based on this structure. The transmitted packets, the PDU header, or the extended header has pointers to a buffer containing the rest of the extended header and another buffer containing the payload.

26.8 Radio Operation Command Descriptions

26.8.1 Bluetooth® 5 Radio Setup Command

When running Bluetooth® 5, the radio should be set up using the CMD_BLE5_RADIO_SETUP command. This command will set up the radio in a similar way as the CMD_RADIO_SETUP command, but with the possibility to switch between the different supported PHYs from command to command.

The parameter *defaultPhy* gives the PHY to use if the command is used with a non-Bluetooth® Low Energy RX or TX command. For the Bluetooth® Low Energy commands, the defaultPhy parameter is ignored. The Bluetooth® Low Energy 5 commands have a selection of PHY in the command, and the legacy Bluetooth® Low Energy commands will always use the
1 Mbps PHY if the radio is configured using the CMD_BLE5_RADIO_SETUP command.

The parameters *config* and *txPower* have the same meaning as for the CMD_RADIO_SETUP command.

The command has four pointers to override lists. These override lists are processed the same way as for the CMD_RADIO_SETUP command. The override list pRegOverrideCommon is processed only when the setup command is run, and should contain any override that applies to all the PHYs. The override lists pRegOverride1Mbps, pRegOverrideCoded, and pRegOverride2Mbps are processed when switching to a given PHY. The radio CPU will store these override pointers and read the override list at each switch, so these override lists must remain available in memory. Any of the override pointers may be NULL if no overrides are needed.

For running legacy Bluetooth® Low Energy mode with 1 Mbps support only, CMD_RADIO_SETUP with mode 0 may be used as on the CC26x0 devices.

26.8.2 Radio Operation Commands for Bluetooth® Low Energy Packet Transfer

Before running any radio operation command described in this document (with exception to the CMD_BLE5_RADIO_SETUP command), the radio must be set up in Bluetooth® Low Energy mode using the CMD_BLE5_RADIO_SETUP command or the CMD_RADIO_SETUP command. Otherwise, the operation will end with error. When running any of the Bluetooth® Low Energy 5 commands, the CMD_BLE5_RADIO_SETUP command must be used.

The operations start with a radio operation command from the system CPU. The actual start of the operation is set up by the radio CPU according to startTrigger and startTime in the command structure. At this time, the radio

CPU starts configuring the transmitter or receiver, depending on the type of operation. The system CPU must take the setup time of the transmitter or receiver into account when calculating the start time of the operation.

The radio CPU sets up the channel based on the channel parameter. If the channel is in the range 0–39, it indicates a data channel index or advertising channel index. In this case, only the values 0–36 are allowed in master and slave commands, and only the values 37–39 are allowed in advertiser, scanner, and initiator commands. If the channel is in the range 60–207, it indicates an RF frequency with an offset of 2300 MHz. If the channel is 255, the radio CPU does not program any frequency word, but keeps the frequency already programmed with CMD_FS. If the channel is 255 and the frequency synthesizer is not running, the operation ends with an error.

The whitening parameter indicates the initialization of the 7-bit LFSR used for data whitening in Bluetooth® Low Energy. If whitening.bOverride is 0 and the channel is in the range 0–39, the LFSR initializes with (0x40 | channel). Otherwise, the LFSR initializes with whitening.init. If whitening.init is 0 in this case, no whitening is used.

All packets transmitted using the Bluetooth® Low Energy radio operation commands shall have a Bluetooth® Low Energy compliant CRC appended. On all packets received using the Bluetooth® Low Energy radio operation commands, a Bluetooth® Low Energy compliant CRC check shall be performed. The initialization of the CRC register is defined for each command.

The Bluetooth® Low Energy 5 commands have some additional parameters. The phyMode parameter is used to select which of the PHYs to use for the command. For the coded PHY, phyMode.coding gives a rule for selecting the coding used for each packet (see Section 26.8.3 for details). The txPower parameter can be used to select a specific TX power for use in this command, which overrides the one set in the radio setup command and the one set with the CMD_SET_TX_POWER command. If txPower is set to 0x0000, the TX power from the setup command or last CMD_SET_TX_POWER command is used. The rangeDelay parameter gives an extra listening time used for a receiver after T_IFS. The number of RAT ticks given is added to the end of the listening window.

To fulfill the requirements for T_IFS, transmissions following receptions is timed and synchronized by the radio CPU. For reception immediately following transmissions, the radio CPU times the start of RX and timeout so that it always receives a packet transmitted at a time within the limits set by the Bluetooth® Low Energy standard, but without excessive margins, to avoid false syncing on advertising channels. For the first receive operation in a slave command, the radio CPU sets up a timeout as defined in pParams->timeoutTrigger and pParams->timeoutTime. The time of this trigger depends on the sleep-clock uncertainty, both in the slave and the peer master.

When the receiver is running, the message is received into an RX entry, as described in Section 26.3.2.7.2. The radio CPU shall have flags bCrcErr and bIgnore, which are to be written to the corresponding fields of the status byte of the RX entry, if present. If there is a CRC error on the received packet, the bCrcErr flag shall be set. If the CRC is OK, the bIgnore flag may be set based on principles defined for each role. This flag means that the system CPU may ignore the packet. After receiving a packet, the radio CPU shall raise an interrupt to the system CPU.

If a packet is received with a length field that is greater than the maximum length defined in the following, the reception is stopped, and this is treated as if sync had not been obtained on the packet. When the radio is set up using the CMD_RADIO_SETUP command, the length field in data channel packets is configured to have 5 bits and the length field in advertising channel packets is configured to have 6 bits, which corresponds to the specification in Bluetooth® 4.0 and Bluetooth® 4.1. In Bluetooth® 4.2, the data channel packets have an 8-bit length filed, which may be configured with an override, see Section 26.8.4. When the radio is set up using the CMD_BLE5_RADIO_SETUP command, all length fields are configured to have 8 bits, which corresponds to the specification in Bluetooth® 5.0.

By default, the maximum allowed payload length of legacy advertising channel packets is 37, and the maximum allowed payload length of extended advertising channel packets is 255. For legacy master and slave commands, the default maximum allowed length of received data channel packets is 31 (which will never be violated with the default setting using the CMD_RADIO_SETUP command because the length field in this case is 5 bits). For Bluetooth® Low Energy 5 master and slave commands, the maximum packet length of received packets is set in the command structure pParams->maxRxPktLen.

If either the bCrcErr or bIgnore flag is set or if the packet was empty (as defined under each operation), the packet may be removed from the RX entry prior to raising the interrupt, depending on the bAutoFlushIgnored, bAutoFlushCrc, and bAutoFlushEmpty bits of pParams->rxConfig.

The status field of the command issued shall be updated during the operation. When submitting the command, the system CPU shall write this field with a state of IDLE. During the operation, the radio CPU shall update the field to indicate the operation mode. When the operation is done, the radio CPU shall write a status indicating that the operation is finished. Table 26-123 lists the status codes to be used by a Bluetooth® Low Energy radio operation.

Table 26-123. Bluetooth® Low Energy Radio Operation Status Codes

Number	Name	Description
Operation Not Finished		
0x0000	IDLE	Operation not started
0x0001	PENDING	Waiting for start trigger
0x0002	ACTIVE	Running operation
Operation Finished Normally		
0x1400	BLE_DONE_OK	Operation ended normally
0x1401	BLE_DONE_RXTIMEOUT	Timeout of first RX of slave operation or end of scan window
0x1402	BLE_DONE_NOSYNC	Timeout of subsequent RX
0x1403	BLE_DONE_RXERR	Operation ended because of receive error (CRC or other)
0x1404	BLE_DONE_CONNECT	CONNECT_IND or AUX_CONNECT_RSP received or transmitted
0x1405	BLE_DONE_MAXNACK	Maximum number of retransmissions exceeded
0x1406	BLE_DONE_ENDED	Operation stopped after end trigger
0x1407	BLE_DONE_ABORT	Operation aborted by abort command
0x1408	BLE_DONE_STOPPED	Operation stopped after stop command
0x1409	BLE_DONE_AUX	Operation ended after receiving AuxPtr pointing a long time ahead
0x140A	BLE_DONE_CONNECT_CHSEL0	CONNECT_IND received or transmitted; peer does not support channel selection algorithm number 2
Operation Finished with Error		
0x1800	BLE_ERROR_PAR	Illegal parameter
0x1801	BLE_ERROR_RXBUF	No available RX buffer (Advertiser, Scanner, Initiator)
0x1802	BLE_ERROR_NO_SETUP	Radio was not set up in Bluetooth® Low Energy mode
0x1803	BLE_ERROR_NO_FS	Synthesizer was not programmed when running RX or TX
0x1804	BLE_ERROR_SYNTH_PROG	Synthesizer programming failed
0x1805	BLE_ERROR_RXOVF	RX overflow observed during operation
0x1806	BLE_ERROR_TXUNF	TX underflow observed during operation
0x1807	BLE_ERROR_AUX	AUX pointer target is too far into the future

The conditions for giving each status are listed for each operation. Some of the error causes listed in Table 26-123 are not repeated in these lists. In some cases, general error causes may occur. In all of these cases, the result of the operation is ABORT.

26.8.3 Coding Selection for Coded PHY

The phyMode.coding parameter of the Bluetooth® 5 commands is used when the coded PHY is selected, which determines the coding rate of transmitted packets. The phyMode.coding parameter has a different meaning for link layer connection commands (master and slave) and other commands.

For master and slave commands, the pParams->maxLenLowRate parameter is also used for determining the coding rate. If the length of the transmitted packet is greater than pParams->maxLenLowRate, the packet is

transmitted with S = 2 (500 kbps) regardless of phyMode.coding. This feature can be used to ensure that the maximum packet duration is not violated. In other cases, the phyMode.coding parameter is used, and the meaning of the bits are described in Table 26-124.

Table 26-124. Coding Selection for Master and Slave Commands

Bit Number	Description
0	0: Default to S = 8 (125 kbps) 1: Default to S = 2 (500 kbps)
1	0: Do not modify default rate based on last received packet 1: Always use S = 8 (125 kbps) if the last received packet in the same connection event was coded with S = 8 (125 kbps)
2	0: Use default rate for empty packets 1: Always use S = 8 (125 kbps) for empty packets
3	0: Use default rate for retransmissions 1: Always use S = 8 (125 kbps) for retransmissions
4–5	Reserved

For the other Bluetooth® 5 commands, the meaning of phyMode.coding is given in Table 26-125. For the CMD_BLE5_ADV_EXT and CMD_BLE5_TX_TEST commands, bits 1 and 2 are not applicable. For the CMD_BLE5_SCANNER and CMD_BLE5_INITIATOR commands, bit 0 is not applicable. For the CMD_BLE5_ADV_AUX command, all bit 0, bit 1, and bit 2 are applicable.

Table 26-125. Coding Selection for Advertiser, Scanner, and Initiator Commands

Bit Number	Description
0	Code rate to use for advertising indications (ADV_EXT_IND, AUX_ADV_IND, AUX_CHAIN_IND) and TX test packets 0: Use S = 8 (125 kbps) 1: Use S = 2 (500 kbps)
1	Default code rate to use for request and response packets (AUX_SCAN_REQ, AUX_CONNECT_REQ, AUX_SCAN_RSP, AUX_CONNECT_RSP) 0: Default to S = 8 (125 kbps) 1: Default to S = 2 (500 kbps)
2	0: Use default rate for request and response packets 1: Always use S = 8 (125 kbps) for a request or response packet if the last received packet was coded with S = 8 (125 kbps)
3–5	Reserved

26.8.4 Parameter Override

Several parameters that are not configurable through the parameter structures can still be overridden by using the CMD_BLE5_RADIO_SETUP, CMD_RADIO_SETUP, CMD_UPDATE_RADIO_SETUP, or CMD_WRITE_FWPAR commands.

The parameters that can be overridden in Bluetooth® Low Energy mode are described in the Bluetooth® Specification documents listed in Related Documentation. The parameters can, for example, be used to improve performance or to create noncompliant behavior that may be useful in some cases. Examples follow:

- Modification of T_IFS and other timing
- Modification of maximum RX packet length (needed for the LE Data Packet Length Extension feature in Bluetooth® 4.2)
- Modification of automatically generated headers and so forth
- Modifications to CRC, to generate bit errors for testing and to use different CRC

26.8.5 Link Layer Connection

At the start of a slave or master operation, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. The channel parameter is not allowed to be 37, 38, or 39 because they are not data channels. For the Bluetooth® Low Energy 5 commands, it shall

also set up the PHY mode given in phyMode.mainMode. The radio CPU shall set up the access address defined in pParams->accessAddress and shall use the CRC initialization value defined in pParams->crcInit. The whitener shall be set up as defined in the whitening parameter. The radio CPU shall then configure the receiver or the transmitter. The operation proceeds with reception and transmission in turn until it ends by one of the end-of-command criteria.

When the demodulator obtains sync on a message, the message is received into the first available RX buffer that can fit the packet. The flags *bCrcErr* and *bIgnore* are set according to Table 26-126, depending on the CRC result and whether the SN field of the header was the same as the SN field of the last successfully received packet. A received packet that has a payload length of 0 shall be viewed as an empty packet, which means that if pParams->rxConfig.bAutoflushEmpty is 1 and bCrcErr and bIgnore are both 0, the packet is removed from the RX buffer.

Table 26-126. Actions on Received Packets

CRC Result	SN Different from Previous	bCrcErr	bIgnore
OK	Yes	0	0
OK	No	0	1
NOK	X	1	0

If there is no available RX buffer with enough available space to hold the received packet, the received data shall be discarded. However, the packet shall be received so that the CRC can be checked. When the packet is received, the radio CPU shall set the sequence bits so that a retransmission of the lost packet is requested (that is, NACK), unless the packet would have been discarded from the RX queue anyway due to the setting of pParams->rxConfig.

If two subsequent packets are received with CRC error, the command shall end, as required by the Bluetooth® Low Energy specification.

When a packet is to be transmitted or retransmitted, it is read from the current data entry in the TX queue unless the TX queue is empty or an auto-empty packet has to be retransmitted. The radio CPU shall create the header as follows.

- The LLID bits shall be inserted from the first byte of the TX data entry.
- The SN and NESN bits will be set to values according to the Bluetooth® Low Energy protocol (this is discussed in the following text)
- The MD bit shall be calculated automatically as discussed in the following text, unless this is overridden (see Section 26.8.4).
- If the TX queue is empty, an empty packet (LLID = 0x1, Length = 0) is transmitted. Such a packet is referred to as an auto-empty packet.

There are a number of interrupts that can be raised on different conditions. The pOutput structure contains a number of counters corresponding to the interrupts. Table 26-127 lists the conditions for incrementing each counter or raising an interrupt. There may be more than one condition fulfilled after a packet is transmitted or received. In the list of conditions, the term *acknowledgment* is used to define a successfully received packet with an NESN value in the header that is different from the SN value of the last transmitted packet.

Table 26-127. Conditions for Incrementing Counters and Raising Interrupts for Master and Slave Commands

Condition	Counter Incremented	Interrupt Generated
Packet transmitted	nTx	Tx_Done
Packet transmitted and acknowledgment received	nTxAck	Tx_Ack
Packet with LLID = 11b transmitted	nTxCtrl	Tx_Ctrl
Packet with LLID = 11b transmitted and acknowledgment received	nTxCtrlAck	Tx_Ctrl_Ack
Packet with LLID = 11b transmitted, acknowledgment received, and acknowledgment sent	nTxCtrlAckAck	Tx_Ctrl_Ack_Ack
Packet transmitted with same SN as previous transmitted packet	nTxRetrans	Tx_Retrans
Packet with payload transmitted and acknowledgment received	nTxEntryDone	Tx_Entry_Done
Packet received with bCrcErr = 0, bIgnore = 0, and payload length > 0	nRxOk	Rx_Ok
Packet received with CRC error (bCrcErr = 1)	nRxNok	Rx_Nok
Packet received with bCrcErr = 0 and bIgnore = 1	nRxIgnored	Rx_Ignored
Packet received with bCrcErr = 0, bIgnore = 0, and payload length = 0	nRxEmpty	Rx_Empty
Packet received with LLID = 11b, bCrcErr = 0 and bIgnore = 0	nRxCtrl	Rx_Ctrl
Packet received with LLID = 11b, bCrcErr = 0 and bIgnore = 0, and acknowledgment sent	nRxCtrlAck	Rx_Ctrl_Ack
Packet received that did not fit in RX buffer and was not to be flushed	nRxBufFull	Rx_Buf_Full
The first RX data entry in the RX queue changed state to finished	—	Rx_Entry_Done

The radio CPU shall maintain two counters: one packet counter *nPkt* and one NACK counter *nNack*. At the start of the master or slave radio operation, both counters shall be initialized to pParams->maxPkt and pParams->maxNack, respectively. The packet counter *nPkt* shall be decremented each time a packet transmitted. The NACK counter *nNack* shall be decremented if a packet is received that does not contain an acknowledgment of the last transmitted packet, otherwise it shall be reset to pParams->maxNack if an acknowledgment is received. If either counter counts to 0, the operation shall end. This shall happen after a packet is received for master and a packet is transmitted for slave. Setting pParams->maxPkt or pParams->maxNack to 0 shall disable the corresponding counter functionality.

A trigger to end the operation is set up by pParams->endTrigger and pParams->endTime. If the trigger that is defined by this parameter occurs, the radio operation shall end as soon as possible. After the radio operation ends, any transmitted packet shall have MD = 0 and the connection event shall end after the next packet is transmitted for a slave or received for a master. If the immediate command *CMD_STOP* is received by the radio CPU, it shall have the same meaning as the end trigger occurring (except that after ending, the status code shall be CMD_DONE_STOPPED). For slave commands, pParams->endTrigger.triggerType may take the value BLE_TRIG_REL_SYNC, which is 15. This value means that pParams->endTime is relative to the timestamp of the first received packet. If no packet is received, the end trigger will not occur (the timeout trigger should be used to handle this situation).

The register *pParams->seqStat* contains bits that shall be updated by the radio CPU during operation and shall be used for getting correct operation on SN, NESN, and retransmissions. The rules that must be followed for the radio CPU are:

- Before the first operation on a connection, the bits in pParams->seqStat shall be set as follows by the system CPU:
  - lastRxSn = 1
  - lastTxSn = 1
  - nextTxSn = 0
  - bFirstPkt = 1
  - bAutoEmpty = 0
  - bLlCtrlRx = 0
  - bLlCtrlAckRx = 0
  - bLlCtrlAckPending = 0
- When determining if the SN field of the header was the same as the SN field of the last successfully received packet, the received SN bit is compared to pParams->seqStat.lastRxSn.
- If a packet is received with correct CRC and the packet fit in an RX buffer, the received SN shall be stored in pParams->seqStat.lastRxSn. If the packet was an LL control packet (LLID = 11b) and the packet was not to be ignored, pParams->seqStat.bLlCtrlAckPending shall be set to 1 and an Rx_Ctrl interrupt shall be raised.
- If a packet is received with correct CRC and the received NESN is different from pParams->seqStat.lastTxSn, pParams->seqStat.nextTxSn shall be set to the value of the received NESN (regardless of whether the packet fit in an RX buffer).
- If pParams->seqStat.bFirstPkt = 0:
  - If pParams->seqStat.nextTxSn was updated and became different from pParams->seqStat.lastTxSn after reception of a packet, nNack shall be set to pParams->maxNack and a Tx_Ack interrupt shall be raised.
  - Otherwise, nNack shall be decremented.
  - If pParams->seqStat.nextTxSn was updated and became different from pParams->seqStat.lastTxSn after reception of a packet and pParams->seqStat.bAutoEmpty = 0, the current TX queue entry shall be finished, the next one shall be set as active (see Section 26.3.2.7.2), and a Tx_Entry_Done interrupt shall be raised. If pParams->seqStat.bLlCtrlTx = 1, a Tx_Ctrl_Ack interrupt shall be raised and pParams->seqStat.bLlCtrlAckRx shall be set to 1.
  - If pParams->seqStat.nextTxSn was updated and became different from pParams->seqStat.lastTxSn after reception of a packet, pParams->seqStat.bAutoEmpty shall be set to 0.
- If no buffer is available in the TX queue, or if pParams->seqStat.nextTxSn is equal to pParams->seqStat.lastTxSn and pParams->seqStat.bAutoEmpty = 1 when transmission of a packet is to take place, an auto-empty packet shall be transmitted. Nothing shall be read from the TX queue. Otherwise, the transmitted packet shall be read from the first entry of the TX queue.

- In the header of a transmitted packet, the SN bit shall be set to the value of pParams->seqStat.nextTxSn, and the NESN bit shall be set to the inverse of pParams->seqStat.lastRxSn.
- After a packet is transmitted:
  - If pParams->seqStat.nextTxSn is equal to pParams->seqStat.lastTxSn, a Tx_Retrans interrupt shall be raised.
  - If pParams->seqStat.nextTxSn is different from pParams->seqStat.lastTxSn after a transmission and the transmitted packet had LLID = 11b, a Tx_Ctrl interrupt shall be raised.
  - If pParams->seqStat.nextTxSn is different from pParams->seqStat.lastTxSn after a transmission and pParams->seqStat.bLlCtrlAckPending = 1, an Rx_Ctrl_Ack interrupt shall be raised.
  - If pParams->seqStat.nextTxSn is different from pParams->seqStat.lastTxSn after a transmission and pParams->seqStat.bLlCtrlAckRx = 1, a Tx_Ctrl_Ack_Ack interrupt shall be raised.
  - pParams->seqStat.lastTxSn shall be set to the value of pParams->seqStat.nextTxSn.
  - pParams->seqStat.bAutoEmpty shall be set to 1 if the packet was not read from the TX queue, otherwise set to 0.
  - pParams->seqStat.bLlCtrlTx shall be set to 1 if the transmitted packet had LLID = 11, otherwise set to 0.
  - pParams->seqStat.firstPkt, pParams->seqStat.bLlCtrlAckPending, and pParams->seqStat.bLlCtrlAckRx shall be set to 0.
  - A Tx_Done interrupt shall be raised.
  - nPkt shall be decremented.

When an interrupt is raised as described in the previous list, the corresponding counter given in Table 26-127 shall be incremented.

In the header of a transmitted packet, the MD bit shall be set according to the following rules:

- If the transmit queue is empty or the packet being transmitted is the last packet of the transmit queue, MD shall be set to 0.
- If the trigger described in pParams->endTrigger has occurred, MD shall be set to 0.
- If the counter nPkt is 1, MD shall be set to 0.
- Otherwise, MD shall be set to 1.

The pOutput structure contains counters, which shall be updated by the radio CPU as explained in the previous list and in Table 26-127. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired. In addition to the counters, the following fields shall be set by the radio CPU:

- If a packet is received, lastRssi is set to the RSSI of that packet.
- For slave commands, timeStamp shall be set to the timestamp of the start of the first received packet (if any packet is received). The bValidTimeStamp field shall be set to 0 at the beginning of the operation and to 1 if a packet is received so that timeStamp is written.

For correct operation, the value of pParams->seqStat shall be the same at the beginning of a command as the value at the end of the previous operation of the same connection. The TX queue should also be unmodified between commands operating on the same connection (except that packets may be appended to the queue).

26.8.6 Slave Command

A slave radio operation is started by a CMD_BLE_SLAVE or CMD_BLE5_SLAVE command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-94 or Table 26-101 and a pOutput parameter of the type defined in Table 26-107. The operation starts with reception. The parameters pParams->timeoutTrigger and pParams->timeoutTime define the time to end the operation if no sync is found by the demodulator. Together, the startTrigger and pParams->timeoutTrigger define the receive window for the slave.

The first received packet of a new LL connection on a slave shall be treated in a special way. This is signaled by the system CPU by setting pParams->seqStat.bFirstPkt to 1 when starting the first slave operation of a new connection. When this flag is set, the received packet shall not be viewed as an ACK or NACK of a packet transmitted previously. When a packet is transmitted, pParams->seqStat.bFirstPkt shall be cleared by the radio CPU.

The radio CPU shall write a timestamp of the first received packet of the radio operation into pOutput->timeStamp. The captured time can be used by the system CPU as an anchor point to calculate the start of future slave commands. This time shall also be defined as *event 1*. This may be used for timing subsequent chained operations. If no anchor point is found, event 1 will be the time of the start of the slave operation.

If a packet is received with CRC error, the radio CPU shall end the radio operation if the previous packet in the same radio operation was also received with CRC error (see Table 26-128). Otherwise if a packet is received, the radio CPU shall start the transmitter and transmit from the TX queue, or transmit an auto-empty packet if the TX queue is empty. The transmission may be a retransmission. Unless the operation is to end by the criteria listed in Table 26-128, the receiver will be started after the transmission is complete.

A slave operation shall end by one of the causes listed in Table 26-128. After the operation has ended, the status field of the command structure indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result is indicated as TRUE, FALSE, or ABORT, which will decide the next action.

Table 26-128. End of Slave Operation

Condition	Status Code	Result
Transmitted packet with MD = 0 after successfully receiving a packet where the MD bit of the header is 0	BLE_DONE_OK	TRUE
Transmitted packet with MD = 0 after receiving a packet that did not fit in the RX queue	BLE_DONE_OK	TRUE
Finished transmitting packet and nPkt counted to 0	BLE_DONE_OK	TRUE
Trigger indicated by pParams->timeoutTrigger occurred before demodulator sync is ever obtained after starting the command	BLE_DONE_RXTIMEOUT	FALSE
No sync obtained on receive operation after transmit	BLE_DONE_NOSYNC	TRUE
Two subsequent packets in the same operation were received with CRC error	BLE_DONE_RXERR	TRUE
Finished transmitting packet after the internal counter nNack has counted down to 0	BLE_DONE_MAXNACK	TRUE
Finished transmitting packet after observing a trigger indicated by pParams->endTrigger	BLE_DONE_ENDED	FALSE
Finished transmitting packet after observing a CMD_STOP	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
Illegal value of channel	BLE_ERROR_PAR	ABORT
TX data entry length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.7 Master Command

A master radio operation is started by a CMD_BLE_MASTER or a CMD_BLE5_MASTER command. In the command structure, it shall have a *pParams* parameter of the type defined in Table 26-95 and a *pOutput* parameter of the type defined in Table 26-107. The operation starts with transmission. After each transmission, the receiver is started.

A master operation shall end by one of the causes listed in Table 26-129. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-129. End of Master Operation

Condition	Status Code	Result
Successfully received packet with MD = 0 after transmitting a packet with MD = 0	BLE_DONE_OK	TRUE
Received packet that did not fit in RX queue after transmitting a packet with MD = 0	BLE_DONE_OK	TRUE
Received a packet after nPkt had counted to 0	BLE_DONE_OK	TRUE
No sync obtained on receive operation after transmit	BLE_DONE_NOSYNC	TRUE
Two subsequent packets in the same operation were received with CRC error	BLE_DONE_RXERR	TRUE
The internal counter nNack counted down to 0 after a packet was received	BLE_DONE_MAXNACK	TRUE
Received a packet after observing a trigger indicated by pParams->endTrigger	BLE_DONE_ENDED	FALSE
Received a packet after observing a CMD_STOP	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
Illegal value of channel	BLE_ERROR_PAR	ABORT
TX data entry length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.8 Legacy Advertiser

At the start of an advertiser operation of any kind, the radio CPU waits for the start trigger, then programs the frequency based on the channel parameter of the command structure. The channel parameter is not allowed to be in the range of 0–36, because these are data channels. The radio CPU sets up the advertising channel access address and uses the CRC initialization value of 0x555555. The whitener is set up as defined in the whitening parameter. The radio CPU then configures the transmitter. Except for an advertiser that is not connectable, the operation goes on with reception after transmission, and if a SCAN_REQ is received, another transmission of a SCAN_RSP may occur.

In Bluetooth® Low Energy mode, advertising is usually done over all three advertising channels. To set this up, three command structures can be chained using the pNextOp parameter. Typically, the parameter and output structures can be the same for all channels.

The first packet transmitted is always an ADV*_IND packet. This packet consists of a header, an advertiser address, and advertising data (except for the ADV_DIRECT_IND packet that is used in directed advertising). The radio CPU constructs these packets as follows (the ADV_DIRECT_IND packet is described in Section 26.8.8.2).

- In the header, the PDU Type bits are as shown in Table 26-130.
- The TXAdd bit is as shown in pParams->advConfig.deviceAddrType.
- The length is calculated from the size of the advertising data, meaning that it is pParams->advLen + 6.
- The RXAdd bit is not used and is 0, along with the RFU bits.
- The payload starts with the 6-byte device address, which are read from pParams->pDeviceAddress.
- The rest of the payload is read from the pParams->pAdvData buffer (if pParams->advLen is nonzero).

Table 26-130. PDU Types for Different Advertiser Commands

Command	Type of Advertising Packet	Value of PDU Type Bits in Header
CMD_BLE_ADV	ADV_IND	0000b
CMD_BLE_ADV_DIR	ADV_DIRECT_IND	0001b
CMD_BLE_ADV_NC	ADV_NONCONN_IND	0010b
CMD_BLE_ADV_SCAN	ADV_SCAN_IND	0110b

Except for the non-connectable advertiser, the receiver shall be started after the ADV*_IND packet is transmitted. Depending on the type of advertiser operation, the receiver shall listen for a SCAN_REQ and (or) a CONNECT_IND (known as *CONNECT_REQ* in Bluetooth® 4.0, 4.1, and 4.2 Specifications listed in Related Documentation). If the demodulator obtains sync, the header shall be checked when it is received, and if it is not a SCAN_REQ or CONNECT_IND message, the demodulator shall be stopped immediately.

A SCAN_REQ or CONNECT_IND message is received into the RX queue given by pParams->pRxQ, as described in Section 26.10.4.1. The bCrcErr and bIgnore bits are set according to the CRC result and the received message. The AdvA field in the message and the TxAdd bit of the received header are compared to the pParams->pDeviceAddress array and pParams->advConfig.deviceAddrType, respectively, to see if the message was addressed to this advertiser. Then, depending on the advertising filter policy that is given by pParams->advConfig.advFilterPolicy, the received ScanA or InitA field, along with the RxAdd bit of the received header, is checked against the whitelist (see Section 26.8.14), except for a directed advertiser, where the received header is compared against the peer address (see Section 26.8.8.2). If the resolvable private address (RPA) mode (given by pParams->advConfig.rpaMode) is nonzero, an extra check is done to see if the peer address is a resolvable private address. If the received TxAdd is 1 and the two most significant bits of the received ScanA or InitA field are 01b, the address is a RPA. If so, a whitelist check is performed regardless of the filter policy. Depending on the received packet, the actions taken shall be as given in Table 26-131, where the definition of each action (including the value that will be used on bCrcErr and bIgnore) is given in Table 26-132. If pParams->advConfig.bStrictLenFilter is 1, only length fields that are compliant with the Bluetooth® Low Energy specification shall be considered valid. For a SCAN_REQ, that means a length field of 12, and for a CONNECT_IND it means a length field of 34. If pParams->advConfig.bStrictLenFilter is 0, all received packets with a length field less than or equal to the maximum length of an advertiser packet shall be considered valid. If the length is not valid, the receiver shall be stopped.

Table 26-131. Actions to Take Based on Received Packets for Advertisers

PDU Type	CRC Result	Adv. Type	Valid Length	AdvA Matches Own Address	Filter Policy	RPA Mode	Resolvable Private Address	ScanA or InitA Present in Whitelist	Action No.
SCAN_REQ	OK	C, S	Yes	No	X	X	X	X	1
SCAN_REQ	OK	C, S	Yes	Yes	1 or 3	X	X	No	1
SCAN_REQ	OK	C, S	Yes	Yes	1 or 3	X	X	Yes	2
SCAN_REQ	OK	C, S	Yes	Yes	0 or 2	0	X	X	2
SCAN_REQ	OK	C, S	Yes	Yes	0 or 2	1	No	X	2
SCAN_REQ	OK	C, S	Yes	Yes	0 or 2	1	Yes	No	1
SCAN_REQ	OK	C, S	Yes	Yes	0 or 2	1	Yes	Yes	2
SCAN_REQ	NOK	C, S	Yes	X	X	X	X	X	3
SCAN_REQ	X	C, S	No	X	X	X	X	X	5
SCAN_REQ	X	D	X	X	X	X	X	X	5
CONNECT_IND	OK	C, D	Yes	No	X	X	X	X	1
CONNECT_IND	OK	C, D	Yes	Yes	2 or 3	X	X	No	1
CONNECT_IND	OK	C, D	Yes	Yes	2 or 3	X	X	Yes	4
CONNECT_IND	OK	C, D	Yes	Yes	0 or 1	0	X	X	4
CONNECT_IND	OK	C, D	Yes	Yes	0 or 1	1	No	X	4
CONNECT_IND	OK	C, D	Yes	Yes	0 or 1	1	Yes	No	1
CONNECT_IND	OK	C, D	Yes	Yes	0 or 1	1	Yes	Yes	4
CONNECT_IND	NOK	C, D	Yes	X	X	X	X	X	3
CONNECT_IND	X	C, D	No	X	X	X	X	X	5
CONNECT_IND	X	S	X	X	X	X	X	X	5
Other	X	X	X	N/A	X	X	N/A	N/A	5
No packet received	N/A	X	N/A	N/A	X	X	N/A	N/A	5

Table 26-132. Descriptions of the Actions to Take on Received Packets

Action No.	bCrcErr	bIgnore	Description
1	0	1	End operation with BLE_DONE_OK status
2	0	0	Transmit SCAN_RSP message
3	1	0	End operation with BLE_DONE_RXERR status
4	0	0	End operation with BLE_DONE_CONNECT or BLE_DONE_CONNECT_CHSEL0 status
5	—	—	Stop receiver immediately and end operation with BLE_DONE_NOSYNC status

If a SCAN_REQ packet is received with a length of 12 (or less), it shall be viewed as an empty packet. This means that if pParams->rxConfig.bAutoflushEmpty is 1 and bCrcErr and bIgnore are both 0, the packet is removed from the RX buffer. If a packet is flagged by bIgnore or bCrcErr, it may also be removed based on the bits in pParams->rxConfig.

If the packet being received did not fit in the RX queue, the packet is received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX queue based on the bits in pParams->rxConfig, the command shall end.

If a CONNECT_IND packet is correctly received (see Action 4 in Table 26-132) and pParams->advConfig.chSel is 1, the ChSel bit of the received header is checked. If this bit is 0 (meaning that the peer does not support Channel Selection Algorithm 2), the status is set to BLE_DONE_CONNECT_CHSEL0 instead of to BLE_DONE_CONNECT.

If the next action (according to Table 26-131 and Table 26-132) is to transmit a SCAN_RSP packet, the radio CPU shall start the transmitter to transmit this packet. This packet consists of a header, an advertiser address, and advertising data.

The radio CPU shall construct these packets as follows:

- In the header, the PDU Type bits shall be 0100b.
- The TxAdd bit shall be as in pParams->advConfig.deviceAddrType.
- The length shall be calculated from the size of the scan response data, meaning that it shall be pParams->scanRspLen + 6.
- The RxAdd and ChSel bits are not used and shall be 0.
- The RFU bit shall be 0.
- The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress.
- The rest of the payload shall be read from the pParams->pScanRspData buffer.
- After the SCAN_RSP is transmitted, the command shall end.

A trigger to end the operation is set up by pParams->endTrigger. If the trigger that is defined by this parameter occurs, the radio operation shall continue to completion, but the status code after ending shall be BLE_DONE_ENDED and the result shall be FALSE. This trigger can be used to stop execution instead of proceeding with the next chained operation by use of the condition in the command structure. If the immediate command CMD_STOP is received by the radio CPU, it shall have the same meaning as when the end trigger occurs (except that the status code after ending shall be CMD_DONE_STOPPED).

The output structure pOutput contains fields that give information on the command being run. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired. The fields shall be updated by the radio CPU as described in the following list. The list also indicates when interrupts shall be raised in the system CPU.

- When the ADV*_IND packet is transmitted, nTxAdvInd is incremented and a Tx_Done interrupt is raised.
- If a SCAN_RSP packet is transmitted, nTxScanRsp is incremented afterward, and a Tx_Done interrupt is raised.
- If a SCAN_REQ packet is received with CRC OK and the bIgnore flag cleared, nRxScanReq is incremented. If the payload length is 12 or less, an Rx_Empty interrupt is raised. If the payload length is greater than 12, an Rx_Ok interrupt is raised.
- If a CONNECT_IND packet is received with CRC OK and the bIgnore flag cleared, nRxConnectReq is incremented and an Rx_OK interrupt is raised.
- If a packet is received with a CRC error, nRxNok is incremented and an Rx_Nok interrupt is raised.
- If a packet is received and the bIgnore flag is set, nRxIgnored is incremented and an Rx_Ignored interrupt is raised.
- If a packet is received that did not fit in the RX queue, nRxBufFull is incremented and an Rx_Buf_Full interrupt is raised.
- If a packet is received, lastRssi is set to the RSSI of that packet.
- If a packet is received, timeStamp is set to a timestamp of the start of that packet. For a CONNECT_IND packet, this can be used to calculate the anchor point of the first packet.
- If the first RX data entry in the RX queue changed state to Finished after a packet was received, an Rx_Entry_Done interrupt is raised.

26.8.8.1 Connectable Undirected Advertiser Command

A connectable undirected advertiser operation is started by a CMD_BLE_ADV command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-96 and a pOutput parameter of the type defined in Table 26-108. The operation starts with transmission and operates as described in Section 26.8.8.

A connectable undirected advertiser operation shall end with one of the statuses listed in Table 26-133. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-133. End of Connectable Undirected Advertiser Operation

Condition	Status Code	Result
Performed Action 1 after running receiver	BLE_DONE_OK	TRUE
Performed Action 2 and transmitted SCAN_RSP	BLE_DONE_OK	TRUE
Performed Action 3 after running receiver	BLE_DONE_RXERR	TRUE
Performed Action 4 after running receiver if pParams->advConfig.chSel = 0 or the ChSel bit of the received CONNECT_IND is 1	BLE_DONE_CONNECT	FALSE
Performed Action 4 after running receiver if pParams->advConfig.chSel = 1 and the ChSel bit of the received CONNECT_IND is 0	BLE_DONE_CONNECT_CHSEL0	FALSE
Performed Action 5 after running receiver	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->endTrigger, then performed Actions 1, 2, 3, or 5	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then performed Actions 1, 2, 3, or 5	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT
Advertising data or scan response data length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.8.2 Connectable Directed Advertiser Command

A connectable directed advertiser operation is started by a CMD_BLE_ADV_DIR command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-96 and a pOutput parameter of the type defined in Table 26-108. The operation starts with transmission and operates as described in Section 26.8.8 with some modifications as described in the following list.

For the directed advertiser, pParams->pWhiteList shall point to a buffer containing only the device address of the device to which to connect. The address type of the peer shall be given in pParams->advConfig.peerAddrType.

The first transmit operation will send an ADV_DIRECT_IND packet. The radio CPU shall construct this packet as follows:

- In the header, the PDU Type bits shall be 0001b as shown in Table 26-130.
- The TxAdd bit shall be as in pParams->advConfig.deviceAddrType.
- The RxAdd bit shall be as in pParams->advConfig.peerAddrType (if the least significant bit of pParams->pWhiteList is 1, the RxAdd bit is inverted, see Section 26.8.17).
- The ChSel bit shall be 1 if pParams->advConfig.chSel is 1; otherwise, ChSel shall be 0.
- The length shall be calculated from the size of the advertising data, which means that it shall be pParams->advLen + 12.
- The RFU bit shall be 0.
- The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress, followed by the 6-byte peer address read from pParams->pWhiteList.
- By the Bluetooth® Low Energy specification, there shall be no more payload, but a noncompliant message may be constructed by setting pParams->advLen to a nonzero value. If so, the rest of the payload shall be read from the pParams->pAdvData buffer.

The receiver is started after the ADV_DIRECT_IND packet is transmitted as described in Section 26.8.8, and received packets are processed as described there. Instead of being checked against the whitelist, the received TargetA field and TxAdd bit are checked against pParams->pWhiteList and pParams->advConfig.peerAddrType, respectively.

A directed advertiser operation shall end with one of the statuses listed in Table 26-134. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-134. End of Directed Advertiser Operation

Condition	Status Code	Result
Performed Action 1 after running receiver	BLE_DONE_OK	TRUE
Performed Action 3 after running receiver	BLE_DONE_RXERR	TRUE
Performed Action 4 after running receiver if pParams->advConfig.chSel = 0 or the ChSel bit of the received CONNECT_IND is 1	BLE_DONE_CONNECT	FALSE
Performed Action 4 after running receiver if pParams->advConfig.chSel = 1 and the ChSel bit of the received CONNECT_IND is 0	BLE_DONE_CONNECT_CHSEL0	FALSE
Performed Action 5 after running receiver	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->endTrigger, then performed Actions 1, 3, or 5	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then performed Actions 1, 3, or 5	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT
Advertising data length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.8.3 Non-connectable Advertiser Command

A non-connectable advertiser operation is started by a CMD_BLE_ADV_NC command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-96 and a pOutput parameter of the type defined in Table 26-108. The operation starts with transmission and operates as described in Section 26.8.8. After transmission of an ADV_NONCONN_IND packet, the operation shall end without any receive operation.

A non-connectable advertiser operation shall end with one of the statuses listed in Table 26-135. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-135. End of Non-connectable Advertiser Operation

Condition	Status Code	Result
Transmitted ADV_NONCONN_IND	BLE_DONE_OK	TRUE
Observed trigger indicated by pParams->endTrigger, then finished transmitting ADV_NONCONN_IND	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then finished transmitting ADV_NONCONN_IND	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
Illegal value of channel	BLE_ERROR_PAR	ABORT
Advertising data length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.8.4 Scannable Undirected Advertiser Command

A scannable undirected advertiser operation is started by a CMD_BLE_ADV_SCAN command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-96 and a pOutput parameter of the type defined in Table 26-108. The operation starts with transmission operation and operates as described in Section 26.8.8.

A scannable undirected advertiser operation shall end with one of the statuses listed in Table 26-136. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-136. End of Scannable Undirected Advertiser Operation

Condition	Status Code	Result
Performed Action 1 after running receiver	BLE_DONE_OK	TRUE
Performed Action 2 and transmitted SCAN_RSP	BLE_DONE_OK	TRUE
Performed Action 3 after running receiver	BLE_DONE_RXERR	TRUE
Performed Action 5 after running receiver	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->endTrigger, then performed Actions 1, 2, 3, or 5	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then performed Actions 1, 2, 3, or 5	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT
Advertising data or scan response data length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.9 Bluetooth® 5 Advertiser Commands

At the start of an extended or secondary channel advertiser, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. The channel parameter must indicate the correct type of advertiser channel, so it must be 37, 38, or 39 for extended advertiser and in the range 0–36 for the secondary channel advertiser. The radio CPU shall also set up the PHY mode given in phyMode.mainMode. Furthermore, it shall set up the advertising channel access address and use the CRC initialization value 0x555555. The whitener shall be set up as defined in the whitening parameter. The radio CPU shall then configure the transmitter. For the secondary channel advertiser, the operation will go on with reception after transmission if the transmitted packet is scannable or connectable, and if a request packet is received, transmission of a response packet may follow.

In Bluetooth® Low Energy, advertising is usually done over all three primary advertising channels, followed by advertising on one or more secondary channels. To set this up, four (or more) command structures can be chained using the pNextOp parameter. The parameter structures can be the same for the three extended advertiser commands, and the output structures can be the same for all channels.

The transmitted packets always use the common extended advertising format. Such packets are constructed by the radio CPU based on the information in a descriptor as given in Table 26-103. The details of the packet handling are described in Section 26.8.9.1. The first transmitted packet is given by the descriptor pointed to by pParams->pAdvPkt.

If the transmitted packet contains an AuxPtr field, the Offset Units, and Aux Offset (see the Specification of the Bluetooth System, Version 5.0 listed in Related Documentation) fields shall be modified by the radio CPU so that they point to a packet transmitted at the time given by pParams->auxPtrType and pParams->auxPtrTime. The values allowed for pParams->auxPtrType are the same as the triggers used generally in commands, but only TRIG_ABSTIME, TRIG_NEVER, and the various relative times are allowed; TRIG_NEVER means that no modification of AuxPtr shall be done by the radio CPU. The radio CPU shall calculate the time from the start of the transmitted packet to the given time and find the smallest offset unit that can be used to represent that time. Aux Offset shall then be set to the time difference divided by the unit rounded down.

Section 26.8.9.2 and Section 26.8.9.3 describe the detail of operation of CMD_ADV_EXT and CMD_ADV_AUX.

26.8.9.1 Common Extended Advertising Packets

A transmitted packet using the common extended advertising format consists of the following:

- A header
- An extended header length and advertiser mode
- An optional extended header
- An optional payload

In the header, the PDU type is set to 0111b for all relevant types except AUX_CONNECT_RSP, where the PDU type is 1000b. The TxAdd and RxAdd bits are inserted automatically as described in the following. The length is

calculated by the radio CPU based on the extended header length and the payload length. The extended header length and advertiser mode fields are inserted from the corresponding fields of extHdrInfo. If the extended header length is greater than 0, the first byte of the extended header is the extended header flags. This is inserted by the radio CPU from extHdrFlags. The rest of the extended header is located in the buffer pointed to by pExtHeader.

The addresses in the extended header may be omitted from the buffer (in this case, the buffer pointed to by pExtHdr shall be 6 or 12 bytes shorter). If extHdrConfig.bSkipAdvA is 1 and the AdvA bit is 1 in extHdrInfo, the advertiser address is inserted from the buffer pointed to by pDeviceAddress in the parameter structure. In this case, the TxAdd bit of the header is set from the parameter structure bit advConfig.deviceAddrType. If extHdrConfig.bSkipAdvA is 0 and the AdvA bit is 1 in extHdrInfo, the TxAdd bit of the header is set from extHdrConfig.deviceAddrType. If the AdvA bit is 0 in extHdrInfo, the TxAdd bit of the header is 0. Similarly, for a directed advertising packet on a secondary channel, if extHdrConfig.bSkipTargetA is 1 and the TargetA bit is 1 in extHdrInfo, the target address is inserted from the buffer pointed to by pWhiteList in the parameter structure. In this case, the RxAdd bit of the header is set from the parameter structure bit advConfig.peerAddrType (inverted if the LSB of pWhiteList is 1 as described in Section 26.8.17). If extHdrConfig.bSkipTargetA is 0 and the TargetA bit is 1 in extHdrInfo, the RxAdd bit of the header is set from extHdrConfig.targetAddrType. If the TargetA bit is 0 in extHdrInfo, the RxAdd bit of the header is 0.

For AUX_CONNECT_RSP packets, the TargetA field and RxAdd bit of the header are always inserted from the received address of the AUX_CONNECT_REQ. If extHdrConfig.bSkipTargetA is 0, the buffer pointed to by pExtHeader shall contain a placeholder for the target address, while if extHdrConfig.bSkipTargetA is 1, the TargetA field shall be omitted.

A trigger to end the operation is set up by pParams->endTrigger. If the trigger that is defined by this parameter occurs, the radio operation shall continue to completion, but the status code after ending shall be BLE_DONE_ENDED and the result shall be FALSE. This can for instance be used to stop execution instead of proceeding with the next chained operation by use of the condition in the command structure. If the immediate command *CMD_STOP* is received by the radio CPU, it shall have the same meaning as the end trigger occurring, except that the status code after ending shall be CMD_DONE_STOPPED.

The output structure pOutput contains fields that give information on the command being run. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired. The fields shall be updated by the radio CPU as described in the following list. The list also indicates when interrupts shall be raised in the system CPU.

- When the ADV_EXT_IND, AUX_ADV_IND, or AUX_CHAIN_IND packet is transmitted, nTxAdvInd is incremented and a Tx_Done interrupt is raised.
- If an AUX_SCAN_RSP packet is transmitted, nTxScanRsp is incremented afterwards, and a Tx_Done interrupt is raised.
- If an AUX_CONNECT_RSP packet is transmitted, nTxConnectRsp is incremented afterwards, and a Tx_Done interrupt is raised.
- If an AUX_SCAN_REQ is received with CRC OK and the bIgnore flag cleared, nRxScanReq is incremented. If the payload length is 12 or less, an Rx_Empty interrupt is raised. If the payload length is greater than 12, an Rx_Ok interrupt is raised.
- If an AUX_CONNECT_REQ message is received with CRC OK and the bIgnore flag cleared, nRxConnectReq is incremented and an Rx_OK interrupt is raised.
- If a packet is received with CRC error, nRxNok is incremented and an Rx_Nok interrupt is raised.
- If a packet is received and the bIgnore flag is set, nRxIgnored is incremented and an Rx_Ignored interrupt is raised.
- If a packet is received that did not fit in the RX queue, nRxBufFull is incremented and an Rx_Buf_Full interrupt is raised.

- If a packet is received, lastRssi is set to the RSSI of that packet.
- If a packet is received, timeStamp is set to a timestamp of the start of that packet. For an AUX_CONNECT_REQ message, this can be used to calculate the anchor point of the first packet
- If the first RX data entry in the RX queue changed state to Finished after a packet was received, an Rx_Entry_Done interrupt is raised.

26.8.9.2 Extended Advertiser Command

An extended advertiser operation is started by a CMD_BLE5_ADV_EXT command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-103 and a pOutput parameter of the type defined in Table 26-108. The operation consists of transmission of one packet on a primary advertising channel. After transmission of an ADV_EXT_IND, the operation shall end without any receive operation.

An extended advertiser operation shall end with one of the statuses listed in Table 26-137. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-137. End of Extended Advertiser Operation

Condition	Status Code	Result
Transmitted ADV_EXT_IND	BLE_DONE_OK	TRUE
Observed trigger indicated by pParams->endTrigger, then finished transmitting ADV_EXT_IND	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then finished transmitting ADV_EXT_IND	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
Illegal parameter value	BLE_ERROR_PAR	ABORT
Aux pointer target time gives higher Aux Offset than can be represented numerically	BLE_ERROR_AUX	ABORT

26.8.9.3 Secondary Channel Advertiser Command

A secondary channel advertiser operation is started by a CMD_BLE5_ADV_AUX command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-104 and a pOutput parameter of the type defined in Table 26-108. The operation starts with transmission.

After an AUX_ADV_IND packet is transmitted, the receiver shall be started if the transmitted packet is connectable of scannable. If the transmitted advertising packet is scannable, the receiver shall look for an AUX_CONNECT_REQ message. If the transmitted advertising packet is connectable, the receiver shall look for an AUX_SCAN_REQ message.

An AUX_SCAN_REQ or AUX_CONNECT_REQ message is received into the RX queue given by pParams->pRxQ, as described in Section 26.10.4.1. The bCrcErr and bIgnore bits are set according to the CRC result and the received message. The AdvA field in the message, along with the TxAdd bit of the received header, is compared to the pParams->pDeviceAddress array and pParams->advConfig.deviceAddrType, respectively, to see if the message was addressed to this advertiser. The received ScanA or InitA field, along with the RxAdd bit of the received header, is checked as follows:

- If pParams->advConfig.bDirected is 1, ScanA or InitA is checked against the single address pointed to by pParams->pWhiteList, and RxAdd is checked against pParams->advConfig.peerAddrType
- If pParams->advConfig.bDirected is 0, depending on the advertising filter policy, given by pParams->advConfig.advFilterPolicy, the received ScanA or InitA field, along with the RxAdd bit of the received header, is checked against the whitelist as described in Section 26.8.14. If the resolvable private address (RPA) mode, given by pParams->advConfig.rpaMode, is nonzero, an extra check is done to see if the peer address is a resolvable private address. If the received TxAdd is 1 and the two most significant bits of the received ScanA or InitA field are 01b, the address is an RPA. If so, a whitelist check is performed regardless of the filter policy.

Depending on the received packet, the actions taken shall be as given in Table 26-138 and Table 26-139, where the definition of each action (including the value that will be used on bCrcErr and bIgnore) is given

in Table 26-140. If pParams->advConfig.bStrictLenFilter is 1, only length fields that are compliant with the Bluetooth® Low Energy specification shall be considered valid. For an AUX_SCAN_REQ packet, it means a length field of 12, and for an AUX_CONNECT_REQ packet, it means a length field of 34. If pParams->advConfig.bStrictLenFilter is 0, all received packets with a length field less than or equal to the maximum length of an advertiser packet (255, but can be overridden) shall be considered valid. If the length is not valid, the receiver shall be stopped.

Table 26-138. Actions to Take Based on Received Packets for Scannable Advertiser (extHdrInfo.advMode = 2)(1)

PDU Type	CRC Result	Directed	Valid Length	AdvA Match	Filter Policy	RPA Mode	Resolvable Private Address	ScanA or InitA Match	Action No.
AUX_SCAN_REQ	OK	X	Yes	No	X	X	X	X	1
AUX_SCAN_REQ	OK	No	Yes	Yes	2 or 3	X	X	No	1
AUX_SCAN_REQ	OK	No	Yes	Yes	2 or 3	X	X	Yes	2
AUX_SCAN_REQ	OK	No	Yes	Yes	0 or 1	0	X	X	2
AUX_SCAN_REQ	OK	No	Yes	Yes	0 or 1	1	No	X	2
AUX_SCAN_REQ	OK	No	Yes	Yes	0 or 1	1	Yes	No	1
AUX_SCAN_REQ	OK	No	Yes	Yes	0 or 1	1	Yes	Yes	2
AUX_SCAN_REQ	OK	Yes	Yes	Yes	X	X	X	No	1
AUX_SCAN_REQ	OK	Yes	Yes	Yes	X	X	X	Yes	2
AUX_SCAN_REQ	NOK	X	Yes	X	X	X	X	X	3
AUX_SCAN_REQ	X	X	No	X	X	X	X	X	5
Other	X	X	X	N/A	X	X	N/A	N/A	5
No packet received	N/A	X	N/A	N/A	X	X	N/A	N/A	5

(1) X = Don't care. N/A = Not applicable.

Table 26-139. Actions to Take Based on Received Packets for Connectable Advertiser (extHdrInfo.advMode = 1)(1)

PDU Type	CRC Result	Directed	Valid Length	AdvA Match	Filter Policy	RPA Mode	Resolvable Private Address	ScanA or InitA Match	Action No.
AUX_CONNECT_REQ	OK	X	Yes	No	X	X	X	X	1
AUX_CONNECT_REQ	OK	No	Yes	Yes	2 or 3	X	X	No	1
AUX_CONNECT_REQ	OK	No	Yes	Yes	2 or 3	X	X	Yes	4
AUX_CONNECT_REQ	OK	No	Yes	Yes	0 or 1	0	X	X	4
AUX_CONNECT_REQ	OK	No	Yes	Yes	0 or 1	1	No	X	4
AUX_CONNECT_REQ	OK	No	Yes	Yes	0 or 1	1	Yes	No	1
AUX_CONNECT_REQ	OK	No	Yes	Yes	0 or 1	1	Yes	Yes	4
AUX_CONNECT_REQ	OK	Yes	Yes	Yes	X	X	X	No	1
AUX_CONNECT_REQ	OK	Yes	Yes	Yes	X	X	X	Yes	4
AUX_CONNECT_REQ	NOK	X	Yes	X	X	X	X	X	3
AUX_CONNECT_REQ	X	X	No	X	X	X	X	X	5
Other	X	X	X	N/A	X	X	N/A	N/A	5
No packet received	N/A	X	N/A	N/A	X	X	N/A	N/A	5

(1) X = Don't care. N/A = Not applicable.

Table 26-140. Descriptions of the Actions to Take on Received Packets

Action No.	bCrcErr	bIgnore	Description
1	0	1	End operation with BLE_DONE_OK status
2	0	0	Transmit AUX_SCAN_RSP message
3	1	0	End operation with BLE_DONE_RXERR status
4	0	0	Transmit AUX_CONNECT_RSP message

Table 26-140. Descriptions of the Actions to Take on Received Packets (continued)

Action No.	bCrcErr	bIgnore	Description
5	—	—	Stop receiver immediately and end operation with BLE_DONE_NOSYNC status

If an AUX_SCAN_REQ packet is received with a length of 12 (or less), it shall be viewed as an empty packet. This means that if pParams->rxConfig.bAutoflushEmpty is 1 and bCrcErr and bIgnore are both 0, the packet is removed from the RX buffer. If a packet is flagged by bIgnore or bCrcErr, it may also be removed, based on the bits in pParams->rxConfig.

If the packet being received did not fit in the RX queue, the packet is received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX queue based on the bits in pParams->rxConfig, the command shall end.

If the next action according to Table 26-140 is to transmit an AUX_SCAN_RSP or AUX_CONNECT_RSP message, the radio CPU shall start the transmitter to transmit this packet from the descriptor in pParams->pRspPkt as described in Section 26.8.9.1.

A secondary channel advertiser operation shall end with one of the statuses listed in Table 26-141. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-141. End of Secondary Channel Advertiser Operation

Condition	Status Code	Result
Performed action 1 after running receiver	BLE_DONE_OK	TRUE
Performed action 2 and transmitted AUX_SCAN_RSP	BLE_DONE_OK	TRUE
Performed action 3 after running receiver	BLE_DONE_RXERR	TRUE
Performed action 4 and transmitted AUX_CONNECT_RSP	BLE_DONE_CONNECT	FALSE
Performed action 5 after running receiver	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->endTrigger, then performed Actions 1, 2, 3, or 5	BLE_DONE_ENDED	FALSE
Observed CMD_STOP, then performed Actions 1, 2, 3, or 5	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal parameter	BLE_ERROR_PAR	ABORT
Aux pointer target time gives higher Aux Offset than can be represented numerically	BLE_ERROR_AUX	ABORT

26.8.10 Scanner Commands

A scanner operation is started by a CMD_BLE_SCANNER or CMD_BLE5_SCANNER command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-97 or Table 26-105 and a pOutput parameter of the type defined in Table 26-97 or Table 26-111. At the start of a scanner operation, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. For CMD_BLE_SCANNER, the channel parameter is not allowed to be in the range 0–36 because they are not primary advertising channels. For CMD_BLE5_SCANNER, it shall also set up the PHY mode given in phyMode.mainMode. The radio CPU shall set up the advertising channel access address and use the CRC initialization value 0x555555. The whitener shall be set up as defined in the whitening parameter. The radio CPU shall then configure receiver.

After tuning to the correct channel, the radio CPU shall start listening for an advertising channel packet. If sync is obtained on the demodulator, the message is received into the RX queue. The header is checked, and if it is not an advertising packet, reception shall be stopped and sync search shall be restarted. The packets accepted and further operation depends on the command run, the channel type, and PHY mode (see the descriptions

in Section 26.8.10.1, Section 26.8.10.2, and Section 26.8.10.3). All scanner commands end as described in Section 26.8.10.5.

26.8.10.1 Scanner Receiving Legacy Advertising Packets on Primary Channel

Legacy advertising packets (ADV_IND, ADV_NONCONN_IND, ADV_SCAN_IND, and ADV_DIRECT_IND), are accepted if running CMD_BLE_SCANNER or if running CMD_BLE5_SCANNER on a primary advertising channel with 1 Mbps PHY. The operation is described as follows:

- The bCrcErr and bIgnore bits are set according to the CRC result and the received message.
- Depending on the scanning filter policy, given by pParams->scanConfig.scanFilterPolicy, the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14. If the resolvable private address (RPA) mode, given by pParams->scanConfig.rpaMode, is nonzero, an extra check is done to see if the peer address is a resolvable private address.
- If the received TxAdd is 1 and the two most significant bits of the received AdvA field are 01b, the address is an RPA. If so, a whitelist check is performed regardless of the filter policy. The complete check on AdvA is performed as listed in Table 26-142. For ADV_DIRECT_IND messages, the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->scanConfig.deviceAddrType, respectively. If the RPA filter policy, given by pParams->scanConfig.rpaFilterPolicy, is 1, a packet is also accepted if InitA and RxAdd indicate an RPA. The full filtering of ADV_DIRECT_IND messages is given in Table 26-143.
- Depending on the received packet, and whether the scan is active or passive, signaled in pParams->scanConfig.bActiveScan, the actions taken shall be as given in Table 26-144, where the definition of each action, including the value that will be used on bCrcErr and bIgnore, is given in Table 26-145.
- If pParams->scanConfig.bStrictLenFilter is 1, only length fields that are compliant with the Bluetooth® Low Energy specification shall be considered valid. For an ADV_DIRECT_IND, that means a length field of 12, and for other ADV*_IND messages it means a length field in the range 6–37.
- If pParams->advConfig.bStrictLenFilter is 0, all received packets with a length field less than or equal to the maximum length of an advertiser packet shall be considered valid. If the length is not valid, the receiver shall be stopped.

Table 26-142. Filtering on Received Advertiser Address

Filter Policy	RPA Mode	AdvA is RPA	AdvA Matches Whitelist Entry with bWIIgn = 1	AdvA Matches Whitelist Entry with bEnable = 1	AdvA Filter Result	Perform Auto-Ignore if bAutoWIIgnore = 1
1	X	X	No	No	Reject	No
1	X	X	No	Yes	Accept	Yes
1	X	X	Yes	X	Reject	No
0	0	X	No	X	Accept	No
0	0	X	Yes	X	Reject	No
0	1	No	No	X	Accept	No
0	1	No	Yes	X	Reject	No
0	1	Yes	No	No	Reject	No
0	1	Yes	No	Yes	Accept	Yes
0	1	Yes	Yes	X	Reject	No

Table 26-143. Filtering on Received Initiator Address of ADV_DIRECT_IND Packets

RPA Filter Policy	TargetA is RPA	TargetA is Equal to Device Address	TargetA Match
0	X	No	No
0	X	Yes	Yes
1	No	No	No
1	No	Yes	Yes

Table 26-143. Filtering on Received Initiator Address of ADV_DIRECT_IND Packets (continued)

RPA Filter Policy	TargetA is RPA	TargetA is Equal to Device Address	TargetA Match
1	Yes	X	Yes

Table 26-144. Actions on Received Packets by Scanner

PDU Type	CRC Result	AdvA Filter Result	TargetA Match	Active Scan	Action No.
ADV_IND	OK	Reject	N/A	X	1
ADV_IND	OK	Accept	N/A	No	2
ADV_IND	OK	Accept	N/A	Yes	3
ADV_IND	NOK	X	N/A	X	4
ADV_SCAN_IND	OK	Reject	N/A	X	1
ADV_SCAN_IND	OK	Accept	N/A	No	2
ADV_SCAN_IND	OK	Accept	N/A	Yes	3
ADV_SCAN_IND	NOK	X	N/A	X	4
ADV_NONCONN_IND	OK	Reject	N/A	X	1
ADV_NONCONN_IND	OK	Accept	N/A	X	2
ADV_NONCONN_IND	NOK	X	N/A	X	4
ADV_DIRECT_IND	OK	Reject	X	X	1
ADV_DIRECT_IND	OK	Accept	No	X	1
ADV_DIRECT_IND	OK	Accept	Yes	X	2
ADV_DIRECT_IND	NOK	X	X	X	4
ADV*_IND with invalid length	X	X	X	X	5
Other (see also Section 26.8.10.2)	X	X	N/A	X	5

Table 26-145. Descriptions of the Actions to Take on Packets Received by Scanner

Action No.	bCrcErr	bIgnore	Description
1	0	1	Continue scanning
2	0	0	Continue scanning or end operation with BLE_DONE_OK status
3	0	0	Perform backoff procedure and send SCAN_REQ and receive SCAN_RSP if applicable. Then continue scanning or end operation
4	1	0	Continue scanning
5	—	—	Stop receiving packet, then continue scanning

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 3, a SCAN_REQ message is to be transmitted if allowed after a backoff procedure. This procedure starts with decrementing pParams->backoffCount. If this variable becomes 0 after the decrement, a SCAN_REQ message shall be transmitted. If not, the operation shall end.

If the action from the received packet is 2 or 3, the next action may either be to continue scanning or to end the operation. This is configured with pParams->scanConfig.bEndOnRpt; if 1, the operation shall end, otherwise scanning shall continue.

When transmitting a SCAN_REQ message, the radio CPU shall construct this packet. In the header, the PDU Type bits shall be 0011b. The TxAdd bit shall be as in pParams->scanConfig.deviceAddrType (if the least significant bit of pParams->pDeviceAddress is 1, the TxAdd bit is inverted, see Section 26.8.17). The RxAdd bit shall be as in the TxAdd field of the header of the received ADV_IND or ADV_SCAN_IND message. The length shall be calculated from the size of the scan request data, meaning that it shall be pParams->scanReqLen + 12. The ChSel and RFU bits shall be 0. The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress, followed by the 6-byte peer address read from the AdvA field of the received message. By the Bluetooth low energy specification, there shall be no more payload, but when

using the CMD_BLE_SCANNER command, a noncompliant message may be constructed by setting pParams->scanReqLen to a nonzero value. If so, the rest of the payload shall be read from the pParams->pScanData buffer.

After a SCAN_REQ message is transmitted, the radio CPU shall configure receiver and look for a SCAN_RSP from the advertiser to which the SCAN_REQ was sent. If sync is obtained on the demodulator, the header is shall be checked once it is received, and if it is not a SCAN_RSP message, the demodulator shall be stopped immediately. If it is a SCAN_RSP message, then it shall be received into the RX queue. Depending on the received SCAN_RSP message, the values of bCrcErr and bIgnore shall be as given in Table 26-146. If pParams->scanConfig.bStrictLenFilter is 1, only length fields that are compliant with the Bluetooth low energy specification shall be considered valid. For a SCAN_RSP, it means a length field in the range 6–37. If pParams->scanConfig.bStrictLenFilter is 0, all received packets with a length field less than or equal to the maximum length of an advertiser packet shall be considered valid. If the length is not valid, the receiver shall be stopped.

Table 26-146. Actions on Packets Received by Scanner After Transmission of SCAN_REQ

PDU Type	CRC Result	AdvA Same as in Request	bCrcErr	bIgnore	Response Packet Result
SCAN_RSP	OK	No	0	1	Failure
SCAN_RSP	OK	Yes	0	0	Success
SCAN_RSP	NOK	X	1	0	Failure
SCAN_RSP with invalid length	X	X	—	—	Failure
Other	X	N/A	—	—	Failure
No packet received	N/A	N/A	—	—	Failure

After receiving or attempting to receive a SCAN_RSP, the backoff parameters shall be updated by the radio CPU as described in Section 26.8.15, based on the result as given in the *Response Packet Result* column of Table 26-146 and the old values of the backoff parameters.

If pParams->scanConfig.bAutoWlIgnore is 1, an auto-ignore feature is enabled. In the cases where stated in the right-most column of Table 26-152, the radio CPU shall then automatically set the bWlIgn bit of the whitelist entry corresponding to the address from which an ADV*_IND message was received. This shall be done either after action 2 is performed or after action 3 is performed and a SCAN_RSP is received with the result Success. This can be used to avoid reporting multiple advertising messages from the same device and to avoid scanning the same device repeatedly.

26.8.10.2 Scanner Receiving Extended Advertising Packets on Primary Channel

Extended advertising packets on primary channel (ADV_EXT_IND), are accepted if running CMD_BLE5_SCANNER on a primary advertising channel, prior to following an AuxPtr.

The bCrcErr and bIgnore bits are set according to the CRC result and the received message. If the extended header flags indicate that AdvA is present, the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14. If the resolvable private address (RPA) mode, given by pParams->scanConfig.rpaMode, is nonzero, an extra check is done to see if the peer address is a resolvable private address. If the received TxAdd is 1 and the two most significant bits of the received AdvA field are 01b, the address is an RPA. If so, a whitelist check is performed regardless of the filter policy. The complete check on AdvA is performed as listed in Table 26-142; however, the rule on the bWlIgn bit set in the whitelist is only applied if no ADI field is present in the extended header and pParams->extFilterConfig.bApplyDuplicateFiltering is 1. The ADI result is found as described in Section 26.8.10.4. If the extended header flags indicate that AdvA is not present, the AdvA filter result is always Accept. If the extended header flags indicate that TargetA is present (that is, the packet is directed), the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->scanConfig.deviceAddrType, respectively. If the RPA filter policy, given by pParams->scanConfig.rpaFilterPolicy, is 1, a packet is also accepted if TargetA and RxAdd indicate an RPA. The full filtering of directed messages is given in Table 26-143. Nondirected messages always have TargetA match. Depending on the received packet, the actions taken shall be as given in Table 26-147, where the definition of each action (including the value that will be used on bCrcErr and bIgnore) is given in Table 26-148. The packet length of a received ADV_EXT_IND packet

is always valid by default, but it is possible to configure a maximum length by overriding the firmware defined parameter maxAdvExtLen. In addition, the extended header length and flags are checked. If the extended header length is too large for the extended header to fit in the packet, or if it is too small to hold the configured, the length is invalid. If pParams->scanConfig.bStrictLenFilter is 1, all defined fields are considered, while if pParams->scanConfig.bStrictLenFilter is 0, the fields that are not automatically checked by the CPE (SyncInfo and TxPower) are ignored when finding the minimum allowed extended header length.

Table 26-147. Actions on Received Extended Advertiser Packets by Scanner

PDU Type	CRC Result	AdvA Filter Result	TargetA Match	ADI Result	AuxPtr Present	Action No.
ADV_EXT_IND	OK	Reject	X	X	X	1
ADV_EXT_IND	OK	Accept	No	X	X	1
ADV_EXT_IND	OK	Accept	Yes	Reject	X	1
ADV_EXT_IND	OK	Accept	Yes	Accept	No	2
ADV_EXT_IND	OK	Accept	Yes	Accept	Yes	6
ADV_EXT_IND	NOK	X	X	X	X	4
ADV_EXT_IND with invalid lengths	X	X	X	X	X	5
Other (see also Section 26.8.10.1)	X	X	X	X	X	5

Table 26-148. Descriptions of the Actions to Take on Extended Advertiser Packets Received by Scanner

Action No.	bCrcErr	bIgnore	Description
1	0	1	Continue scanning
2	0	0	Continue scanning or end operation with BLE_DONE_OK status
4	1	0	Continue scanning
5	—	—	Stop receiving packet, then continue scanning
6	0	0	Follow Aux pointer and receive on the new channel, or end operation with BLE_DONE_AUX

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 2, the next action may either be to continue scanning or to end the operation. This is configured with pParams->scanConfig.bEndOnRpt; if 1, the operation shall end, otherwise scanning shall continue.

If pParams->extFilterConfig.bAutoWlIgnore is 1, an auto-ignore feature is enabled. In the cases where stated in the right-most column of Table 26-142 and no ADI field is present in the extended header, the radio CPU shall then automatically set the bWlIgn bit of the whitelist entry corresponding to the address from which an ADV_EXT_IND message was received. This shall be done either after action 2 is performed or when action 6 is about to be performed. This can be used to avoid reporting multiple advertising messages from the same device and to avoid scanning the same device repeatedly.

If the action from the received packet is 6, the radio CPU shall follow the procedure in Section 26.8.16. This will either cause the receiver to look for a secondary advertising packet as described in Section 26.8.10.3 or to end the operation with status BLE_DONE_AUX.

26.8.10.3 Scanner Receiving Extended Advertising Packets on Secondary Channel

Extended advertising packets on secondary channel (AUX_ADV_IND and AUX_CHAIN_IND), are accepted if running CMD_BLE5_SCANNER on a secondary advertising channel, or if following an AuxPtr.

The bCrcErr and bIgnore bits are set according to the CRC result and the received message. Depending on the scanning filter policy, given by pParams->scanConfig.scanFilterPolicy, the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14. If the resolvable private address (RPA) mode, given by pParams->scanConfig.rpaMode, is nonzero,

an extra check is done to see if the peer address is a resolvable private address. If the received TxAdd is 1 and the two most significant bits of the received AdvA field are 01b, the address is an RPA. If so, a whitelist check is performed regardless of the filter policy. The complete check on AdvA is performed as listed in Table 26-152; however, entries with the bWIIgn bit set in the whitelist are only considered if no ADI field is present in the extended header and pParams->extFilterConfig.bApplyDuplicateFiltering is 1. The ADI result is found as described in Section 26.8.10.4. If the extended header flags indicate that TargetA is present (that is, the packet is directed), the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->scanConfig.deviceAddrType, respectively (see also Section 26.8.17). If the RPA filter policy, given by pParams->scanConfig.rpaFilterPolicy, is 1, a packet is also accepted if TargetA and RxAdd indicate an RPA. The full filtering of directed messages is given in Table 26-143. Nondirected messages always have TargetA match. Depending on the received packet, and whether the scan is active or passive, signaled in pParams->scanConfig.bActiveScan, the actions taken shall be as given in Table 26-149, where the definition of each action, including the value that will be used on bCrcErr and bIgnore, is given in Table 26-150. The packet length of a received AUX_ADV_IND or AUX_CHAIN_IND packet is always valid by default, but it is possible to configure a maximum length by overriding the firmware defined parameter maxAdvExtLen. In addition, the extended header length and flags are checked. If the extended header length is too large for the extended header to fit in the packet, or if it is too small to hold the configured, the length is invalid. If pParams->scanConfig.bStrictLenFilter is 1, all defined fields are considered, while if pParams->scanConfig.bStrictLenFilter is 0, the fields that are not automatically checked by the CPE (SyncInfo and TxPower) are ignored when finding the minimum allowed extended header length.

Table 26-149. Actions on Received Secondary Channel Packets by Scanner

PDU Type	AdvMode	CRC Result	AdvA Filter Result	TargetA Match	ADI Result	AuxPtr Present	Active Scan	Action No.
AUX_ADV_IND	X	OK	Reject	X	X	X	X	1
AUX_ADV_IND	X	OK	Accept	No	X	X	X	1
AUX_ADV_IND	X	OK	Accept	Yes	Reject	X	X	1
AUX_ADV_IND	00	OK	Accept	Yes	Accept	No	X	2
AUX_ADV_IND	00	OK	Accept	Yes	Accept	Yes	X	6
AUX_ADV_IND	01, 11	OK	Accept	Yes	Accept	X	X	2
AUX_ADV_IND	10	OK	Accept	Yes	Accept	X	No	2
AUX_ADV_IND	10	OK	Accept	Yes	Accept	X	Yes	3
AUX_ADV_IND	X	NOK	X	X	X	X	X	4
AUX_ADV_IND with invalid length	X	X	X	X	X	X	X	5
Other	X	X	X	X	X	X	X	5

Table 26-150. Descriptions of the Actions to Take on Secondary Channel Packets Received by Scanner

Action No.	bCrcErr	bIgnore	Description
1	0	1	End operation with BLE_DONE_RXERR status
2	0	0	End operation with BLE_DONE_OK status
3	0	0	Perform backoff procedure and send AUX_SCAN_REQ and receive AUX_SCAN_RSP if applicable. Then end operation
4	1	0	End operation with BLE_DONE_RXERR status
5	—	—	Stop receiving packet, then continue scanning
6	0	0	Follow Aux pointer and receive on the new channel, or end operation with BLE_DONE_AUX

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 3, an AUX_SCAN_REQ is to be transmitted if allowed after a backoff procedure. This procedure starts with decrementing pParams->backoffCount. If this variable becomes 0 after the decrement, an AUX_SCAN_REQ shall be transmitted. If not, the operation shall end.

When transmitting an AUX_SCAN_REQ, the radio CPU shall construct this packet. In the header, the PDU Type bits shall be 0011b. The TxAdd bit shall be as in pParams->scanConfig.deviceAddrType (if the least significant bit of pParams->pDeviceAddress is 1, the TxAdd bit is inverted, see Section 26.8.17). The RxAdd bit shall be as in the TxAdd field of the header of the received AUX_ADV_IND message. The length shall be 12. The ChSel and RFU bits shall be 0. The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress, followed by the 6-byte peer address read from the AdvA field of the received message.

After an AUX_SCAN_REQ message is transmitted, the radio CPU shall configure receiver and look for an AUX_SCAN_RSP. The packet will be received using the rules from Table 26-149, except that action 3 is not performed, and action 2 is performed instead, and action 5 in this case is end operation with BLE_DONE_NOSYNC status. This means that a packet may be accepted even if it has a different AdvA than the first AUX_ADV_IND packet. To correct this issue, a check could be made in the system CPU, or a patch could be considered.

After receiving or attempting to receive an AUX_SCAN_RSP, the backoff parameters shall be updated by the radio CPU, based on the result as given in the *Response Packet Result* column of Table 26-146 and the old values of the backoff parameters.

If the action from the received packet is 6, the radio CPU shall follow the procedure in Section 26.8.16. This will either cause the receiver to look for a secondary advertising packet as described in this section or to end the operation with status BLE_DONE_AUX.

26.8.10.4 ADI Filtering

If a received extended advertiser packet contains an AdvDataIndex field in the extended header and pParams->extFilterConfig.bCheckAdi is 1, the AdvDataIndex field is checked against an entry of the list pointed to by pParams->pAdiList. This list consists of 16 entries of the type defined in Table 26-117.

The AdvDataIndex field consists of an Advertising Set ID (SID) and an Advertising Data ID (DID) field. When checking against the list, the entry with the index given by the received SID is used. The mode field of this entry is first checked. If it is 0, no filtering is used for this DID, and the ADI is accepted. If mode is 2, the ADI is always rejected. If mode is 1, the received DID is checked against the advDataId field of the entry. If they are equal, the ADI is rejected, otherwise it is accepted.

If the packet is received with CRC OK, not ignored, and the buffer did not go full, and pParams->extFilterConfig.bCheckAdi and pParams->extFilterConfig.bAutoAdiUpdate are both 1, the radio CPU shall update the ADI list automatically. It shall then copy the received DID field into the advDataId field of the entry given by the received SID.

Note

If the mode is 0, the copying is done, but the mode is not modified automatically, future packets with the same DID will be ignored, the mode must be modified by the system CPU.

26.8.10.5 End of Scanner Commands

Two triggers to end the operation are set up by pParams->endTrigger/pParams->endTime and pParams->timeoutTrigger/pParams->timeoutTime, respectively. The triggers have the following behavior:

- If the timeout trigger occurs while waiting for sync on an advertiser packet, the operation shall end immediately
- If the timeout trigger occurs at another time while operating on a primary channel, the operation shall continue until the scan would otherwise be resumed on the primary channel, and then end.
- If an Aux pointer is followed, any timeout trigger (past or future) is ignored.
- If the end trigger occurs while waiting for sync on an advertiser packet, the operation shall end immediately.
- If the end trigger occurs at another time, the operation shall continue until an Aux pointer would be followed or the scan would otherwise be resumed on the primary channel, and then end.

If the immediate command CMD_STOP is received by the radio CPU, it shall have the same meaning as the end trigger occurring, except that the status code after ending shall be CMD_DONE_STOPPED. Typically, timeoutTrigger can be used at the end of a scan window, while endTrigger can be used when scanning is to end entirely.

The output structure pOutput contains fields which give information on the command being run. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired. The fields shall be updated by the radio CPU as described below. The list also indicates when interrupts shall be raised in the system CPU.

- If a SCAN_REQ or AUX_SCAN_REQ packet is transmitted, nTxScanReq (CMD_BLE_SCANNER) or nTxReq (CMD_BLE5_SCANNER) is incremented and a Tx_Done interrupt is raised.
- If a SCAN_REQ or AUX_SCAN_REQ is not transmitted due to the backoff procedure, nBackedOffScanReq (CMD_BLE_SCANNER) or nBackedOffReq (CMD_BLE5_SCANNER) is incremented
- If an *ADV*_IND or AUX_CHAIN_IND packet is received with CRC OK and the bIgnore flag cleared, nRxAdvOk is incremented, an Rx_Ok interrupt is raised, and timeStamp is set to a timestamp of the start of the packet.
- If an *ADV*_IND or AUX_CHAIN_IND packet is received with CRC OK and the bIgnore flag set, nRxAdvIgnored is incremented and an Rx_Ignored interrupt is raised.
- If an *ADV*_IND or AUX_CHAIN_IND packet is received with CRC error, nRxAdvNok is incremented and an Rx_Nok interrupt is raised.
- If an *ADV*_IND or AUX_CHAIN_IND packet is received and did not fit in the RX queue, nRxAdvBufFull is incremented and an Rx_Buf_Full interrupt is raised.
- If a SCAN_RSP or AUX_SCAN_RSP packet is received with CRC OK and the bIgnore flag cleared, nRxScanRspOk (CMD_BLE_SCANNER) or nRxRspOk (CMD_BLE5_SCANNER) is incremented and an Rx_Ok interrupt is raised.
- If a SCAN_RSP or AUX_SCAN_RSP packet is received with CRC OK and the bIgnore flag set, nRxScanRspIgnored (CMD_BLE_SCANNER) or nRxRspIgnored (CMD_BLE5_SCANNER) is incremented and an Rx_Ignored interrupt is raised.
- If a SCAN_RSP or AUX_SCAN_RSP packet is received with CRC error, nRxScanRspNok (CMD_BLE_SCANNER) or nRxRspNok (CMD_BLE5_SCANNER) is incremented and an Rx_Nok interrupt is raised.
- If a SCAN_RSP or AUX_SCAN_RSP packet is received and did not fit in the RX queue, nRxScanRspBufFull (CMD_BLE_SCANNER) or nRxRspBufFull (CMD_BLE5_SCANNER) is incremented and an Rx_Buf_Full interrupt is raised.
- If a packet is received, lastRssi is set to the RSSI of that packet.
- If the first RX data entry in the RX queue changed state to Finished after a packet was received, an Rx_Entry_Done interrupt is raised.

A scanner operation shall end with one of the statuses listed in Table 26-151. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action.

Table 26-151. End of Scanner Operation

Condition	Status Code	Result
Performed action 2 with pParams->scanConfig.bEndOnRpt = 1 or when receiving on secondary channel	BLE_DONE_OK	TRUE
Performed action 3 with pParams->scanConfig.bEndOnRpt = 1 or on secondary channel, and did not send SCAN_REQ or AUX_SCAN_REQ due to backoff	BLE_DONE_OK	TRUE
Performed action 3 with pParams->scanConfig.bEndOnRpt = 1 or on secondary channel, sent SCAN_REQ or AUX_SCAN_REQ and received SCAN_RSP or AUX_SCAN_RSP with bCrcErr = 0 and bIgnore = 0	BLE_DONE_OK	TRUE
Performed action 1 or action 4 on secondary channel	BLE_DONE_RXERR	TRUE
Performed action 3 with pParams->scanConfig.bEndOnRpt = 1 or on secondary channel, sent SCAN_REQ or AUX_SCAN_REQ and received SCAN_RSP or AUX_SCAN_RSP with bCrcErr = 1 or bIgnore = 1	BLE_DONE_RXERR	TRUE
Performed action 3 with pParams->scanConfig.bEndOnRpt = 1 or on secondary channel, sent SCAN_REQ or AUX_SCAN_REQ, but did not get sync or found wrong packet type or invalid length	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->timeoutTrigger while waiting for sync on advertiser packet	BLE_DONE_RXTIMEOUT	TRUE
Observed trigger indicated by pParams->timeoutTrigger while on primary channel, then performed actions 1, 2, 3, 4, or 5.	BLE_DONE_RXTIMEOUT	TRUE
Observed trigger indicated by pParams->endTrigger while waiting for sync on advertiser packet	BLE_DONE_ENDED	FALSE
Observed trigger indicated by pParams->endTrigger, then performed actions 1, 2, 3, 4, 5, or 6.	BLE_DONE_ENDED	FALSE
Observed CMD_STOP while waiting for sync on advertiser packet	BLE_DONE_STOPPED	FALSE
Observed CMD_STOP, then performed actions 1, 2, 3, 4, 5, or 6.	BLE_DONE_STOPPED	FALSE
Performed action 6 with wait time to AUX packet more than pParams->maxWaitTimeForAuxCh.	BLE_DONE_AUX	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT
Scan request data length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.11 Initiator Command

An initiator operation is started by a CMD_BLE_INITIATOR or CMD_BLE5_INITIATOR command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-98 or Table 26-106 and a pOutput parameter of the type defined in Table 26-98 or Table 26-111. At the start of an initiator operation, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. For CMD_BLE_INITIATOR, the channel parameter is not allowed to be in the range 0–36 because they are not primary advertising channels. For CMD_BLE5_INITIATOR, it shall also set up the PHY mode given in phyMode.mainMode. The radio CPU shall set up the advertising channel access address and use the CRC initialization value 0x555555. The whitener shall be set up as defined in the whitening parameter. The radio CPU shall then configure receiver.

After tuning to the correct channel, the radio CPU shall start listening for an advertising channel packet. If sync is obtained on the demodulator, the message is received into the RX queue. The header is checked, and if it is not a connectable advertising packet, reception shall be stopped and sync search shall be restarted. The packets accepted and further operation depends on the command run and the channel type and PHY mode.

26.8.11.1 Initiator Receiving Legacy Advertising Packets on Primary Channel

Legacy advertising packets (ADV_IND and ADV_DIRECT_IND), are accepted if running CMD_BLE_INITIATOR or if CMD_BLE5_INITIATOR on a primary advertising channel with 1-Mbps PHY. The operation is a described in the following list:

- The bCrcErr and bIgnore bits are set according to the CRC result and the received message.
- The parameter pParams->initConfig.bUseWhiteList determines if the initiator should try to connect to a specific device or against the whitelist. If this parameter is 0, the whitelist is not used, and pParams->pWhiteList shall point to a buffer containing only the device address of the device to connect to.
- The address type of the peer shall be given in pParams->advConfig.peerAddrType (see also Section 26.8.17). Otherwise, pParams->pWhiteList shall point to a whitelist, and the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14.
- The filtering is summarized in Table 26-152.
- For ADV_DIRECT_IND messages, the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->initConfig.deviceAddrType, respectively (see also Section 26.8.17). Depending on this, the actions taken shall be as given in Table 26-153, where the definition of each action (including the value that will be used on bCrcErr and bIgnore) is given in Table 26-154.
- If pParams->initConfig.bStrictLenFilter is 1, only length fields that are compliant with the Bluetooth low energy specification shall be considered valid.
- For an ADV_DIRECT_IND, that means a length field of 12, and for ADV_IND messages it means a length field in the range 6–37.
- If pParams->initConfig.bStrictLenFilter is 0, all received packets with a length field less than or equal to the maximum length of an advertiser packet shall be considered valid. If the length is not valid, the receiver shall be stopped.

Table 26-152. Filtering on Received Advertiser Address

bUseWhiteList	AdvA Matches Specific Peer Address	AdvA Matches Whitelist Entry with bEnable = 1	AdvA Matches Whitelist Entry with bIrkValid = 1	AdvA Filter Result
0	No	N/A	N/A	Reject
0	Yes	N/A	N/A	Accept
1	N/A	No	X	Reject
1	N/A	Yes	No	Accept
1	N/A	Yes	Yes	Reject

Table 26-153. Actions on Received Packets by Initiator

PDU Type	CRC result	AdvA filter result	TargetA match	Action No.
ADV_IND	OK	Reject	N/A	1
ADV_IND	OK	Accept	N/A	3
ADV_IND	NOK	X	N/A	4
ADV_DIRECT_IND	OK	Reject	X	1
ADV_DIRECT_IND	OK	Accept	No	1
ADV_DIRECT_IND	OK	Accept	Yes	3
ADV_DIRECT_IND	NOK	X	X	4
ADV*_IND with invalid length	X	X	X	5
Other (see also Section 26.8.11.2)	X	N/A	N/A	5

Table 26-154. Descriptions of the Actions to Take on Packets Received by Initiator

Action No.	bCrcErr	bIgnore	Description
1	0	1	Continue scanning
3	0	0	Send CONNECT_IND and end operation

Table 26-154. Descriptions of the Actions to Take on Packets Received by Initiator (continued)

Action No.	bCrcErr	bIgnore	Description
4	1	0	Continue scanning
5	—	—	Stop receiving packet, then continue scanning

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 2, a CONNECT_IND (known as CONNECT_REQ in Bluetooth® 4.0, 4.1, and 4.2 Specifications listed in Related Documentation) packet shall be transmitted. When transmitting a CONNECT_IND, the radio CPU shall construct this packet. In the header, the PDU Type bits shall be 0101b. The TxAdd bit shall be as in pParams->initConfig.deviceAddrType (if the least significant bit of pParams->pDeviceAddress is 1, the TxAdd bit is inverted, see Section 26.8.17). The RxAdd bit shall be as in the TxAdd field of the header of the received ADV_IND or ADV_DIRECT_IND message. The ChSel bit shall be 1 if pParams->initConfig.chSel is 1; otherwise, ChSel shall be 0. The length shall be calculated from the length of the LLData, meaning that it shall be pParams->connectReqLen + 12. The RFU bit shall be 0. The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress, followed by the 6-byte peer address read from the AdvA field of the received message. The rest of the payload shall be read from the pParams->pConnectData buffer. If pParams->initConfig.bDynamicWinOffset is 1, the radio CPU shall replace the bytes in the WinSize and WinOffset position (see Specification of the Bluetooth® System, Version 5.0 listed in Related Documentation) with a calculated value as explained in Section 26.8.11.4. After a CONNECT_IND message is transmitted, the operation shall end. If pParams->initConfig.chSel is 1, the ChSel bit of the received ADV_IND or ADV_DIRECT_IND packet shall be checked to determine the end status; see Section 26.8.11.5.

26.8.11.2 Initiator Receiving Extended Advertising Packets on Primary Channel

Extended advertising packets on primary channel (ADV_EXT_IND), are accepted if running CMD_BLE5_SCANNER on a primary advertising channel, prior to following an AuxPtr.

The bCrcErr and bIgnore bits are set according to the CRC result and the received message. Only connectable packets are accepted by an initiator. The parameter pParams->initConfig.bUseWhiteList determines if the initiator should try to connect to a specific device or against the whitelist. If this parameter is 0, the whitelist is not used, and pParams->pWhiteList shall point to a buffer containing only the device address of the device to connect to. The address type of the peer shall be given in pParams->advConfig.peerAddrType (see also Section 26.8.17). Otherwise, pParams->pWhiteList shall point to a whitelist, and the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14. The complete check on AdvA is performed as listed in Table 26-142. If the extended header flags indicate that AdvA is not present, the AdvA filter result is always Accept. If the extended header flags indicate that TargetA is present (that is, the packet is directed), the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->initConfig.deviceAddrType, respectively (see also Section 26.8.17), and match if they are equal. Nondirected messages always have TargetA match. Depending on the received packet, the actions taken shall be as given in Table 26-155, where the definition of each action (including the value that will be used on bCrcErr and bIgnore) is given in Table 26-156. The packet length of a received ADV_EXT_IND packet is always valid by default, but it is possible to configure a maximum length by overriding the firmware defined parameter maxAdvExtLen. In addition, the extended header length and flags are checked. If the extended header length is too large for the extended header to fit in the packet, or if it is too small to hold the configured, the length is invalid. If pParams->initConfig.bStrictLenFilter is 1, all defined fields are considered, while if pParams->initConfig.bStrictLenFilter is 0, the fields that are not automatically checked by the CPE (SyncInfo and TxPower) are ignored when finding the minimum allowed extended header length.

Table 26-155. Actions on Received Extended Advertiser Packets by Initiator

PDU Type	CRC Result	AdvMode	AdvA Filter Result	TargetA Match	AuxPtr Present	Action No.
ADV_EXT_IND	OK	01	Reject	X	X	1
ADV_EXT_IND	OK	01	Accept	No	X	1
ADV_EXT_IND	OK	01	Accept	Yes	No	2
ADV_EXT_IND	OK	01	Accept	Yes	Yes	6
ADV_EXT_IND	NOK	01	X	X	X	4
ADV_EXT_IND	X	00, 10, or 11	X	X	X	5
ADV_EXT_IND with invalid lengths	X	X	X	X	X	5
Other (see also Section 26.8.10.1)	X	X	X	X	X	5

Table 26-156. Descriptions of the Actions to Take on Extended Advertiser Packets Received by Initiator

Action No.	bCrcErr	bIgnore	Description
1	0	1	Continue scanning
2	0	0	Continue scanning
4	1	0	Continue scanning
5	—	—	Stop receiving packet, then continue scanning
6	0	0	Follow Aux pointer and receive on the new channel, or end operation with BLE_DONE_AUX

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 6, the radio CPU will either look for a secondary advertising packet or end the operation with status BLE_DONE_AUX.

26.8.11.3 Initiator Receiving Extended Advertising Packets on Secondary Channel

Extended advertising packets on secondary channel (AUX_ADV_IND) are accepted if running CMD_BLE5_INITIATOR on a secondary advertising channel or if following an AuxPtr.

The bCrcErr and bIgnore bits are set according to the CRC result and the received message. Only connectable packets are accepted by an initiator. The parameter pParams->initConfig.bUseWhiteList determines if the initiator should try to connect to a specific device or against the whitelist. If this parameter is 0, the whitelist is not used, and pParams->pWhiteList shall point to a buffer containing only the device address of the device to connect to. The address type of the peer shall be given in pParams->advConfig.peerAddrType (see also Section 26.8.17). Otherwise, pParams->pWhiteList shall point to a whitelist, and the received AdvA field in the message, along with the TxAdd bit of the received header is checked against whitelist as described in Section 26.8.14. The complete check on AdvA is performed as listed in Table 26-142. If the extended header flags indicate that AdvA is not present, the AdvA filter result is always Accept. If the extended header flags indicate that TargetA is present (that is, the packet is directed), the received TargetA field and RxAdd bit are checked against pParams->pDeviceAddress and pParams->initConfig.deviceAddrType, respectively (see also Section 26.8.17), and match if they are equal. Nondirected messages always have TargetA match. Depending on the received packet, the actions taken shall be as given in Table 26-157, where the definition of each action, including the value that will be used on bCrcErr and bIgnore, is given in Table 26-158. The packet length of a received AUX_ADV_IND packet is always valid by default, but it is possible to configure a maximum length by overriding the firmware defined parameter maxAdvExtLen. In addition, the extended header length and flags are checked. If the extended header length is too large for the extended header to fit in the packet, or if it is too small to hold the configured, the length is invalid. If pParams->initConfig.bStrictLenFilter is 1, all defined fields are considered, while if pParams->initConfig.bStrictLenFilter is 0, the fields that are not automatically checked by the CPE (SyncInfo and TxPower) are ignored when finding the minimum allowed extended header length.

Table 26-157. Actions on Received Secondary Channel Packets by Initiator

PDU Type	CRC Result	AdvMode	AdvA filter result	TargetA Match	Action No.
AUX_ADV_IND	OK	01	Reject	X	1
AUX_ADV_IND	OK	01	Accept	No	1
AUX_ADV_IND	OK	01	Accept	Yes	3
AUX_ADV_IND	NOK	01	X	X	4
AUX_ADV_IND	X	00, 10, or 11	X	X	5
AUX_ADV_IND with invalid length	X	X	X	X	5
Other	X	X	X	X	5

Table 26-158. Descriptions of the Actions to Take on Secondary Channel Packets Received by Initiator

Action No.	bCrcErr	bIgnore	Description
1	0	1	End operation with BLE_DONE_RXERR status
3	0	0	Perform backoff procedure and send AUX_CONNECT_REQ and receive AUX_CONNECT_RSP, if applicable; then end operation
4	1	0	End operation with BLE_DONE_RXERR status
5	—	—	Stop receiving packet, then continue scanning

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

If the action from the received packet is 3, an AUX_CONNEXT_REQ is to be transmitted if allowed after a backoff procedure. This procedure starts with decrementing pParams->backoffCount. If this variable becomes 0 after the decrement, an AUX_CONNECT_REQ shall be transmitted. If not, the operation shall end.

When transmitting an AUX_CONNECT_REQ, the radio CPU shall construct this packet. In the header, the PDU Type bits shall be 0101b. The TxAdd bit shall be as in pParams->initConfig.deviceAddrType (if the least significant bit of pParams->pDeviceAddress is 1, the TxAdd bit is inverted, Section 26.8.17). The RxAdd bit shall be as in the TxAdd field of the header of the received AUX_ADV_IND message. The length shall be calculated from the length of the LLData, meaning that it shall be pParams->connectReqLen + 12. The ChSel and RFU bits shall be 0. The payload shall start with the 6-byte device address, which shall be read from pParams->pDeviceAddress, followed by the 6-byte peer address read from the AdvA field of the received message. The rest of the payload shall be read from the pParams->pConnectData buffer. If pParams->initConfig.bDynamicWinOffset is 1, the radio CPU will replace the bytes in the WinSize and WifOffset position with a calculated value (see the Specification of the Bluetooth® System, Version 5.0 listed in Related Documentation).

After an AUX_CONNECT_REQ message is transmitted, the radio CPU shall configure receiver and look for an AUX_CONNECT_RSP. The packet will be received using the rules from Table 26-159 and Table 26-160. The rules in Table 26-159 mean that a packet may be accepted even if it has a different AdvA than the first AUX_ADV_IND packet. To correct this issue, a check could be made in the system CPU, or a patch could be considered.

Table 26-159. Actions on Received AUX_CONNECT_RSP Packets by Initiator

PDU Type	CRC Result	AdvA Filter Result	TargetA Match	Action No.
AUX_CONNECT_RSP	OK	Reject	X	1
AUX_CONNECT_RSP	OK	Accept	No	1
AUX_CONNECT_RSP	OK	Accept	Yes	3
AUX_CONNECT_RSP	NOK	X	X	4
AUX_CONNECT_RSP with invalid length	X	X	X	5
Other	X	X	X	5

Table 26-160. Descriptions of the Actions to Take on Secondary Channel Packets Received by Initiator

Action No.	bCrcErr	bIgnore	Description
1	0	1	End operation with BLE_DONE_RXERR status
3	0	0	End operation with BLE_DONE_CONNECT status
4	1	0	End operation with BLE_DONE_RXERR status
5	—	—	Stop receiving packet, then end operation with BLE_DONE_NOSYNC status

After receiving or attempting to receive an AUX_CONNECT_RSP, the backoff parameters shall be updated by the radio CPU as described in Section 26.8.15, based on the result as given in the *Response Packet Result* column of Table 26-146 and the old values of the backoff parameters.

26.8.11.4 Automatic Window Offset Insertion

If pParams->initConfig.bDynamicWinOffset is 1, the radio CPU shall perform automatic calculation of the WinSize and WinOffset parameters in the transmitted CONNECT_IND or AUX_CONNECT_REQ message. WinSize is byte 7 of the payload, and WinOffset is byte 8 and 9 (see the Bluetooth® Specification documents listed in Related Documentation). The radio CPU shall find the possible start times of the first connection event from the pParams->connectTime parameter and the connection interval, which shall be given in units of 1.25 ms by the Interval field (byte 10 and 11) from the payload to be transmitted (see the Bluetooth® Specification documents listed in Related Documentation). The possible times of the first connection event are any whole multiple of connection intervals from pParams->connectTime, which may be in the past or the future from the start of the initiator command.

The radio CPU shall insert a WinOffset parameter in the transmitted CONNECT_IND or AUX_CONNECT_RSP such that the first connection event is signaled to be at the first connection event that comes sufficiently long after the end of the CONNECT_IND or AUX_CONNEXT_REQ packet to be transmitted. This means that the time from the end of the CONNECT_IND or AUX_CONNECT_REQ packet to the start of the first packet in the connection is between transmitWindowDelay + WinOffset and transmitWindowDelay + WinOffset + WinSize, where transmitWindowDelay is 1.25 ms when a CONNECT_IND PDU is used, 2.5 ms when an AUX_CONNECT_REQ PDU is used on an LE Uncoded PHY, and 3.75 ms when an AUX_CONNECT_REQ PDU is used on the LE Coded PHY, (for details, see the Specification of the Bluetooth® System, Version 5.0 listed in Related Documentation). The radio CPU shall set up the transmit window (WinOffset and WinSize) so that there is margin both between the start of the transmit window and the start of the first master packet and between the start of the first master packet and the end of the transmit window (see the Specification of the Bluetooth® System, Version 5.0 listed in Related Documentation). The inserted WinSize shall be either 1 or 2; ensuring such a margin. The margin is set to the correct value through an override in the BLE stack (see Section 26.8.4).

The radio CPU shall write the calculated values for WinSize and WinOffset into the corresponding locations in the pParams->pConnectData buffer. The start time of the first connection event to be used in order to transmit the first packet within the signaled transmit window shall be written back by the radio CPU in pParams->connectTime. If no connection is made, the radio CPU shall add a multiple of connection intervals to pParams->connectTime so that it is the first possible time of a connection event after the operation ended.

26.8.11.5 End of Initiator Commands

Two triggers to end the operation are set up by pParams->endTrigger/pParams->endTime and pParams->timeoutTrigger/pParams->timeoutTime, respectively. If either of these triggers occurs, the radio operation shall end as soon as possible. If it occurs while waiting for sync on an ADV*_IND packet, the operation shall end immediately. If it occurs at another time, the operation shall continue until the scan would otherwise be resumed, and then end. If the immediate command CMD_STOP (see Section 26.3.3.2.2) is received by the radio CPU, it shall have the same meaning as the end trigger occurring, except that the status code after ending shall be CMD_DONE_STOPPED. The differences between the two triggers are the status and result at the end of the operation. Typically, timeoutTrigger can be used at the end of a scan window, while endTrigger can be used when scanning is to end entirely.

The output structure pOutput contains fields that give information on the command being run. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired. The fields shall be updated by the radio CPU as described in the list that follows. The list also indicates when interrupts shall be raised in the system CPU.

- If a CONNECT_IND or AUX_CONNECT_REQ packet is transmitted, nTxConnectReq (CMD_BLE_INITIATOR) or nTxReq (CMD_BLE5_INITIATOR) is incremented and a Tx_Done interrupt is raised.
- If an AUX_CONNECT_REQ is not transmitted due to the backoff procedure, nBackedOffReq is incremented
- If an *ADV*_IND packet is received with CRC OK and the bIgnore flag cleared, nRxAdvOk is incremented, an Rx_Ok interrupt is raised, and timeStamp is set to a timestamp of the start of the packet.
- If an *ADV*_IND packet is received with CRC OK and the bIgnore flag set, nRxAdvIgnored is incremented and an Rx_Ignored interrupt is raised.
- If an *ADV*_IND packet is received with CRC error, nRxAdvNok is incremented and an Rx_Nok interrupt is raised.
- If an *ADV*_IND is received and did not fit in the RX queue, nRxAdvBufFull is incremented and an Rx_Buf_Full interrupt is raised.
- If an AUX_CONNECT_RSP packet is received with CRC OK and the bIgnore flag cleared, nRxRspOk is incremented and an Rx_Ok interrupt is raised.
- If an AUX_CONNECT_RSP packet is received with CRC OK and the bIgnore flag set, nRxRspIgnored is incremented and an Rx_Ignored interrupt is raised.
- If an AUX_CONNECT_RSP packet is received with CRC error, nRxRspNok is incremented and an Rx_Nok interrupt is raised.
- If an AUX_SCAN_RSP packet is received and did not fit in the RX queue, nRxRspBufFull is incremented and an Rx_Buf_Full interrupt is raised.
- If a packet is received, lastRssi is set to the RSSI of that packet.
- If the first RX data entry in the RX queue changed state to Finished after a packet was received, an Rx_Entry_Done interrupt is raised.

An initiator operation shall end with one of the statuses listed in Table 26-161. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action, as defined in Section 26.3.2.5.2.

Table 26-161. End of Initiator Operation

Condition	Status Code	Result
Performed action 3 after receiving a legacy packet (transmitted CONNECT_IND) if pParams->advConfig.chSel = 0 or the ChSel bit of the received ADV_IND or ADV_DIRECT_IND was 1	BLE_DONE_CONNECT	FALSE
Performed action 3 after receiving a legacy packet (transmitted CONNECT_IND) if pParams->advConfig.chSel = 1 and the ChSel bit of the received ADV_IND or ADV_DIRECT_IND was 0	BLE_DONE_CONNECT_CHSEL0	FALSE
Correctly received AUX_CONNECT_RSP after sending AUX_CONNECT_REQ	BLE_DONE_CONNECT	FALSE
Received connectable ADV_EXT_IND packet not to be ignored without AUX pointer	BLE_DONE_OK	TRUE
Performed action 3 on secondary channel, and did not send AUX_CONNECT_REQ due to backoff	BLE_DONE_OK	TRUE
Performed action 1 or action 4 on secondary channel	BLE_DONE_RXERR	TRUE
Performed action 3 on secondary channel, sent AUX_CONNECT_REQ and received AUX_CONNECT_RSP with bCrcErr = 1 or bIgnore = 1	BLE_DONE_RXERR	TRUE
Performed action 3 on secondary channel, sent AUX_CONNECT_REQ, but did not get sync or found wrong packet type or invalid length	BLE_DONE_NOSYNC	TRUE
Observed trigger indicated by pParams->timeoutTrigger while waiting for sync on advertiser packet	BLE_DONE_RXTIMEOUT	TRUE
Observed trigger indicated by pParams->timeoutTrigger, then performed actions 1, 2, 3, 4, or 5.	BLE_DONE_RXTIMEOUT	TRUE

Table 26-161. End of Initiator Operation (continued)

Condition	Status Code	Result
Observed trigger indicated by pParams->endTrigger while waiting for sync on advertiser packet	BLE_DONE_ENDED	FALSE
Observed trigger indicated by pParams->endTrigger, then performed actions 1, 2, 3, 4, 5, or 6.	BLE_DONE_ENDED	FALSE
Observed CMD_STOP while waiting for sync on advertiser packet	BLE_DONE_STOPPED	FALSE
Observed CMD_STOP, then performed actions 1, 2, 3, 4, 5, or 6.	BLE_DONE_STOPPED	FALSE
Performed action 6 with wait time to AUX packet more than pParams->maxWaitTimeForAuxCh.	BLE_DONE_AUX	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT
LLData length field has illegal value	BLE_ERROR_PAR	ABORT

26.8.12 Generic Receiver Command

The generic receiver command may be used to receive physical layer test packets or to receive any packet, such as in a packet sniffer application.

A generic receiver operation is started by a CMD_BLE_GENERIC_RX or CMD_BLE5_GENERIC_RX command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-99 and a pOutput parameter of the type defined in Table 26-112. At the start of a generic receiver operation, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. For CMD_BLE5_GENERIC_RX, it shall also set up the PHY mode given in phyMode.mainMode. The radio CPU shall set up the access address defined in pParams->accessAddress and use the CRC initialization value defined in pParams->crcInit. The whitener shall be set up as defined in the whitening parameter. The radio CPU shall then configure receiver.

In a generic receiver operation, the only assumption made on the packet format is that the second received byte is a length field that indicates the length of the payload following that byte, and that a standard BLE-type CRC is appended to the packet. The number of bits in the length field is assumed to be the same as for an advertising packet, that is by default 8 bits if configured using CMD_BLE5_RADIO_SETUP and 6 bits otherwise.

After tuning to the correct channel, the radio CPU shall start listening for a packet. If sync is obtained on the demodulator, the message is received into the RX queue (if any). If the length is greater than the maximum allowed length for BLE legacy advertising packets (37, but can be overridden), reception shall be stopped and restarted.

If pParams->pRxQ is NULL, the received packets shall not be stored. The counters shall still be updated and interrupts generated.

If a packet is received with CRC error, the bCrcErr bit shall be set. The bIgnored flag shall never been set for the generic RX command.

If the packet being received did not fit in the RX queue, the packet shall be received to the end, but the received bytes are not stored. If the packet would normally not have been discarded from the RX buffer, the operation shall end.

A trigger to end the operation is set up by pParams->endTrigger and pParams->endTime. If the trigger that is defined by this parameter occurs, the radio operation shall end as soon as possible. If it occurs while waiting for sync, the operation shall end immediately. If it occurs at another time, the operation shall continue until the current packet is fully received, and then end. If the immediate command CMD_STOP (see Section 26.3.3.2.2) is received by the radio CPU, it shall have the same meaning as the end trigger occurring, except that the status code after ending shall be CMD_DONE_STOPPED.

The output structure pOutput contains fields that give information on the command being run. The radio CPU shall not initialize the fields, so this must be done by the system CPU when a reset of the counters is desired.

The fields shall be updated by the radio CPU as described in the following list. The list also indicates when interrupts shall be raised in the system CPU.

- If a packet is received with CRC OK, nRxOk is incremented and an Rx_Ok interrupt is raised.
- If a packet is received with CRC error, nRxNok is incremented and an Rx_Nok interrupt is raised.
- If a packet is received and did not fit in the RX queue, nRxBufFull is incremented and an Rx_Buf_Full interrupt is raised.
- If a packet is received, lastRssi is set to the RSSI of that packet.
- If a packet is received, timeStamp is set to a timestamp of the start of that packet.
- If the first RX data entry in the RX queue changed state to Finished after a packet was received, an Rx_Entry_Done interrupt is raised.

After a packet is received, reception shall be restarted on the same channel if pParams->bRepeat = 1, the end event has not been observed, and the packet fit in the receive queue. If pParams->bRepeat = 0, the operation shall always end after receiving a packet.

A generic RX operation shall end with one of the statuses listed in Table 26-162. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT, is indicated, which will decide the next action, as defined in Section 26.3.2.5.2. The pNextOp field of a generic RX command structure may point to the same command structure. That way, RX may be performed until the end trigger or until the RX buffer goes full.

Table 26-162. End of Generic RX Operation

Condition	Status Code	Result
Received a packet with CRC OK and pParams->bRepeat = 0	BLE_DONE_OK	TRUE
Received a packet with CRC error and pParams->bRepeat = 0	BLE_DONE_RXERR	TRUE
Observed trigger indicated by pParams->endTrigger while waiting for sync	BLE_DONE_ENDED	FALSE
Observed trigger indicated by pParams->endTrigger, then finished receiving packet	BLE_DONE_ENDED	FALSE
Observed CMD_STOP while waiting for sync	BLE_DONE_STOPPED	FALSE
Observed CMD_STOP, then finished receiving packet	BLE_DONE_STOPPED	FALSE
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
No space in RX buffer to store received packet	BLE_ERROR_RXBUF	FALSE
Illegal value of channel	BLE_ERROR_PAR	ABORT

26.8.13 PHY Test Transmit Command

The test packet transmitter command may be used to transmit physical layer test packets.

A test packet transmitter operation is started by a CMD_BLE_TX_TEST or CMD_BLE5_TX_TEST command. In the command structure, it shall have a pParams parameter of the type defined in Table 26-100 and a pOutput parameter of the type defined in Table 26-113. At the start of a test TX operation, the radio CPU shall wait for the start trigger. It shall then program the frequency based on the channel parameter of the command structure. For CMD_BLE5_TX_TEST, it shall also set up the PHY mode given in phyMode.mainMode. The radio CPU shall set up the test packet synchronization word (see the Bluetooth® Specification documents listed in Related Documentation) and use the CRC initialization value 0x555555. The whitener shall be set up as defined in the whitening parameter. To produce PHY test packets conforming to the Specification of the Bluetooth® System, Version 5.0 document listed in Related Documentation), the whitener should be disabled.

The radio CPU shall transmit pParams->numPackets packets and then end the operation. If pParams->numPackets is 0, transmission shall continue until the operation ends for another reason (timeout, stop, or abort command). The time (number of radio timer ticks) between the start of each packet shall be given by pParams->period. If this time is smaller than the duration of a packet, each packet shall be transmitted as soon as possible. Each packet shall be assembled as follows by the radio CPU. The first byte is a header byte. It shall contain the value of pParams->packetType, provided this one of the values listed in Table 26-163. The next byte is the length byte, which shall be the value of pParams->payloadLength. This shall be followed by a

number of payload bytes that shall be as listed in Table 26-163. The number of payload bytes shall be equal to pParams->payloadLength. If pParams->packetType is 0, the bytes shall be from the PRBS9 sequence defined in Specification of the Bluetooth® System documents listed in Related Documentation. Otherwise, all the bytes shall be the same, as listed in Table 26-163. A 3-byte CRC, according to the Bluetooth® low energy specification listed in Related Documentation, shall be appended.

Table 26-163. Supported PHY Test Packet Types

Value of packetType	Transmitted Bytes
0	PRBS9 sequence
1	Repeated 0x0F
2	Repeated 0x55
3	PRBS15 sequence
4	Repeated 0xFF
5	Repeated 0x00
6	Repeated 0xF0
7	Repeated 0xAA

The PRBS15 payload type defined in the Bluetooth® Specification documents listed in Related Documentation, which corresponds to payload type 3, shall be implemented using the polynomial x15 + x14 + 1. The initialization shall be taken from the radio timer for the first packet transmitted and not re-initialized for subsequent packets.

If pParams->config.overrideDefault is 1, the packet shall be nonstandard. The header shall contain the value given in pParams->packetType, and each byte transmitted shall be as given in pParams->byteVal. If pParams->config.bUsePrbs9 is 1, the sequence shall be generated through an XOR operation where each byte of the PRBS9 sequence used for packet type 0 with pParams->byteVal. If pParams->config.bUsePrbs15 is 1, the sequence shall be generated through an XOR operation where each byte of the PRBS15 sequence used for packet type 3 with pParams->byteVal.

If either of the PRBS sequences is used, whitening will be disabled regardless of the setting in the whitening parameter.

A trigger to end the operation is set up by pParams->endTrigger and pParams->endTime. If the trigger that is defined by this parameter occurs, the radio operation shall end as soon as possible. If it occurs while waiting between packets, the operation shall end immediately. If it occurs at another time, the operation shall continue until the current packet is fully transmitted, and then end. If the immediate command CMD_STOP (see Section 26.3.3.2.2) is received by the radio CPU, it shall have the same meaning as the end trigger occurring, except that the status code after ending shall be CMD_DONE_STOPPED.

The output structure pOutput contains only the field nTx. The nTx field shall be incremented each time a packet is transmitted. The radio CPU shall not initialize the field, so this must be done by the system CPU when a reset of the counters is desired. A Tx_Done interrupt shall be raised each time a packet is transmitted.

A PHY test TX operation will end with one of the statuses listed in Table 26-164. The status field of the command structure after the operation is ended indicates the reason why the operation ended. In all cases, a Command_Done interrupt is raised. In each case, the result of TRUE, FALSE, or ABORT is indicated, which will decide the next action, as defined in Section 26.3.3.2.2.

Table 26-164. End of PHY Test TX Operation

Condition	Status Code	Result
Transmitted pParams->numPackets packets	BLE_DONE_OK	TRUE
Observed trigger indicated by pParams->endTrigger while waiting between packets	BLE_DONE_ENDED	FALSE
Observed trigger indicated by pParams->endTrigger, then finished transmitting packet	BLE_DONE_ENDED	FALSE
Observed CMD_STOP while waiting between packets	BLE_DONE_STOPPED	FALSE
Observed CMD_STOP, then finished transmitting packet	BLE_DONE_STOPPED	FALSE

Table 26-164. End of PHY Test TX Operation (continued)

Condition	Status Code	Result
Received CMD_ABORT	BLE_DONE_ABORT	ABORT
Illegal value of channel	BLE_ERROR_PAR	ABORT
Illegal value of pParams->packetType	BLE_ERROR_PAR	ABORT

26.8.14 Whitelist Processing

A whitelist is used in advertiser, scanner, and initiator operation. The whitelist consists of a configurable number of entries. The whitelist is an array of entries of the type defined in Table 26-116. The first entry of the array shall contain the array size in the size field.

The minimum number of entries in a whitelist array is 1, but if no whitelist is to be used, pParams->pWhiteList may be NULL. The maximum number of entries is limited by the performance obtained in the radio CPU for doing the filtering. Testing indicates that up to at least 24 entries can be used.

Each entry contains one address and three configuration bits. The bEnable bit is 1 if the entry is enabled, otherwise the address shall be ignored when doing whitelist filtering. The addrType bit tells if the entry is a public or random address. The bIgnore bit can be used by a scanner to avoid reporting and scanning the same device multiple times.

When an address is to be checked against the whitelist, the address is compared against the address field of each entry in the whitelist. The address is said to be present in the whitelist if and only if there is an entry where:

- bEnable is 1
- addrType is equal to the address type of the address to check
- All bytes of the address array are equal to the bytes of the address to check
- For scanner only: bWIIgn is 0
- For initiator only: bIrkValid is 0

For scanners, the bWIIgn bit may be set in the whitelist to indicate that a device shall be ignored even if the whitelist entry would otherwise be a match. This can be used to check for advertisers that have already been scanned or where the advertising data has already been reported. Even if no whitelist filtering is to be performed, this feature may be used. The whitelist shall be scanned for devices that match the address and address type, and where bWIIgn is 1. Such devices shall be ignored. The bEnable bit shall not be checked in this case. The check is always done when receiving legacy packets, while for extended advertiser packets, the check is only done when pParams->extFilterConfig.bApplyDuplicateFiltering is 1 and no ADI field is present in the extended header. It is possible to configure the radio CPU to automatically set the bWIIgn bit, see Section 26.8.10.

For initiators, the bIrkValid bit may be set in the whitelist to indicate that a device shall be ignored even if the whitelist entry would otherwise be a match. This can be used to avoid connecting to a device with a resolvable private address where a valid identity resolving key (IRK) exists. The bIrkValid bit should never be set by the system CPU for address entries not containing an RPA.

26.8.15 Backoff Procedure

After receiving or attempting to receive a SCAN_RSP, AUX_SCAN_RSP or AUX_CONNECT_RSP packet, the backoff parameters shall be updated by the radio CPU. The update depends on the Response Packet Result and the old values of the backoff parameters. The backoff parameters given in pParams->backoffPar shall be updated as shown in Table 26-165. After this update, the radio CPU shall set pParams->backoffCount to a pseudo-random number between 1 and $2^{\text{pParams->backoffPar.logUpperLimit}}$.

Table 26-165. Update of Backoff Parameters

Response Packet Result	Old pParams->backoffPar		New pParams->backoffPar		
	bLastSucceeded	bLastFailed	bLastSucceeded	bLastFailed	logUpperLimit
Failure	X	0	0	1	logUpperLimit
Failure	0	1	0	0	min(logUpperLimit+1, 8)

Table 26-165. Update of Backoff Parameters (continued)

Response Packet Result	Old pParams->backoffPar		New pParams->backoffPar		
	bLastSucceeded	bLastFailed	bLastSucceeded	bLastFailed	logUpperLimit
Success	0	X	1	0	logUpperLimit
Success	1	0	0	0	max(logUpperLimit-1, 0)

The pseudo-random algorithm shall be based on a maximum-length 16-bit linear feedback shift register (LFSR). The seed shall be as provided in pParams->randomState. When the operation ends, the radio CPU shall write the current state back to this field. If pParams->randomState is 0, the radio CPU shall self-seed by initializing the LFSR to the 16 least significant bits of the radio timer. This shall only be done when the LFSR is first needed (that is, after receiving an ADV*_IND), so there will be some randomness to this value. If the 16 least significant bits of the radio timer are all 0, another fixed value shall be substituted.

When the device enters the scanning or initiating (Bluetooth® 5 only) state, the system CPU should initialize pParams->backoffCount to 1, pParams->backoffPar.logUpperLimit to 0, pParams->backoffPar.bLastSucceeded and pParams->backoffPar.bLastFailed to 0, and pParams->randomState to a true-random value (or a pseudo-random number based on a true-random seed). When starting new scanner operations while remaining in the scanning state, the system CPU should keep pParams->randomState, pParams->backoffCount, and pParams->backoffPar at the values they had at the end of the last scanner operation.

26.8.16 AUX Pointer Processing

If an extended advertiser packet that contains an Aux pointer is received, the radio CPU shall follow the Aux pointer or return information on how the Aux pointer can be followed in a subsequent command. An Aux pointer is not followed in any of the following cases:

- The packet has CRC error
- The packet is ignored
- The receive buffer is full
- The packet is a scannable or connectable AUX_ADV_IND packet or an AUX_CONNECT_RSP packet (these packets shall not have an Aux pointer)

If the Aux pointer is followed, the AUX Offset is read and derived values are stored in the parameter structure. pParams->rxStartTime is set to the time that the receiver should be started in order to receive the packet, taking into account the length of the received packet, clock inaccuracies, and receiver startup time for the PHY to use for the next packet. pParams->rxListenTime is set to the time from pParams->rxStartTime to timeout of the receive operation. pParams->channelNo is set to the secondary channel pointed to, and pParams->phyMode indicates the PHY to be used.

If the time from the end of the packet containing the Aux pointer to the time given in pParams->rxStartTime is more than pParams->maxWaitTimeForAuxCh, the operation will end after the packet is received with status BLE_DONE_AUX. It is then up to the system CPU to start a new scanner or initiator command on a secondary channel at the specified time, usually after having been in low-power mode. This command should be started with an absolute start time of pParams->rxStartTime - startToSynthRatOffset, where startToSynthRatOffset is a firmware defined parameter. The command should have a timeout trigger at the time pParams->rxStartTime + pParams->rxListenTime.

If the time given in pParams->rxStartTime is earlier than this, the radio CPU shall automatically schedule the channel and PHY switch. The receiver will be stopped after receiving the packet with the Aux pointer and restarted at pParams->rxStartTime, after the synthesizer is reprogrammed and the PHY mode switched. If the time from the end of the received packet to pParams->rxStartTime is too small, the receiver will start as early as possible. The receiver will run as described in Section 26.8.10.3 for a scanner or Section 26.8.11.3 for an initiator. It will have a timeout after pParams->rxListenTime, and if timed out the operation ends with status BLE_DONE_RXTIMEOUT.

26.8.17 Dynamic Change of Device Address

For advertiser, scanner and initiator commands, it is possible to change the device address while a command is running. If a new pointer is written to the pDeviceAddress or pWhiteList field of an advertiser, scanner or initiator parameter structure, this pointer is taken into use for the device address or peer address at the following time:

- For advertisers, the next time an advertiser command (in a command chain) is started. For a command already running, the address is not updated.
- For scanners and initiators, the next time sync is found on an advertiser packet.

Until the new pointer is taken into use, the previous pointer will be used for the device address.

The advConfig, scanConfig, and initConfig fields are re-read at the same time as the device address pointer. However, it is not recommended to change the device address type or peer address type in these fields while a command or command chain is running, as there is a small probability that the address type could be out of sync with the address. Instead, the least significant bit of pDeviceAddress can be set to 1 to indicate that the device address type should be inverted compared to the type found in advConfig, scanConfig, or initConfig. Similarly, the least significant bit of pWhiteList can be set to 1 to indicate that the peer address type should be inverted compared to the type found in advConfig or initConfig. This means that the address and address type may be modified in an atomic operation. The least significant bit of pDeviceAddress and pWhiteList is ignored when finding the address of the array that holds the address; the radio CPU will always read from a half-word-aligned address.

Note

The contents of the array pointed to by pDeviceAddress or pWhiteList (when pointing to a single address) should not be modified while a command is running. If the pointer is updated in the command structure, the array pointed to by the old value of pDeviceAddress or pWhiteList should not be modified until it can be known that the address pointed to is no longer in use by the radio CPU.

26.9 Immediate Commands

In addition to the immediate commands otherwise available, the immediate command *Update Advertising Payload Command* shall be supported (for more details, see Section 26.9.1).

26.9.1 Update Advertising Payload Command

The CMD_BLE_ADV_PAYLOAD command can be used to change the payload buffer for a legacy advertising command. It may be issued regardless of whether an advertising command is running or not.

The command structure shall have the format given in Table 26-93. When received, the radio CPU shall check if an advertiser radio operation command is running using the parameter structure given in pParams of the immediate command structure. If not, the radio CPU shall update the parameter structure immediately. If a radio operation command is running using the parameter structure to be updated, the radio CPU shall only modify the parameter structure if the payload to be changed is not currently being transmitted. If it is being transmitted, the radio CPU shall store the request and update as soon as transmission of the packet has finished.

When updating the parameter structure, the payload to change depends on the payloadType parameter of the command structure. If payloadType is 0, the radio CPU shall set pParams->advLen equal to newLen and pParams->pAdvData equal to newData. If payloadType is 1, the radio CPU shall set pParams->scanRspLen equal to newLen and pParams->pScanRspData equal to newData. After the update has taken place, the radio CPU shall raise a Tx_Buffer_Changed interrupt, see Table 26-122. This interrupt shall be raised regardless of whether the update was delayed or not.

If any of the parameters are illegal, the radio CPU shall respond with ParError in CMDSTAT and not perform any update. Otherwise, the radio CPU shall respond with Done in CMDSTAT. This may be done before the update has taken place.

This command is provided to provide an atomic change of the payload and length. For the Bluetooth® 5 advertiser command, this is not needed. The pointers pAdvPkt and pRspPkt may safely be changed at any time; however, the packet entry pointed to must not be modified until the command has ended.

26.10 Proprietary Radio

This section describes proprietary radio command structure, data handling, radio operations commands, and immediate commands. The commands define a flexible packet handling compatible with the CC110x, CC111x, CC112x, CC120x, CC2500, and CC251x devices, as well as supporting other legacy modes.

26.10.1 Packet Formats

For compatibility with existing TI parts, the packet format given in Figure 26-9 can be used in most cases. This packet format is supported through the use of the commands CMD_PROP_TX and CMD_PROP_RX.

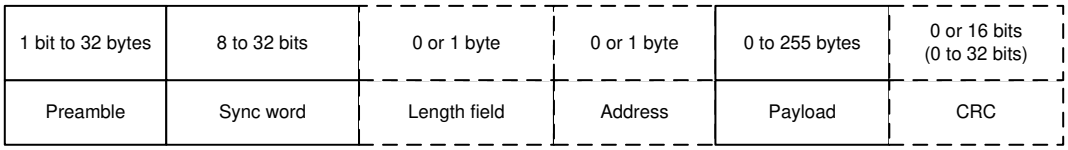


Figure 26-9. Standard Packet Format

A more flexible packet format is also possible, as defined in Figure 26-10. This format is supported by the commands CMD_PROP_RX_ADV and CMD_PROP_TX_ADV. The format in Figure 26-9 is an example of this format.

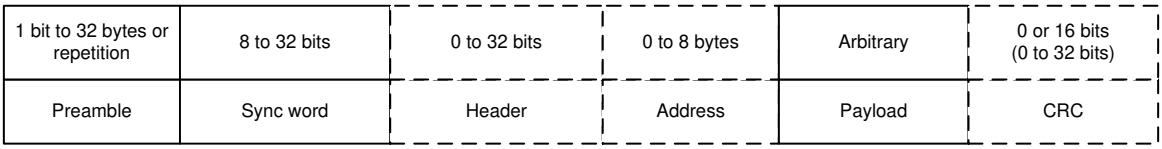


Figure 26-10. Advanced Packet Format

26.10.2 Commands

Table 26-166 defines the proprietary radio operation commands.

Table 26-166. Proprietary Radio Operation Commands

ID	Command Name	Description
0x3801	CMD_PROP_TX	Transmit packet
0x3802	CMD_PROP_RX	Receive packet or packets
0x3803	CMD_PROP_TX_ADV	Transmit packet with advanced modes
0x3804	CMD_PROP_RX_ADV	Receive packet or packets with advanced modes
0x3805	CMD_PROP_CS	Run carrier sense command
0x3806	CMD_PROP_RADIO_SETUP	Set up radio in proprietary mode
0x3807	CMD_PROP_RADIO_DIV_SETUP	Set up radio in proprietary mode
0x3808	CMD_PROP_RX_SNIFF	Receive packet or packets with sniff mode support
0x3809	CMD_PROP_RX_ADV_SNIFF	Receive packet or packets with advanced modes and sniff mode support

Table 26-167 defines the proprietary immediate commands.

Table 26-167. Proprietary Immediate Commands

ID	Command Name	Description
0x3401	CMD_PROP_SET_LEN	Set length of packet being received
0x3402	CMD_PROP_RESTART_RX	Stop receiving a packet and go back to sync search

26.10.2.1 Command Data Definitions

This section defines data types used in describing the data structures used for communication between the system CPU and the radio CPU. The data structures are listed with tables. The Byte Index is the offset from the pointer to that structure. Multibyte fields are little endian, and halfword or word alignment is required. For bit numbering, 0 is the LSB. The R/W column is used as follows:

R: The system CPU can read a result back; the radio CPU does not read the field.

W: The system CPU writes a value, the radio CPU reads it and does not modify the value.

R/W: The system CPU writes an initial value, the radio CPU may modify the initial value.

26.10.2.1.1 Command Structures

For all the radio operation commands, the first 14 bytes are as defined in Table 26-9.

Table 26-168. CMD_PROP_TX Command Structure

Byte Index	Field Name	Bits	Bit Field name	Type	Description
14	pktConf	0	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		1–2			Reserved
		3	bUseCrc	W	0: Do not append CRC. 1: Append CRC.
		4	bVarLen	W	0: Fixed length 1: Transmit length as first byte
		5–7			Reserved
15	pktLen			W	Packet length
16–19	syncWord			W	Sync word to transmit
20–23	pPkt			W	Pointer to packet

Table 26-169. CMD_PROP_TX_ADV Command Structure

Byte Index	Field Name	Bits	Bit Field name	Type	Description
14	pktConf	0	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		1–2			Reserved
		3	bUseCrc	W	0: Do not append CRC. 1: Append CRC.
		4	bCrcIncSw	W	0: Do not include sync word in CRC calculation. 1: Include sync word in CRC calculation.
		5	bCrcIncHdr	W	0: Do not include header in CRC calculation. 1: Include header in CRC calculation.
		6–7			Reserved
15	numHdrBits			W	Number of bits in header (0–32)
16–17	pktLen			W	Packet length. 0: Unlimited
18	startConf	0	bExtTxTrig	W	0: Start packet on a fixed time from the command start trigger. 1: Start packet on an external trigger (Contact TI to enable this feature).
		1–2	inputMode	W	Input mode if external trigger is used for TX start. 00: Rising edge 01: Falling edge 10: Both edges 11: Reserved
		3–7	source	W	RAT input event number used for capture if external trigger is used for TX start.

Table 26-169. CMD_PROP_TX_ADV Command Structure (continued)

Byte Index	Field Name	Bits	Bit Field name	Type	Description
19	preTrigger			W	Trigger for transition from preamble to sync word. If this is set to “now”, one preamble as configured in the setup is sent. Otherwise, the preamble is repeated until this trigger is observed.
20–23	preTime			W	Time parameter for preTrigger
24–27	syncWord			W	Sync word to transmit
28–31	pPkt			W	Pointer to packet, or TX queue for unlimited length

Table 26-170. CMD_PROP_RX and CMD_PROP_RX_SNIFF Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	pktConf	0	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		1	bRepeatOk	W	0: End operation after receiving a packet correctly. 1: Go back to sync search after receiving a packet correctly.
		2	bRepeatNok	W	0: End operation after receiving a packet with CRC error. 1: Go back to sync search after receiving a packet with CRC error.
		3	bUseCrc	W	0: Do not check CRC. 1: Check CRC.
		4	bVarLen	W	0: Fixed length 1: Receive length as first byte.
		5	bChkAddress	W	0: No address check. 1: Check address.
		6	endType	W	0: Packet is received to the end if end trigger happens after sync is obtained. 1: Packet reception is stopped if end trigger happens.
		7	filterOp	W	0: Stop receiver and restart sync search on address mismatch. 1: Receive packet and mark it as ignored on address mismatch.
15	rxConf			W	RX configuration, see Table 26-177 for details
16–19	syncWord			W	Sync word to listen for
20	maxPktLen			W	Packet length for fixed length, maximum packet length for variable length 0: Unlimited or unknown length
21	address0			W	Address
22	address1			W	Address (set equal to address0 to accept only one address. If 0xFF, accept 0x00 as well)
23	endTrigger			W	Trigger classifier for ending the operation
24–27	endTime			W	Time to end the operation
28–31	pQueue			W	Pointer to receive queue
32–35	pOutput			W	Pointer to output structure
36–47	CMD_PROP_RX_SNIFF only: carrier sense options as given in Table 26-176				

Table 26-171. CMD_PROP_RX_ADV and CMD_PROP_RX_ADV_SNIFF Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	pktConf	0	bFsOff	W	0: Keep frequency synthesizer on after command. 1: Turn frequency synthesizer off after command.
		1	bRepeatOk	W	0: End operation after receiving a packet correctly. 1: Go back to sync search after receiving a packet correctly.
		2	bRepeatNok	W	0: End operation after receiving a packet with CRC error. 1: Go back to sync search after receiving a packet with CRC error.
		3	bUseCrc	W	0: Do not check CRC. 1: Check CRC.
		4	bCrcIncSw	W	0: Do not include sync word in CRC calculation. 1: Include sync word in CRC calculation.
		5	bCrcIncHdr	W	0: Do not include header in CRC calculation. 1: Include header in CRC calculation.
		6	endType	W	0: Packet is received to the end if end trigger happens after sync is obtained. 1: Packet reception is stopped if end trigger happens.
		7	filterOp	W	0: Stop receiver and restart sync search on address mismatch. 1: Receive packet and mark it as ignored on address mismatch.
15	rxConf			W	RX configuration (for details, see Table 26-177)
16–19	syncWord0			W	Sync word to listen for
20–23	syncWord1			W	Alternative sync word if nonzero
24–25	maxPktLen			W	Maximum length of received packets 0: Unlimited or unknown length
26–27	hdrConf	0–5	numHdrBits	W	Number of bits in header (0–32)
		6–10	lenPos	W	Position of length field in header (0–31)
		11–15	numLenBits	W	Number of bits in length field (0–16)
28–29	addrConf	0	addrType	W	0: Address after header
					1: Address in header
		1–5	addrSize	W	If addrType = 0: Address size in bytes. If addrType = 1: Address size in bits.
		6–10	addrPos	W	If addrType = 1: Bit position of address in header If addrType = 0: nonzero to extend address with sync word identifier
		11–15	numAddr	W	Number of addresses in address list
30	lenOffset			W	Signed value to add to length field
31	endTrigger			W	Trigger classifier for ending the operation
32–35	endTime			W	Time to end the operation
36–39	pAddr			W	Pointer to address list
40–43	pQueue			W	Pointer to receive queue
44–47	pOutput			W	Pointer to output structure
48–59	CMD_PROP_RX_ADV_SNIFF only: carrier sense options as given in Table 26-176				

Table 26-172. CMD_PROP_CS Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14	csFsConf	0	bFsOffIdle	W	0: Keep synthesizer running if command ends with channel Idle. 1: Turn off synthesizer if command ends with channel Idle.
		1	bFsOffBusy	W	0: Keep synthesizer running if command ends with channel Busy. 1: Turn off synthesizer if command ends with channel Busy.
15					Reserved

Table 26-172. CMD_PROP_CS Command Structure (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
16–27	Carrier sense options as given in Table 26-176.				

Table 26-173. CMD_PROP_RADIO_SETUP and CMD_PROP_RADIO_DIV_SETUP Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	modulation	0–2	modType	W	0: FSK 1: GFSK 7: O-QPSK Others: Reserved
		3–13	deviation	W	Deviation (step size given by deviationStepSz)
		14–15	deviationStepSz		Deviation step size: 0: 250 Hz 1: 1000 Hz 2: 15.625 Hz 3: 62.5 Hz
16–19	symbolRate	0–7	preScale	W	Prescaler value (see Section 26.10.5.2)
		8–28	rateWord	W	Rate word (see Section 26.10.5.2)
		29–31			Reserved, set to 0
20	rxBw			W	Receiver bandwidth (see Table 26-181) 1–18: Legacy mode (bandwidth 87–4236 kHz) 32–52: Normal mode (bandwidth 45–4236 kHz) 64–103: Enhanced mode (bandwidth 4.8–4236 kHz)
21	preamConf	0–5	nPreamBytes	W	0: 1 preamble bit 1–16: Number of preamble bytes 18, 20, ..., 30: Number of preamble bytes 31: 4 preamble bits 32: 32 preamble bytes Others: Reserved
		6–7	preamMode	W	00: Send 0 as the first preamble bit 01: Send 1 as the first preamble bit 10: Send same first bit in preamble and sync word 11: Send different first bit in preamble and sync word
22–23	formatConf	0–5	nSwBits	W	Number of sync word bits. Valid values are from 8–32
		6	bBitReversal	W	0: Use positive deviation for 1 1: Use positive deviation for 0
		7	bMsbFirst	W	0: Least significant bit transmitted first 1: Most significant bit transmitted first
		8–11	fecMode	W	Select Coding 0000: Uncoded binary modulation 1000: Long range mode 1010: Manchester coded binary modulation (FSK/GFSK) Others: Reserved
		12			Reserved
		13–15	whitenMode	W	000: No whitening 001: CC1101 and CC2500 compatible whitening 010: PN9 whitening without byte reversal 011: Reserved 100: No whitener, 32-bit IEEE 802.15.4g compatible CRC 101: IEEE 802.15.4g compatible whitener and 32-bit CRC 110: No whitener, dynamically IEEE 802.15.4g compatible 16-bit or 32-bit CRC 111: Dynamically IEEE 802.15.4g compatible whitener and 16-bit or 32-bit CRC

Table 26-173. CMD_PROP_RADIO_SETUP and CMD_PROP_RADIO_DIV_SETUP Command Structure (continued)

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
24–25	config	0–2	Reserved	W	
		3	Reserved	W	
		4-9	Reserved	W	
		10	Reserved	W	
		11	Reserved	W	
		12	Reserved	W	
		13–15	Reserved		
26–27	txPower			W	Output power setting, use value from Smart RF™ Studio or SysConfig. See Section 26.3.3.2.16 for more details. 0xFFFF: Use 20 dBm PA ("P" devices only)
28–31	pRegOverride			W	Pointer to a list of hardware and configuration registers to override. If NULL, no override is used.
32–33	centerFreq			W	CMD_PROP_RADIO_DIV_SETUP only: Center frequency of the band. To be used in the initial parameter computations.
34–35	intFreq			W	CMD_PROP_RADIO_DIV_SETUP only: Intermediate frequency to use for RX, in MHz on 4.12 signed format. TX will use same intermediate frequency if supported, otherwise 0. 0x8000: Use default
36	loDivider			W	CMD_PROP_RADIO_DIV_SETUP only: Divider setting to use. See the Smart RF Studio for the recommended settings per device and band.

Table 26-174. CMD_PROP_SET_LEN Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0–1	RXLen			W	Payload length to use

26.10.2.2 Output Structures

Table 26-175. Receive Commands

Byte Index	Field Name	Type	Description
0–1	nRxOk	R/W	Number of packets that have been received with payload, CRC OK and not ignored
2–3	nRxNok	R/W	Number of packets that have been received with CRC error
4	nRxIgnored	R/W	Number of packets that have been received with CRC OK and ignored due to address mismatch
5	nRxStopped	R/W	Number of packets not received due to illegal length or address mismatch with pktConf.filterOp = 1
6	nRxBufFull	R/W	Number of packets that have been received and discarded due to lack of buffer space
7	lastRssi	R	RSSI of last received packet. RSSI is captured when sync word is found.
8–11	timeStamp	R	Timestamp of last received packet

26.10.2.3 Other Structures and Bit Fields

Table 26-176. Carrier Sense Fields for CMD_PROP_RX_SNIFF, CMD_PROP_RX_ADV_SNIFF, and CMD_PROP_CS

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
0	csConf	0	bEnaRssi	W	If 1, enable RSSI as a criterion.
		1	bEnaCorr	W	If 1, enable correlation as a criterion.
		2	operation	W	0: Busy if either RSSI or correlation indicates BUSY. 1: Busy if both RSSI and correlation indicates BUSY.
		3	busyOp	W	0: Continue carrier sense on channel BUSY. 1: End carrier sense on channel BUSY. For an RX command, the receiver continues when carrier sense ends, then it does not end if the channel goes IDLE.
		4	idleOp	W	0: Continue on channel IDLE. 1: End on channel IDLE.
		5	timeoutRes	W	0: Timeout with channel state Invalid treated as BUSY. 1: Timeout with channel state Invalid treated as IDLE.
1	rssiThr			W	RSSI threshold
2	numRssiIdle			W	Number of consecutive RSSI measurements below the threshold needed before the channel is declared IDLE.
3	numRssiBusy			W	Number of consecutive RSSI measurements above the threshold needed before the channel is declared BUSY.
4–5	corrPeriod			W	Number of RAT ticks for a correlation observation period
6	corrConfig	0–3	numCorrInv	W	Number of subsequent correlation tops with maximum corrPeriod RAT ticks between them needed to go from IDLE to INVALID.
		4–7	numCorrBusy	W	Number of subsequent correlation tops with maximum corrPeriod RAT ticks between them needed to go from INVALID to BUSY.
7	csEndTrigger			W	Trigger classifier for ending the carrier sense
8–11	csEndTime			W	Time to end carrier sense

Table 26-177. Receive Queue Entry Configuration Bit Field(1)

Bits	Bit Field Name	Description
0	bAutoFlushIgnored	If 1, automatically discard ignored packets from RX queue.
1	bAutoFlushCrcErr	If 1, automatically discard packets with CRC error from RX queue.
2		Reserved
3	bIncludeHdr	If 1, include the received header or length byte in the stored packet; otherwise discard it.
4	bIncludeCrc	If 1, include the received CRC field in the stored packet; otherwise discard it. This requires pktConf.bUseCrc to be 1.
5	bAppendRssi	If 1, append an RSSI byte to the packet in the RX queue.
6	bAppendTimestamp	If 1, append a timestamp to the packet in the RX queue.
7	bAppendStatus	If 1, append a status byte to the packet in the RX queue.

(1) This bit field is used for the rxConf byte of the parameter structures.

Table 26-178. Receive Status Byte Bit Field (1)

Bits	Bit Field Name	Description
0–4	addressInd	Index of address found (0 if not applicable)
5	syncWordId	0 for primary sync word, 1 for alternate sync word.
6–7	result	00: Packet received correctly, not ignored. 01: Packet received with CRC error. 10: Packet received correctly, but can be ignored. 11: Packet reception was aborted.

(1) A byte of this bit field is appended to the received entries if configured.

26.10.3 Interrupts

The radio CPU signals events back to the system CPU using firmware defined interrupts. Table 26-179 lists the interrupts to be used by the proprietary commands. Each interrupt may be enabled individually in the system CPU. Section 26.10.4 and Section 26.10.5 give details about when the interrupts are generated.

Table 26-179. Interrupt Definitions

Interrupt Number	Interrupt Name	Description
0	COMMAND_DONE	A radio operation command has finished
1	LAST_COMMAND_DONE	The last radio operation command in a chain of commands has finished
10	TX_ENTRY_DONE	For transmission of packets with unlimited length: Reading from a TX entry is finished
16	RX_OK	Packet received with CRC OK, payload, and not to be ignored
17	RX_NOK	Packet received with CRC error
18	RX_IGNORED	Packet received with CRC OK, but to be ignored
22	RX_BUF_FULL	Packet received that did not fit in RX buffer
23	RX_ENTRY_DONE	RX queue data entry changing state to Finished
24	RX_DATA_WRITTEN	Data written to partial read RX buffer
25	RX_N_DATA_WRITTEN	Specified number of bytes written to partial read RX buffer
26	RX_ABORTED	Packet reception stopped before packet was done
28	SYNTH_NO_LOCK	The synthesizer has reported loss of lock
29	MODULES_UNLOCKED	As part of the boot process, the Arm Cortex-M0 processor has opened access to RF core modules and memories
30	BOOT_DONE	The RF core CPU boot is finished
31	INTERNAL_ERROR	The radio CPU has observed an unexpected error

26.10.4 Data Handling

For the proprietary mode TX commands, data received over the air is stored in a receive queue. Partial-read RX buffers are supported, and mandatory for unlimited length. Data transmitted is fetched from a specific buffer.

26.10.4.1 Receive Buffers

A packet being received is stored in a receive buffer. First, a length byte or word is stored if configured in the RX entry by config.lenSz, and calculated from the length received over the air and the configuration of appended information, or for a partial-read RX buffer initialized to maximal possible size of that segment, and set to the length of the segment in one buffer when finished.

Following the optional length field, the received header is stored as received over the air if rxConf.bIncludeHdr is 1. This header is the length byte for CMD_PROP_RX and a field with up to 32 bits for CMD_PROP_RX_ADV. In the latter case, the last byte of the header is padded with zeros in the MSBs if the number of bits does not divide by 8, and followed by the received address (if configured) and payload.

If rxConf.bIncludeCrc is 1, the received CRC value is stored in the RX buffer; otherwise, it is not stored, but only used to check the CRC result. If rxConf.bAppendRssi is 1, a byte indicating the received RSSI value is appended. If rxConf.bAppendStatus is 1, a status byte of the type defined in Table 26-178 is appended. If rxConf.bAppendTimeStamp is 1, a timestamp indicating the start of the packet is appended. This timestamp corresponds to the ratmr_t data type. Though the timestamp is multibyte, no word-address alignment is made, so the timestamp must be written and read bytewise.

If the reception of a packet is aborted, the packet is immediately removed from the receive queue, except if a partial-read RX entry is used. In that case, the RSSI, Timestamp, and Status fields are appended if configured (except if no more buffer space is available), and the Status byte indicates that the reception was aborted.

Figure 26-11 shows the format of an entry element in the RX queue.

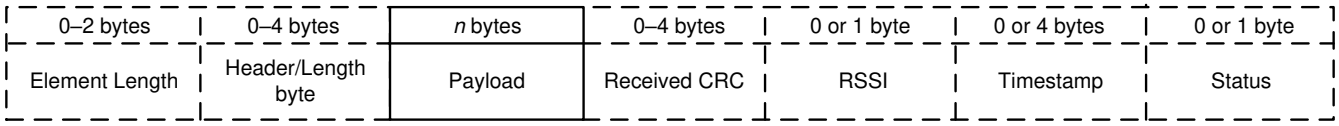


Figure 26-11. Receive Buffer Entry Element

An RX_ENTRY_DONE interrupt is raised when an RX entry changes its state to Finished. Depending on the type of RX entry used, this means:

- For a general or pointer entry, an RX_ENTRY_DONE interrupt is raised after a packet is fully received, unless the packet is automatically flushed.
- For a partial-read entry, an RX_ENTRY_DONE interrupt is raised when an RX entry is full, so writing must continue in the next entry.

For partial-read entries, an RX_Data_Written interrupt is raised whenever data is written to the receive buffer. An RX_N_Data_Written is raised whenever a multiple of config.irqIntv (as given in the data entry) bytes have been written since the start of the packet.

26.10.4.2 Transmit Buffers

The transmit operations contain a buffer with the data to be transmitted. The number of bytes in this buffer is given by pktLen. For the CMD_PROP_TX command, the length given in pktLen is transmitted as the first byte if pktConf.bVarLen is 1, and then followed by the contents of the transmit buffer.

For CMD_PROP_TX_ADV, the first bytes of the buffer contain the header if the header length is greater than 0. The number of bytes is the number of bits in the header divided by 8, rounded up. The MSBs of the last header byte are not sent if the number of bits does not divide by 8. If a length field is to be transmitted using CMD_PROP_TX_ADV, it must be given explicitly from the system side as part of the header.

If unlimited length is configured, a TX queue is used instead of one buffer. In this case, transmission of payload continues until the queue is emptied. Every time transmission from one entry is finished, meaning reading continues from the next entry or the entire payload is entered into the modem, a TX_ENTRY_DONE interrupt is raised.

26.10.5 Radio Operation Command Descriptions

Before running any of the proprietary RX or TX radio operation commands, the radio must be set up in proprietary mode using the command CMD_PROP_RADIO_SETUP or CMD_PROP_RADIO_DIV_SETUP, or in another compatible mode with CMD_RADIO_SETUP. Otherwise, the operation ends with an error. The RX and TX commands also need the synthesizer to be programmed using the CMD_FS command, which can typically be done by a command chain where an RX or TX command follows immediately after the CMD_FS.

26.10.5.1 End of Operation

The status field of the command issued is updated during the operation. When submitting the command, the system CPU must write this field with a state of IDLE. During the operation, the radio CPU updates the field to indicate the operation mode. When the operation is done, the radio CPU writes a status indicating that the operation is finished. The status codes used by a proprietary radio operation are listed in Table 26-180.

Table 26-180. Proprietary Radio Operation Status Codes

Number	Name	Description
Operation Not Finished		
0x0000	IDLE	Operation not started
0x0001	PENDING	Waiting for start trigger
0x0002	ACTIVE	Running operation
Operation Finished Normally		
0x3400	PROP_DONE_OK	Operation ended normally
0x3401	PROP_DONE_RXTIMEOUT	Operation stopped after end trigger while waiting for sync
0x3402	PROP_DONE_BREAK	RX stopped due to timeout in the middle of a packet
0x3403	PROP_DONE_ENDED	Operation stopped after end trigger during reception
0x3404	PROP_DONE_STOPPED	Operation stopped after stop command
0x3405	PROP_DONE_ABORT	Operation aborted by abort command
0x3406	PROP_DONE_RXERR	Operation ended after receiving packet with CRC error
0x3407	PROP_DONE_IDLE	Carrier sense operation ended because of idle channel
0x3408	PROP_DONE_BUSY	Carrier sense operation ended because of busy channel
0x3409	PROP_DONE_IDLETIMEOUT	Carrier sense operation ended because of time-out with csConf.timeoutRes = 1
0x340A	PROP_DONE_BUSYTIMEOUT	Carrier sense operation ended because of time-out with csConf.timeoutRes = 0
Operation Finished with Error		
0x3800	PROP_ERROR_PAR	Illegal parameter
0x3801	PROP_ERROR_RXBUF	No RX buffer large enough for the received data available at the start of a packet
0x3802	PROP_ERROR_RXFULL	Out of RX buffer during reception in a partial read buffer
0x3803	PROP_ERROR_NO_SETUP	Radio was not set up in proprietary mode
0x3804	PROP_ERROR_NO_FS	Synthesizer was not programmed when running RX or TX
0x3805	PROP_ERROR_RXOVF	TX overflow observed during operation
0x3806	PROP_ERROR_TXUNF	TX underflow observed during operation

The conditions for giving each status are listed for each operation. Some of the error causes listed in Table 26-180 are not repeated in these lists. If CMD_STOP or CMD_ABORT is received while waiting for the start trigger, the end cause is DONE_STOPPED or DONE_ABORT, with an end result of FALSE and ABORT, respectively. In some cases, general error causes may occur. For all these error cases, the result of the operation is ABORT.

26.10.5.2 Proprietary Mode Setup Command

The CMD_PROP_RADIO_SETUP and CMD_PROP_RADIO_DIV_SETUP commands are used instead of CMD_RADIO_SETUP for proprietary mode radio. When CMD_PROP_RADIO_SETUP or CMD_PROP_RADIO_DIV_SETUP is executing, trim values are read from FCFG1 unless they have been provided elsewhere (see Section 26.3.3.1.2 for more details).

On start, the radio CPU sets up parameters for the proprietary mode with parameters given in Table 26-173. The modulation.modType parameter selects between O-QPSK, GFSK and unshaped FSK. For FSK and GFSK, modulation.deviation gives the deviation in a step size that is programmable in modulation.stepSize. The radio CPU uses this parameter to calculate a proper shape for use in TX.

Note

The accuracy of the deviation may be less than the programmed step size due to hardware limitations.

If selecting O-QPSK as modulation format, the packets over the air will be formatted as per IEEE 802.15.4g MR-O-QPSK PHY specification. Only half sine baseband symbol shaping is supported (no RRC filtering).

The symbol rate is programmed with symbolRate. The parameters are passed directly to the modem and may be calculated using an external tool. The symbol rate is given by Equation 15.

$$f_{baud} = (R \times f_{clk}) / (p \times 2^{21}) \quad (15)$$

where

- f_{baud} is the obtained baud rate
- f_{clk} is the system clock frequency of 48 MHz
- R is the rate word given by symbolRate.rateWord
- p is the prescaler value, given by symbolRate.preScale, which can be from 4 to 255

The rxBw parameter gives the receiver bandwidth. Values 64–108 give the supported bandwidths with the recommended settings. Values 32–52 give a subset of this and is supported for backwards compatibility with CC13x0. Values 1–18 give the same bandwidths as settings 35–52, for compatibility with the CC13x4x10 and CC26x4x10 device platform. The values supported and corresponding settings are summarized in Table 26-181. The receiver bandwidths are stated for an RF frequency of 868 MHz (LO divider 5), 915 MHz (LO divider 5), and 2432 MHz (LO divider 2). The bandwidth is proportional to the RF frequency multiplied by the LO divider.

Table 26-181. Receiver Bandwidth Settings

Setting (Legacy)	Setting (Normal)	Setting (Enhanced)	Receiver Bandwidth (868 MHz)	Receiver Bandwidth (915 MHz)	Receiver Bandwidth (2432 MHz)	Default Intermediate Frequency (kHz)
—	—	64	4.3	4.5	4.8	62.5
—	—	65	4.9	5.1	5.4	62.5
—	—	66	6.1	6.5	6.9	62.5
—	—	67	7.4	7.8	8.2	62.5
—	—	68	8.5	9.0	9.6	125
—	—	69	9.7	10.2	10.9	125
—	—	70	12.2	12.9	13.7	125
—	—	71	14.7	15.5	16.5	125
—	—	72	17.1	18.0	19.1	125
—	—	73	19.4	20.5	21.8	125
—	—	74	24.5	25.8	27.4	125
—	—	75	29.4	31.0	33.0	125
—	—	76	34.1	36.0	38.3	250
—	32	77	38.9	41.0	43.5	250
—	33	78	49.0	51.6	54.9	250
—	34	79	58.9	62.1	66.0	250
—	—	80	68.3	72.0	76.5	250
1	35	81	77.7	81.9	87.1	250
2	36	82	98.0	103.3	109.8	250
3	37	83	117.7	124.1	131.9	250
—	—	84	136.6	144.0	153.1	500
4	38	85	155.4	163.8	174.2	500
5	39	86	195.9	206.5	219.6	500
6	40	87	235.5	248.2	263.9	500
—	—	88	273.1	287.9	306.1	1000
7	41	89	310.8	327.6	348.3	1000
8	42	90	391.8	413.0	439.1	1000
9	43	91	470.9	496.4	527.8	1000

Table 26-181. Receiver Bandwidth Settings (continued)

Setting (Legacy)	Setting (Normal)	Setting (Enhanced)	Receiver Bandwidth (868 MHz)	Receiver Bandwidth (915 MHz)	Receiver Bandwidth (2432 MHz)	Default Intermediate Frequency (kHz)
—	-	92	546.3	575.8	612.2	1000
10	44	93	621.6	655.3	696.7	1000
11	45	94	783.6	826.0	878.2	1000
12	46	95	941.8	992.8	1055.6	1000
—	—	96	1092.5	1151.7	1224.4	1000
13	47	97	1243.2	1310.5	1393.3	1000
14	48	98	1567.2	1652.1	1756.4	1000
15	49	99	1883.7	1985.7	2111.1	1000
—	—	100	2185.1	2303.4	2448.9	1000
16	50	101	2486.5	2621.1	2786.7	1000
17	51	102	3134.4	3304.2	3512.9	1000
18	52	103	3767.4	3971.4	4222.2	1000
Others			Reserved			

The CMD_PROP_RADIO_DIV_SETUP command contains settings for frequency band and intermediate frequency. The center frequency of the band to use is given by centerFreq, and used for calculating the transmitter shaping filter and the TX IF. The divider to use in the synthesizer is given by loDivider. The user must ensure that the setting is compatible with the given frequency. A value of 0 or 2 is only allowed for devices supporting operation in the 2.4 GHz band, and a value greater than 2 is only allowed for devices supporting operation in the Sub-1 GHz band. In the CMD_PROP_RADIO_SETUP command, centerFreq defaults to 2432 MHz and loDivider defaults to 0.

For CMD_PROP_RADIO_DIV_SETUP, the intermediate frequency can be specified through the intFreq parameter, which calculates the setting in the modem for RX and is written to the configuration parameter area. If this parameter is 0x8000 and for CMD_PROP_RADIO_SETUP, a default intermediate frequency as given in Table 26-181 is used. It is checked whether the configured intermediate frequency is supported for TX. If not, the TX intermediate frequency is set to 0. This happens if the intermediate frequency is greater than the RF center frequency divided by 15500. This causes the synthesizer to be reprogrammed without recalibration between RX and TX. This may increase the necessary turnaround time or in some cases cause the frequency synthesizer to get out of lock, meaning that a recalibration is necessary when switching between RX and TX.

The preamConf setting gives the preamble. The preamble is a sequence of 1010... or 0101..., where preamConf.preamMode gives the first transmitted bit. For more than 16 bytes, only an even number of bytes is supported. Setting preamConf.nPreamBytes = 31 gives a 4-bit preamble, and setting preamConf.nPremBytes = 0 gives a 1-bit preamble.

The formatConf setting is used for various setup of the packet format. The sync word length is given by nSwBits, which can be up to 32 bits. The bit polarity for FSK type modulation is given by bBitReversal, which must be 1 for compatibility with CC1101. The bit ordering is given by bMsbFirst, where 1 gives compatibility with the CC1101 device, and so forth. The whitenMode setting can select a whitener scheme. Other whiteners are obtained using override settings. Details of the IEEE 802.15.4g settings are given in Section 26.10.5.2.1. The fecMode setting can be used to change the encoding of the transmitted or received signal. For long-range mode (fecMode = 8), the nSwBits setting and the sync word programmed in the RX and TX commands are ignored, and a hard-coded 64-bit sync word with good performance is used. Setting fecMode to 10 enables Manchester coding. Only encoding/decoding of the payload and CRC is supported. A 0 will be encoded as 01b and a 1 as 10b.

The command sets up a 16-bit CRC with the polynomial $x^{16} + x^{15} + x^2 + 1$ and initialization of all 1s. This is compatible with the CC1101 device. Other polynomials, lengths, and initializations can be obtained by parameter overrides.

The pRegOverride parameter gives a pointer to an override structure, just as the one given for CMD_RADIO_SETUP. This parameter can be used for overriding parameters calculated from the other settings in the commands, as well as other parameters. If the value is NULL, no overrides are used.

26.10.5.2.1 IEEE 802.15.4g Packet Format

IEEE 802.15.4g PHY, including header, is supported by using the CMD_PROP_RX_ADV and CMD_PROP_TX_ADV commands.

The radio is configured to IEEE 802.15.4g mode by setting the formatConf.whitenMode field to the values 4, 5, 6, or 7, and formatConf.bMsbFirst must be set to 1 using the CMD_PROP_RADIO_DIV_SETUP command. For the CMD_PROP_TX_ADV and CMD_PROP_RX_ADV commands, pktConf.bCrcIncSw and pktConf.bCrcIncHdr must both be set to 0. For CMD_PROP_RX_ADV, hdrConf.numHdrBits must be set to 16, hdrConf.lenPos must be set to 0, hdrConf.numLenBits must be set to 11, and lenOffset must be −4.

When formatConf.whitenMode is 5 or 7, the radio is configured to produce the 32-bit CRC and whitening defined in IEEE 802.15.4g. When formatConf.whitenMode is 6 or 7, the radio also processes the headers in both receive and transmit as follows:

- If bit 15 of the header (counted from the LSB) is 1, the frame is assumed to consist of only a header, with no payload or CRC.
- If bit 12 of the header (counted from the LSB) is 1, the 16-bit CRC defined in IEEE 802.15.4g is assumed instead of the 32-bit CRC. For TX, 2 is added to the length offset to account for this, assuming the CRC is included in the received frame length.
- For mode 7: If bit 11 of the header (counted from the LSB) is 1, whitening is enabled; otherwise it is disabled.

Note

The IEEE 802.15.4g PHY header must be presented MSB first to the RF Core. In IEEE 802.15.4g specification, the payload part is LSb first, however the payload length info in physical layer header (PHR) is MSb first. This means that the payload needs to be flipped in the System CPU. This can be achieved with the System CPU assembly instruction RBIT.

The following example shows how to send a CRC-32 IEEE 802.15.4g frame with whitening enabled using the automatic headers processing feature (formatConf.whitenMode = 7).

```
/*
 * Prepare the .15.4g PHY header
 * MS=0, Length MSBits=0, DW and CRC settings read from 15.4g header (PHDR) by RF core.
 * Total length = transmit_len (payload) + CRC length
 *
 * The Radio will flip the bits around, so tx_buf[0] must have the
 * length LSBs (PHR[15:8] and tx_buf[1] will have PHR[7:0]
 */
/* Length in .15.4g PHY HDR includes the CRC but not the HDR itself */
```

```
uint16_t total_length;
```

```
total_length = transmit_len + CRC_LEN; /* CRC_LEN is 2 for CRC-16 and 4 for CRC-32 */
```

```
tx_buf[0] = total_length & 0xFF;
tx_buf[1] = (total_length >> 8) + 0x08 + 0x0; /* Whitening and CRC-32 bits */
tx_buf[2] = data;
```

An MCE patch is necessary to support FEC, Mode Switch, or other advanced features of IEEE 802.15.4g PHY.

26.10.5.3 Transmitter Commands

There are two commands for sending packets, CMD_PROP_TX and CMD_PROP_TX_ADV. The latter gives more flexibility in how the packet can be formed. Details of this are described in Section 26.10.5.3.1 and Section 26.10.5.3.2, respectively.

Both commands require the radio is set up in a compatible mode (such as proprietary mode), and that the synthesizer is programmed using CMD_FS.

For both commands, after the packet is transmitted, the frequency synthesizer is turned off when the command ends if pktConf.bFsOff is 1. If pktConf.bFsOff is 0, the synthesizer keeps running, so that the command must either be followed by an RX or TX command (which operate on the same frequency) or a CMD_FS_OFF command to turn off the synthesizer.

The end statuses for use with CMD_PROP_TX and CMD_PROP_TX_ADV are listed in Table 26-182. This status decides the next operation, see Section 26.10.5.1.

Table 26-182. End of Radio CMD_PROP_TX and CMD_PROP_TX_ADV Commands

Condition	Status Code	Result
Transmitted packet	PROP_DONE_OK	TRUE
Received CMD_STOP while transmitting packet and finished transmitting packet	PROP_DONE_STOPPED	FALSE
Received CMD_ABORT while transmitting packet	PROP_DONE_ABORT	ABORT
Observed illegal parameter	PROP_ERROR_PAR	ABORT
Command sent without setting up the radio in a supported mode using CMD_PROP_RADIO_SETUP or CMD_RADIO_SETUP	PROP_ERROR_NO_SETUP	ABORT
Command sent without the synthesizer being programmed	PROP_ERROR_NO_FS	ABORT
TX underflow observed during operation	PROP_ERROR_TXUNF	ABORT

26.10.5.3.1 Standard Transmit Command, CMD_PROP_TX

The CMD_PROP_TX command transmits a packet with the format from Table 26-170. The parameters are as given in Table 26-166.

The packet transmission starts at the given start trigger, with a fixed delay. The modem first transmits the preamble and sync word as configured. The sync word to transmit is given in the syncWord field, in the LSBs if less than 32 bits are used. The word is transmitted in the bit order programmed in the radio.

If pktConf.bVarLen is 1, a length byte equal to the value of pktLen is sent next. After this, the content of the buffer pointed to by pPkt is sent. This buffer consists of the number of bytes given in pktLen. If an address byte as shown in Figure 26-9 is needed, it must be sent as the first payload byte.

If pktConf.bUseCrc is 1, a CRC is calculated and transmitted at the end. The number of CRC bits, polynomial, and initialization are as configured in the radio. The CRC is calculated over the length byte (if present) and over the entire contents of the buffer pointed to by pPkt.

If whitening is enabled, the optional length byte, the entire contents of the buffer pointed to by pPkt, and the CRC are subject to whitening. The whitening is done after the data is used for CRC calculation.

26.10.5.3.2 Advanced Transmit Command, CMD_PROP_TX_ADV

The CMD_PROP_TX_ADV command transmits a packet with the format from Figure 26-10. As a special case, the user can set up packets as in Figure 26-9. The radio must be set up in a compatible mode (such as proprietary mode) and the synthesizer programmed using CMD_FS. The parameters are as given in Table 26-171.

The packet transmission starts at the given start trigger, with a fixed delay. Alternatively, if startConf.bExtTXTrig is 1, the packet transmission starts on an external trigger to the RF core. The trigger is identified as one of the inputs to the radio timer, and can be configured as rising edge, falling edge, or both edges through the startConf parameter. The system must ensure that this trigger comes after the start trigger, otherwise it is lost. The minimum delay after the start trigger is implementation-dependent and subject to characterization.

The modem first transmits the preamble and sync word as configured. If preTrigger is not TRIG_NOW, the configured preamble is repeated until that trigger (seen in combination with preTime) is observed. After the trigger is observed, the configured preamble under transmission finishes before the sync word transmission starts. If preTrigger is TRIG_NOW, the preamble is sent once, followed by the sync word. The sync word to

transmit is given in the syncWord field, in the LSBs if less than 32 bits are used, and is transmitted in the bit order programmed in the radio.

If numHdrBits is greater than 0, a header of numHdrBits is sent next. The header may contain a length field or an address. If so, these fields must be inserted correctly in the packet buffer. The header to be transmitted is the first bytes of the buffer pointed to by pPkt. If numHdrBits does not divide by 8, the MSBs of the last byte of the header are ignored.

The header is transmitted as one field in the bit ordering programmed in the radio. If the header has more than 8 bits, it is always read from the transmit buffer in little-endian byte order. If the radio is configured to transmit the MSB first, the last header byte from the TX buffer is transmitted first.

After the header, the remaining bytes in the buffer pointed to by pPkt are transmitted. The payload is transmitted byte by byte, so after the header, no swapping of bytes occurs regardless of bit ordering over the air. The total number of bytes (including the header) in this buffer is given by pktLen. If this length is too small to fit the header, the operation ends with PROP_ERROR_PAR as status. If an address field after the header as shown in Figure 26-10 is needed, it must be sent as the first payload byte.

If pktLen is 0, unlimited length is used. In this case, pPkt points to a transmit queue instead of a buffer, see Section 26.5.3.2.

If pktConf.bUseCrc is 1, a CRC is calculated and transmitted at the end. The number of CRC bits, polynomial, and initialization are as configured in the radio. If pktConf.bCrcIncSw is 1, the transmitted sync word is included in the data set over which the CRC is calculated. If pktConf.bCrcIncHdr is 1, the transmitted header is included in the data set over which the CRC is calculated. The payload is always used for calculating the CRC.

If whitening is enabled, the optional header is subject to whitening if pktConf.bCrcIncHdr is 1. The entire payload and the CRC are always subject to whitening when enabled. The whitening is done after the data is used for CRC calculation.

26.10.5.4 Receiver Commands

There are two commands for receiving packets, CMD_PROP_RX and CMD_PROP_RX_ADV. The latter gives more flexibility in how the packet can be formed. Details of this are described in Section 26.10.5.4.1 and Section 26.10.5.4.2, respectively.

For both commands, the radio must be set up in a compatible mode (such as proprietary mode), and the synthesizer must be programmed using CMD_FS before the command is sent to the radio core.

Both commands have an end trigger, given by endTrigger and endTime. If this trigger occurs while the receiver is searching for sync, the operation ends with the status PROP_DONE_RXTIMEOUT. If the trigger occurs while receiving a packet, the action depends on pktConf.endType. If pktConf.endType = 0, the packet is received to the end and the operation then ends with PROP_DONE_ENDED as the status. If pktConf.endType = 1, the packet reception is aborted and the operation ends with PROP_DONE_BREAK as the status. The radio receives packets according to the details given in Section 26.10.5.4.1 and Section 26.10.5.4.2, respectively. After receiving a packet, an interrupt is raised. If pOutput is not NULL, an output structure as given in Table 26-171, pointed to by pOutput, is updated as well. The interrupt to raise and field to update is given in Table 26-183. This table also gives the result to write in the status field of the receive buffer, if enabled. The condition for packets being ignored is described in Section 26.10.5.4.1 and Section 26.10.5.4.2, respectively.

Table 26-183. Interrupt, Counter, and Result Field for Received Packets (1)

Condition	Interrupt Raised	Counter Incremented	Result Field of Status Byte
Packet fully received with CRC OK and not to be ignored	RX_OK	nRxOk	0
Packet fully received with CRC error	RX_NOK	nRxNok	1
Packet fully received with CRC OK and address mismatch (pktConf.filterOp = 1)	RX_IGNORED	nRxIgnored	2
Packet reception aborted due to timeout (pktConf.endType = 1), CMD_ABORT, too short length in CMD_PROP_SET_LEN, or CMD_PROP_RESTART_RX	RX_ABORTED	nRxStopped	3 (1)

Table 26-183. Interrupt, Counter, and Result Field for Received Packets (1) (continued)

Condition	Interrupt Raised	Counter Incremented	Result Field of Status Byte
Packet reception aborted due to illegal length or address mismatch (pktConf.filterOp = 0)	RX_ABORTED	nRxStopped	–
Packet could not be stored due to lack of buffer space	RX_BUF_FULL	nRxBufFull	3 (1)

(1) Provided partial read entry is used and data is written to the buffer.

For both types of commands, the packet length may be configured as unlimited or unknown at the start of packet reception, by setting maxPktLen to 0. This mode can only be used with partial-read RX buffers. If the length is later determined, it can be set by the immediate or direct command CMD_PROP_SET_LEN, where the number of bytes between the header (if any) and the CRC is given. In addition to setting the length this way, packet reception may be stopped in the following ways (CRC check is not performed in the following cases):

- If CMD_PROP_SET_LEN is called with a smaller number of bytes than already received
- If CMD_PROP_RESTART_RX is given
- If no more RX buffer is available
- If the end trigger occurs and pktConf.endType is 1
- If the command is aborted with CMD_ABORT

For ignored packets and packets with CRC error, automatic flush of the receive buffer can be configured. In this case, packets are removed from the receive buffer after they have been received, so the next packet overwrites it and the counters are not updated to reflect the packet received.

Note

Automatic flush is not supported for partial-read RX entries. Packets with CRC error (that is, for which the RX_NOK interrupt is raised) are automatically flushed if rxConf.bAutoFlushCrcErr is 1.

Ignored packets (that is, for which the RX_IGNORED interrupt is raised) are automatically flushed if rxConf.bAutoFlushIgnored is 1. After a packet is received, the next action depends on pktConf.bRepeat. If this is 0, the command ends. Otherwise, it goes back into RX, unless another criterion exists that leads to the command to end. When the command ends, the frequency synthesizer is turned off if pktConf.bFsOff is 1. If pktConf.bFsOff is 0, the synthesizer keeps running, so that the command must either be followed by an RX or TX command (which operate on the same frequency) or a CMD_FS_OFF command to turn off the synthesizer.

The end statuses for CMD_PROP_RX and CMD_PROP_RX_ADV are listed in Table 26-184. This status decides the next operation, see Section 26.10.5.1.

Table 26-184. End of Radio CMD_PROP_RX and CMD_PROP_RX_ADV Commands

Condition	Status Code	Result
Received packet with CRC OK and pktConf.bRepeatOk = 0	PROP_DONE_OK	TRUE
Received packet with CRC error and pktConf.bRepeatNok = 0	PROP_DONE_RXERR	FALSE
Observed end trigger while in sync search	PROP_DONE_RXTIMEOUT	FALSE
Observed end trigger while receiving packet with pktConf.endType = 1	PROP_DONE_BREAK	FALSE
Received packet after observing an end trigger while receiving packet with pktConf.endType = 0	PROP_DONE_ENDED	FALSE
Received CMD_STOP after command started and, if sync found, packet is received	PROP_DONE_STOPPED	FALSE
Received CMD_ABORT after command started	PROP_DONE_ABORT	ABORT
No available RX buffer at the start of a packet	PROP_ERROR_RXBUF	FALSE
Out of RX buffer during reception in a partial read buffer	PROP_ERROR_RXFULL	FALSE
Observed illegal parameter	PROP_ERROR_PAR	ABORT
Command sent without setting up the radio in a supported mode using CMD_PROP_RADIO_SETUP or CMD_RADIO_SETUP	PROP_ERROR_NO_SETUP	ABORT
Command sent without the synthesizer being programmed	PROP_ERROR_NO_FS	ABORT

Table 26-184. End of Radio CMD_PROP_RX and CMD_PROP_RX_ADV Commands (continued)

Condition	Status Code	Result
TX overflow observed during operation	PROP_ERROR_RXOVF	ABORT

26.10.5.4.1 Standard Receive Command, CMD_PROP_RX

The CMD_PROP_RX receives packets with the format from Figure 26-9. The parameters are as given in Table 26-172.

The modem configures the receiver and starts listening for sync. The sync word to listen for is given in the syncWord field, in the LSBs if less than 32 bits are used. The word is in the bit order programmed in the radio.

If sync is found, the radio CPU starts receiving data. If pktConf.bVarLen is 1 and maxPktLen is nonzero, a length byte is assumed as the next byte. This length byte is compared to maxPktLen, and if it is greater, reception is stopped and synch search is restarted. Otherwise, this indicates the number of bytes after the length byte and before the CRC. If pktConf.bVarLen is 0, the length is fixed, and the receiver assumes maxPktLen bytes after the sync word and before the CRC. If maxPktLen is zero, the length is unlimited as described in the beginning of Section 26.10.5.4.

If pktConf.bChkAddress is 1, an address byte is checked next. The address byte is checked against the values of address0 and address1. If only one address is needed, these two fields must be set to the same value. If address1 is 0xFF, it is also checked against the value 0x00. To check for 0xFF without checking for 0x00, address0 must be the one set to 0xFF. If the address byte does not match the configured addresses, the further treatment depends on pktConf.filterOp. If this is 0, reception is stopped and synch search is restarted. If it is 1, the packet is received as if the address had matched, but it is marked as ignored.

If the packet is being received, the data is placed in the receive buffer. This receive buffer is found from the receive queue pointed to by pQueue. If pQueue is NULL, the packet is never stored.

If pktConf.bUseCrc is 1, a CRC is received and checked at the end. The number of CRC bits, polynomial, and initialization are as configured in the radio. The CRC is calculated over the length byte (if present), the optional address, and the payload. If pktConf.bUseCrc is 0, the treatment is as for CRC OK.

If whitening is enabled, the optional length byte, the payload including the optional address, and the received CRC are subject to dewhitening. The dewhitening is done before the CRC is evaluated.

If a status byte is appended (rxConf.bAppendStatus is 1) to the packet, it is formatted as follows (see Table 26-178). If pktConf.addressMode is nonzero, the addressInd field is 0 if the address matched address0, 1 if it matched address1, 2 if it matched 0x00 and this address was enabled, and 3 if it matched 0xFF and this address was enabled. Otherwise, addressInd is 0. The syncWordId field is always 0 for CMD_PROP_RX. The result field is written according to Table 26-184.

26.10.5.4.2 Advanced Receive Command, CMD_PROP_RX_ADV

The CMD_PROP_RX_ADV is used for receiving packets with the format from Figure 26-10. As a special case, the user can set up packets as in Figure 26-9. The parameters are as given in Table 26-173.

The modem configures the receiver and starts listening for sync. The sync word to listen for is given in the syncWord0 field, in the LSBs if less than 32 bits are used. The word is in the bit order programmed in the radio. If syncWord1 is nonzero, the receiver also listens for the sync word given in the syncWord1 field (formatted in the same way) if supported in the MCE. It is not possible to use two sync words when using CMD_PROP_RX_ADV_SNIFF with csConf.bEnaCorr set to 1.

If sync is found, the radio CPU starts receiving data. The packet may contain a header, which can consist of any number of bits up to 32, given by hdrConf.numHdrBits. If the number of bits in the header does not divide by 8, it is considered to consist of a sufficient number of bytes to contain all the stored bits, as shown in Section 26.5.3.1. This header may contain a length field or an address.

The received packet may have fixed or variable length. If hdrConf.numLenBits is 0 and maxPktLen is nonzero, the packet has a fixed length, consisting of maxPktLen bytes after the header and before the CRC. If hdrConf.numLenBits is greater than 0, a field of hdrConf.numLenBits, read from bit number hdrConf.lenPos

from the LSB of the header, is taken as a length field. The signed number lenOffset is added to the received length to give the number of bytes after the header and before the CRC. If this number is less than or equal to maxPktLen, the packet is received. If maxPktLen is zero, the length is unlimited as described in the beginning of Section 26.10.5.4. The definition of packet length for CMD_PROP_RX_ADV differs from the one for CMD_PROP_TX_ADV, see Section 26.10.5.4.2 where the header is included in the packet length.

Two kinds of addresses are supported. With the first option, the address is part of the header. In this case, the address size can be from 1 to 31 bits. The other option is to have an address after the header. If so, this address consists of between 1 and 8 bytes. To use an address as part of the header, addrConf.addrType must be set to 1. The number of bits in the address is given by addrConf.addrSize. These bits are read from bit number addrConf.addrPos from the first bit of the header. To use an address after the header, addrConf.addrType must be set to 0. In this case, the number of bytes in the address is given by addrConf.addrSize.

The received address is compared to an address list pointed to by pAddr. The address to compare against this list is as received. In addition, one bit identifying the sync word is concatenated with the address as the MSBs, if one of the following conditions is fulfilled:

- syncWord1 ≠ 0 and addrConf.addrType = 1
- syncWord1 ≠ 0, addrConf.addrType = 0, and addrConf.addrPos ≠ 0

This extra bit is 0 if the received sync word was syncWord0, and 1 if the received sync word was syncWord1. The entries in the address list have a size of 8, 16, 32, or 64 bits; the smallest size that can fit the address size, including the sync word identification bit if applicable. The number of entries in the address list is given by addrConf.numAddr. The radio CPU scans through the addresses in the address list and compares it to the received address. If there is no match, the further treatment depends on pktConf.filterOp. If this is 0, reception is stopped and synch search is restarted. If it is 1, the packet is received as if the address had matched, but marked as ignored.

If addrConf.addrSize is zero, no address is used. In this case, pAddr is ignored and must be NULL.

If the packet is being received, the data is placed in the receive buffer, as in Section 26.5.3.1. This receive buffer is found from the receive queue pointed to by pQueue. If pQueue is NULL, the packet is never stored.

The header is received as one field in the bit ordering programmed in the radio. If the header has more than 8 bits and rxConf.bIncludeHdr is 1, the header is always written in little-endian byte order to the receive buffer. If the radio is configured to receive the MSB first, the last header byte stored in the receive buffer is received first. The payload is stored byte by byte, so after the header, no swapping of bytes occurs regardless of bit ordering over the air.

If pktConf.bUseCrc is 1, a CRC is received and checked at the end. The number of CRC bits, polynomial, and initialization are as configured in the radio. If pktConf.bCrcIncSw is 1, the received sync word (assuming it to be exactly equal to syncWord0 or syncWord1) is included in the data set over which the CRC is calculated. If pktConf.bCrcIncHdr is 1, the received header is included in the data set over which the CRC is calculated. The payload, including the optional address after the header, is always used for calculating the CRC. If pktConf.bUseCrc is 0, the treatment is as for CRC OK.

If whitening is enabled, the optional header is subject to dewhitening only if pktConf.bCrcIncHdr is 1. The payload including the optional address after the header, and the received CRC, are always subject to dewhitening when enabled. The dewhitening is done before the CRC is evaluated.

If a status byte is appended (rxConf.bAppendStatus is 1) to the packet, it is formatted as follows (see Receive Buffers

Table 26-178). If addrConf.addrSize is nonzero, the addressInd field is the first index into the address list that matched the received address if an address match existed. Otherwise, addressInd is 0. The syncWordId field is 0 if the received sync word was syncWord0, and 1 if syncWord1. The result field is written according to Table 26-183.

26.10.5.5 Carrier-Sense Operation

The carrier-sense operation detects if a signal is present, which has the following main purposes:

- Turns off the radio instead of receiving when no signal is present
- Turns the radio to transmit only if no signal is present

The carrier sense operation can be used with the command CMD_PROP_CS to chain with another operation (for example, a transmit operation), or with the commands CMD_PROP_RX_SNIFF or CMD_PROP_RX_ADV_SNIFF to combine with a normal receive operation in order to implement sniff mode. The details of these commands are described in Section 26.10.5.5.1, Section 26.10.5.5.2, and Section 26.10.5.5.3.

26.10.5.5.1 Common Carrier-Sense Description

Section 26.10.2.3 gives the parameters for the carrier-sense operation, which are common for all the commands. The offset from the first byte used for carrier-sense parameters are also given in Section 26.10.2.3.

The channel can be in one of three states: BUSY, IDLE, or INVALID. BUSY indicates a signal on the channel. IDLE indicates no signal is present on the channel. INVALID indicates that the state cannot be determined. There are two sources of channel information, RSSI and correlation, and a separate state is maintained for each source.

The operation starts when the radio is set up in receive mode. The RSSI or correlation is monitored, according to the enable bits csConf.bEnaRssi and csConf.bEnaCorr. If csConf.bEnaRssi is 1, the RSSI is monitored. If csConf.bEnaCorr is 1, the correlator is set up to correlate against the preamble. It is not possible to set both enable bits to 0.

If csConf.bEnaRssi is 1, the RSSI is monitored every time a new value is available from the radio. At each update, the RSSI is compared against the signed value rssiThr. If the RSSI is below rssiThr and numRssiIdle consecutive RSSI measurements below the threshold have been observed, the RSSI state is IDLE. If the RSSI is above rssiThr and numRssiBusy consecutive RSSI measurements above the threshold have been observed, the RSSI state is BUSY. Otherwise, the RSSI state is INVALID.

If csConf.bEnaCorr is 1, the radio CPU monitors correlation peaks from the modem. When the radio starts, the state is INVALID. If no correlation top is observed until corrPeriod RAT ticks after the carrier sense command was started, the state becomes IDLE. If the state is IDLE and at least corrConfig.numCorrInv correlation tops with at most corrPeriod RAT ticks between them are observed, the state becomes Invalid. If the state is INVALID and at least corrConfig.numCorrBusy correlation tops with at most corrPeriod RAT ticks between them are observed, the state becomes BUSY. If corrConfig.numCorrBusy is 0, the state goes directly to BUSY from IDLE. The value of corrConfig.numCorrIdle must be greater than 0. If the state is not Idle and corrTime RAT ticks pass after the last correlation top, the state becomes IDLE again.

If only one of the enable bits is 1, the channel state is equal to the state of the corresponding source. If both enable bits are 1, the channel state depends on the state of the two sources and the csConf.operation bit, as shown in Table 26-185.

Table 26-185. Channel State when both Sources are Enabled

csConf.operation = 0			
RSSI State	Correlation State		
	INVALID	IDLE	BUSY
INVALID	INVALID	INVALID	BUSY
IDLE	INVALID	IDLE	BUSY
BUSY	BUSY	BUSY	BUSY
csConf.operation = 1			
RSSI State	Correlation State		
	INVALID	IDLE	BUSY
INVALID	INVALID	IDLE	INVALID
IDLE	IDLE	IDLE	IDLE
BUSY	INVALID	IDLE	BUSY

If the state of the channel changes to BUSY, the action depends on csConf.busyOp and the command being run. If csConf.busyOp is 0, the operation continues. If csConf.busyOp is 1 and the command is CMD_PROP_CS, the operation ends with PROP_DONE_BUSY as status. If csConf.busyOp is 1 and the command is CMD_PROP_RX_SNIFF or CMD_PROP_RX_ADV_SNIFF, the receive operation continues, but carrier sense is stopped, so the operation is not affected if the channel state later changes to IDLE.

If the state of the channel changes to IDLE, the action depends on csConf.idleOp. If the value of this field is 0, the receiver and carrier sense operation continues. If it is 1, the operation ends with PROP_DONE_IDLE as status.

If the trigger given by csEndTrigger and csEndTime is observed, the action depends on the command being run and the channel state at that time. The details are described in Section 26.10.5.5.2 and Section 26.10.5.5.3.

26.10.5.5.2 Carrier-Sense Command, CMD_PROP_CS

When the carrier-sense command starts, the radio is set up in receive mode, and the operations described in Section 26.10.5.5.1 are performed. The radio must be set up in a compatible mode (such as proprietary mode) and the synthesizer programmed using CMD_FS.

If the trigger given by csEndTrigger and csEndTime is observed, the operation ends, and the current channel state is checked. If the state is BUSY or IDLE, the status is PROP_DONE_BUSY or PROP_DONE_IDLE, respectively. If the state is INVALID, the status depends on csConf.timeoutRes. If 0, the status is PROP_DONE_BUSYTIMEOUT; if 1, PROP_DONE_IDLETIMEOUT.

When CMD_PROP_CS ends and the status is PROP_DONE_BUSY or PROP_DONE_BUSYTIMEOUT, the synthesizer is turned off if csFsConf.bFsOffBusy is 1. If the command ends and the status is PROP_DONE_IDLE or PROP_DONE_IDLETIMEOUT, the synthesizer is turned off if csFsConf.bFsOffIdle is 1. If it ends with another status, the synthesizer is turned off if either of these bits is 1.

The end statuses for use with CMD_PROP_CS are summarized in Table 26-186. This status decides the next operation, as shown in Section 26.10.5.1.

Table 26-186. End of CMD_PROP_CS Command

Condition	Status Code	Result
Observed channel state Busy with csConf.busyOp = 1	PROP_DONE_BUSY	TRUE
Observed channel state Idle with csConf.idleOp = 1	PROP_DONE_IDLE	FALSE
Timeout trigger observed with channel state Busy	PROP_DONE_BUSY	TRUE
Timeout trigger observed with channel state Idle	PROP_DONE_IDLE	FALSE
Timeout trigger observed with channel state Invalid and csConf.timeoutRes = 0	PROP_DONE_BUSYTIMEOUT	TRUE
Timeout trigger observed with channel state Invalid and csConf.timeoutRes = 1	PROP_DONE_IDLETIMEOUT	FALSE
Received CMD_STOP after command started	PROP_DONE_STOPPED	FALSE

Table 26-186. End of CMD_PROP_CS Command (continued)

Condition	Status Code	Result
Received CMD_ABORT after command started	PROP_DONE_ABORT	ABORT
Observed illegal parameter	PROP_ERROR_PAR	ABORT
Command sent without setting up the radio in a supported mode using CMD_PROP_RADIO_SETUP or CMD_RADIO_SETUP	PROP_ERROR_NO_SETUP	ABORT
Command sent without the synthesizer being programmed	PROP_ERROR_NO_FS	ABORT

26.10.5.5.3 Sniff Mode Receiver Commands, CMD_PROP_RX_SNIFF and CMD_PROP_RX_ADV_SNIFF

The commands CMD_PROP_RX_SNIFF and CMD_PROP_RX_ADV_SNIFF behave like the commands CMD_PROP_RX and CMD_PROP_RX_ADV, respectively, but they perform carrier-sense operations during sync search.

When started, the commands perform the carrier-sense operations described in Section 26.10.5.5.1. As described, the operation may end if the channel state becomes IDLE.

If the trigger given by csEndTrigger and csEndTime is observed, the current channel state is checked. If BUSY, the receiver continues, but may end later if the channel state becomes IDLE and csConf.busyOp is 0. If the channel state is IDLE, the operation ends (even if csConf.idleOp is 0), and the status is PROP_DONE_IDLE. If the channel state is INVALID, the action depends on csConf.timeoutRes. If 0, the receive operation continues, and if csConf.busyOp is 1, carrier sense is no longer checked. If csConf.timeoutRes is 1, the operation ends and the status is PROP_DONE_IDLETIMEOUT.

If sync is found, the receiver operates as described in Section 26.10.5.4. If sync search is restarted after a packet is received or after reception is stopped due to an invalid length field or address mismatch, the carrier-sense operation is resumed if it was running when sync was found.

The end statuses for use with CMD_PROP_RX_SNIFF and CMD_PROP_RX_ADV_SNIFF are listed in Table 26-184 and Table 26-187. This status decides the next operation, as in Section 26.10.5.1.

Table 26-187. Additional End Statuses for CMD_PROP_RX_SNIFF and CMD_PROP_RX_ADV_SNIFF

Condition	Status Code	Result
Observed channel state IDLE with csConf.idleOp = 1	PROP_DONE_IDLE	FALSE
Timeout trigger observed with channel state IDLE	PROP_DONE_IDLE	FALSE
Timeout trigger observed with channel state INVALID and csConf.timeoutRes = 1	PROP_DONE_IDLETIMEOUT	FALSE

26.10.6 Immediate Commands

26.10.6.1 Set Packet Length Command, CMD_PROP_SET_LEN

The CMD_PROP_SET_LEN command takes a command structure as defined in Table 26-174.

CMD_PROP_SET_LEN must only be sent while a CMD_PROP_RX or CMD_PROP_RX_ADV command is running configured with unlimited packet length. On reception of the command, the radio CPU sets the number of bytes to receive between the header and the CRC to RXLen. If at least this number of bytes has already been received, reception is aborted, as in Section 26.10.5.4 and Section 26.10.5.4.2.

The command may be sent as a direct command if the payload length to set is 255 bytes or less. In this case, the RXLen parameter is written in bits 8–16 of CMDR, and the 8 MSBs of this parameter is 0.

If the command is issued without a CMD_PROP_RX or CMD_PROP_RX_ADV command running, or if such a command is not configured with unlimited length, the radio CPU returns the result ContextError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.10.6.2 Restart Packet RX Command, CMD_PROP_RESTART_RX

The CMD_PROP_RESTART_RX command is a direct command that takes no parameters.

CMD_PROP_RESTART_RX must only be sent while a CMD_PROP_RX or CMD_PROP_RX_ADV command is running. If a packet is being received, reception is aborted as described in Section 26.10.5.4 and the packet returns to sync search.

If the command is issued without an RX command running, the radio CPU returns the result ContextError in CMDSTA. Otherwise, the radio CPU returns DONE.

26.11 Radio Registers

26.11.1 RFC_RAT Registers

Table 26-188 lists the memory-mapped registers for the RFC_RAT registers. All register offset addresses not listed in Table 26-188 should be considered as reserved locations and the register contents should not be modified.

Table 26-188. RFC_RAT Registers

Offset	Acronym	Register Name	Section
4h	RATCNT	Radio Timer Counter Value	Section 26.11.1.1
80h	RATCH0VAL	Timer Channel 0 Capture/Compare Register	Section 26.11.1.2
84h	RATCH1VAL	Timer Channel 1 Capture/Compare Register	Section 26.11.1.3
88h	RATCH2VAL	Timer Channel 2 Capture/Compare Register	Section 26.11.1.4
8Ch	RATCH3VAL	Timer Channel 3 Capture/Compare Register	Section 26.11.1.5
90h	RATCH4VAL	Timer Channel 4 Capture/Compare Register	Section 26.11.1.6
94h	RATCH5VAL	Timer Channel 5 Capture/Compare Register	Section 26.11.1.7
98h	RATCH6VAL	Timer Channel 6 Capture/Compare Register	Section 26.11.1.8
9Ch	RATCH7VAL	Timer Channel 7 Capture/Compare Register	Section 26.11.1.9

Complex bit access types are encoded to fit into small table cells. Table 26-189 shows the codes that are used for access types in this section.

Table 26-189. RFC_RAT Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

26.11.1.1 RATCNT Register (Offset = 4h) [Reset = 00000000h]

RATCNT is shown in Table 26-190.

Return to the Summary Table.

Radio Timer Counter Value

Table 26-190. RATCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CNT	R/W	0h	Counter value. This is not writable while radio timer counter is enabled.

26.11.1.2 RATCH0VAL Register (Offset = 80h) [Reset = 00000000h]

RATCH0VAL is shown in Table 26-191.

Return to the Summary Table.

Timer Channel 0 Capture/Compare Register

Table 26-191. RATCH0VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.3 RATCH1VAL Register (Offset = 84h) [Reset = 00000000h]

RATCH1VAL is shown in Table 26-192.

Return to the Summary Table.

Timer Channel 1 Capture/Compare Register

Table 26-192. RATCH1VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.4 RATCH2VAL Register (Offset = 88h) [Reset = 00000000h]

RATCH2VAL is shown in Table 26-193.

Return to the Summary Table.

Timer Channel 2 Capture/Compare Register

Table 26-193. RATCH2VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.5 RATCH3VAL Register (Offset = 8Ch) [Reset = 00000000h]

RATCH3VAL is shown in Table 26-194.

Return to the Summary Table.

Timer Channel 3 Capture/Compare Register

Table 26-194. RATCH3VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.6 RATCH4VAL Register (Offset = 90h) [Reset = 00000000h]

RATCH4VAL is shown in Table 26-195.

Return to the Summary Table.

Timer Channel 4 Capture/Compare Register

Table 26-195. RATCH4VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.7 RATCH5VAL Register (Offset = 94h) [Reset = 00000000h]

RATCH5VAL is shown in Table 26-196.

Return to the Summary Table.

Timer Channel 5 Capture/Compare Register

Table 26-196. RATCH5VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.8 RATCH6VAL Register (Offset = 98h) [Reset = 00000000h]

RATCH6VAL is shown in Table 26-197.

Return to the Summary Table.

Timer Channel 6 Capture/Compare Register

Table 26-197. RATCH6VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.1.9 RATCH7VAL Register (Offset = 9Ch) [Reset = 00000000h]

RATCH7VAL is shown in Table 26-198.

Return to the Summary Table.

Timer Channel 7 Capture/Compare Register

Table 26-198. RATCH7VAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	VAL	R/W	0h	Capture/compare value. Only writable when the channel is configured for compare mode. In compare mode, a write to this register will auto-arm the channel.

26.11.2 RFC_DBELL Registers

Table 26-199 lists the memory-mapped registers for the RFC_DBELL registers. All register offset addresses not listed in Table 26-199 should be considered as reserved locations and the register contents should not be modified.

Table 26-199. RFC_DBELL Registers

Offset	Acronym	Register Name	Section
0h	CMDR	Doorbell Command Register	Section 26.11.2.1
4h	CMDSTA	Doorbell Command Status Register	Section 26.11.2.2
8h	RFHWIFG	Interrupt Flags From RF Hardware Modules	Section 26.11.2.3
Ch	RFHWIEN	Interrupt Enable For RF Hardware Modules	Section 26.11.2.4
10h	RFCPEIFG	Interrupt Flags For Command and Packet Engine Generated Interrupts	Section 26.11.2.5
14h	RFCPEIEN	Interrupt Enable For Command and Packet Engine Generated Interrupts	Section 26.11.2.6
18h	RFCPEISL	Interrupt Vector Selection For Command and Packet Engine Generated Interrupts	Section 26.11.2.7
1Ch	RFACKIFG	Doorbell Command Acknowledgement Interrupt Flag	Section 26.11.2.8
20h	SYSGPOCTL	RF Core General Purpose Output Control	Section 26.11.2.9

Complex bit access types are encoded to fit into small table cells. Table 26-200 shows the codes that are used for access types in this section.

Table 26-200. RFC_DBELL Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

26.11.2.1 CMDR Register (Offset = 0h) [Reset = 00000000h]

CMDR is shown in Table 26-201.

Return to the Summary Table.

Doorbell Command Register

Table 26-201. CMDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CMD	R/W	0h	Command register. Raises an interrupt to the Command and packet engine (CPE) upon write.

26.11.2.2 CMDSTA Register (Offset = 4h) [Reset = 00000000h]

CMDSTA is shown in Table 26-202.

Return to the Summary Table.

Doorbell Command Status Register

Table 26-202. CMDSTA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	STAT	R	0h	Status of the last command used

26.11.2.3 RFHWIFG Register (Offset = 8h) [Reset = 00000000h]

RFHWIFG is shown in Table 26-203.

Return to the Summary Table.

Interrupt Flags From RF Hardware Modules

Table 26-203. RFHWIFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19	RATCH7	R/W	0h	Radio timer channel 7 interrupt flag. Write zero to clear flag. Write to one has no effect.
18	RATCH6	R/W	0h	Radio timer channel 6 interrupt flag. Write zero to clear flag. Write to one has no effect.
17	RATCH5	R/W	0h	Radio timer channel 5 interrupt flag. Write zero to clear flag. Write to one has no effect.
16	RATCH4	R/W	0h	Radio timer channel 4 interrupt flag. Write zero to clear flag. Write to one has no effect.
15	RATCH3	R/W	0h	Radio timer channel 3 interrupt flag. Write zero to clear flag. Write to one has no effect.
14	RATCH2	R/W	0h	Radio timer channel 2 interrupt flag. Write zero to clear flag. Write to one has no effect.
13	RATCH1	R/W	0h	Radio timer channel 1 interrupt flag. Write zero to clear flag. Write to one has no effect.
12	RATCH0	R/W	0h	Radio timer channel 0 interrupt flag. Write zero to clear flag. Write to one has no effect.
11	RFESOFT2	R/W	0h	RF engine software defined interrupt 2 flag. Write zero to clear flag. Write to one has no effect.
10	RFESOFT1	R/W	0h	RF engine software defined interrupt 1 flag. Write zero to clear flag. Write to one has no effect.
9	RFESOFT0	R/W	0h	RF engine software defined interrupt 0 flag. Write zero to clear flag. Write to one has no effect.
8	RFEDONE	R/W	0h	RF engine command done interrupt flag. Write zero to clear flag. Write to one has no effect.
7	RESERVED	R	0h	Reserved
6	TRCTK	R/W	0h	Debug tracer system tick interrupt flag. Write zero to clear flag. Write to one has no effect.
5	MDMSOFT	R/W	0h	Modem software defined interrupt flag. Write zero to clear flag. Write to one has no effect.
4	MDMOUT	R/W	0h	Modem FIFO output interrupt flag. Write zero to clear flag. Write to one has no effect.
3	MDMIN	R/W	0h	Modem FIFO input interrupt flag. Write zero to clear flag. Write to one has no effect.
2	MDMDONE	R/W	0h	Modem command done interrupt flag. Write zero to clear flag. Write to one has no effect.
1	FSCA	R/W	0h	Frequency synthesizer calibration accelerator interrupt flag. Write zero to clear flag. Write to one has no effect.
0	RESERVED	R	0h	Reserved

26.11.2.4 RFHWIEN Register (Offset = Ch) [Reset = 00000000h]

RFHWIEN is shown in Table 26-204.

Return to the Summary Table.

Interrupt Enable For RF Hardware Modules

Table 26-204. RFHWIEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19	RATCH7	R/W	0h	Interrupt enable for RFHWIFG.RATCH7.
18	RATCH6	R/W	0h	Interrupt enable for RFHWIFG.RATCH6.
17	RATCH5	R/W	0h	Interrupt enable for RFHWIFG.RATCH5.
16	RATCH4	R/W	0h	Interrupt enable for RFHWIFG.RATCH4.
15	RATCH3	R/W	0h	Interrupt enable for RFHWIFG.RATCH3.
14	RATCH2	R/W	0h	Interrupt enable for RFHWIFG.RATCH2.
13	RATCH1	R/W	0h	Interrupt enable for RFHWIFG.RATCH1.
12	RATCH0	R/W	0h	Interrupt enable for RFHWIFG.RATCH0.
11	RFESOFT2	R/W	0h	Interrupt enable for RFHWIFG.RFESOFT2.
10	RFESOFT1	R/W	0h	Interrupt enable for RFHWIFG.RFESOFT1.
9	RFESOFT0	R/W	0h	Interrupt enable for RFHWIFG.RFESOFT0.
8	RFEDONE	R/W	0h	Interrupt enable for RFHWIFG.RFEDONE.
7	RESERVED	R	0h	Reserved
6	TRCTK	R/W	0h	Interrupt enable for RFHWIFG.TRCTK.
5	MDMSOFT	R/W	0h	Interrupt enable for RFHWIFG.MDMSOFT.
4	MDMOUT	R/W	0h	Interrupt enable for RFHWIFG.MDMOUT.
3	MDMIN	R/W	0h	Interrupt enable for RFHWIFG.MDMIN.
2	MDMDONE	R/W	0h	Interrupt enable for RFHWIFG.MDMDONE.
1	FSCA	R/W	0h	Interrupt enable for RFHWIFG.FSCA.
0	RESERVED	R	0h	Reserved

26.11.2.5 RFCPEIFG Register (Offset = 10h) [Reset = 00000000h]

RFCPEIFG is shown in Table 26-205.

Return to the Summary Table.

Interrupt Flags For Command and Packet Engine Generated Interrupts

Table 26-205. RFCPEIFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	INTERNAL_ERROR	R/W	0h	Interrupt flag 31. The command and packet engine (CPE) has observed an unexpected error. A reset of the CPE is needed. This can be done by switching the RF Core power domain off and on in PRCM:PDCTL1RFC. Write zero to clear flag. Write to one has no effect.
30	BOOT_DONE	R/W	0h	Interrupt flag 30. The command and packet engine (CPE) boot is finished. Write zero to clear flag. Write to one has no effect.
29	MODULES_UNLOCKED	R/W	0h	Interrupt flag 29. As part of command and packet engine (CPE) boot process, it has opened access to RF Core modules and memories. Write zero to clear flag. Write to one has no effect.
28	SYNTH_NO_LOCK	R/W	0h	Interrupt flag 28. The phase-locked loop in frequency synthesizer has reported loss of lock. Write zero to clear flag. Write to one has no effect.
27	IRQ27	R/W	0h	Interrupt flag 27. Write zero to clear flag. Write to one has no effect.
26	RX_ABORTED	R/W	0h	Interrupt flag 26. Packet reception stopped before packet was done. Write zero to clear flag. Write to one has no effect.
25	RX_N_DATA_WRITTEN	R/W	0h	Interrupt flag 25. Specified number of bytes written to partial read Rx buffer. Write zero to clear flag. Write to one has no effect.
24	RX_DATA_WRITTEN	R/W	0h	Interrupt flag 24. Data written to partial read Rx buffer. Write zero to clear flag. Write to one has no effect.
23	RX_ENTRY_DONE	R/W	0h	Interrupt flag 23. Rx queue data entry changing state to finished. Write zero to clear flag. Write to one has no effect.
22	RX_BUF_FULL	R/W	0h	Interrupt flag 22. Packet received that did not fit in Rx queue. BLE mode: Packet received that did not fit in the Rx queue. IEEE 802.15.4 mode: Frame received that did not fit in the Rx queue. Write zero to clear flag. Write to one has no effect.
21	RX_CTRL_ACK	R/W	0h	Interrupt flag 21. BLE mode only: LL control packet received with CRC OK, not to be ignored, then acknowledgement sent. Write zero to clear flag. Write to one has no effect.
20	RX_CTRL	R/W	0h	Interrupt flag 20. BLE mode only: LL control packet received with CRC OK, not to be ignored. Write zero to clear flag. Write to one has no effect.
19	RX_EMPTY	R/W	0h	Interrupt flag 19. BLE mode only: Packet received with CRC OK, not to be ignored, no payload. Write zero to clear flag. Write to one has no effect.
18	RX_IGNORED	R/W	0h	Interrupt flag 18. Packet received, but can be ignored. BLE mode: Packet received with CRC OK, but to be ignored. IEEE 802.15.4 mode: Frame received with ignore flag set. Write zero to clear flag. Write to one has no effect.

Table 26-205. RFCPEIFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	RX_NOK	R/W	0h	Interrupt flag 17. Packet received with CRC error. BLE mode: Packet received with CRC error. IEEE 802.15.4 mode: Frame received with CRC error. Write zero to clear flag. Write to one has no effect.
16	RX_OK	R/W	0h	Interrupt flag 16. Packet received correctly. BLE mode: Packet received with CRC OK, payload, and not to be ignored. IEEE 802.15.4 mode: Frame received with CRC OK. Write zero to clear flag. Write to one has no effect.
15	IRQ15	R/W	0h	Interrupt flag 15. Write zero to clear flag. Write to one has no effect.
14	IRQ14	R/W	0h	Interrupt flag 14. Write zero to clear flag. Write to one has no effect.
13	FG_COMMAND_STARTED	R/W	0h	Interrupt flag 13. IEEE 802.15.4 mode only: A foreground radio operation command has gone into active state.
12	COMMAND_STARTED	R/W	0h	Interrupt flag 12. A radio operation command has gone into active state.
11	TX_BUFFER_CHANGED	R/W	0h	Interrupt flag 11. BLE mode only: A buffer change is complete after CMD_BLE_ADV_PAYLOAD. Write zero to clear flag. Write to one has no effect.
10	TX_ENTRY_DONE	R/W	0h	Interrupt flag 10. Tx queue data entry state changed to finished. Write zero to clear flag. Write to one has no effect.
9	TX_RETRANS	R/W	0h	Interrupt flag 9. BLE mode only: Packet retransmitted. Write zero to clear flag. Write to one has no effect.
8	TX_CTRL_ACK_ACK	R/W	0h	Interrupt flag 8. BLE mode only: Acknowledgement received on a transmitted LL control packet, and acknowledgement transmitted for that packet. Write zero to clear flag. Write to one has no effect.
7	TX_CTRL_ACK	R/W	0h	Interrupt flag 7. BLE mode: Acknowledgement received on a transmitted LL control packet. Write zero to clear flag. Write to one has no effect.
6	TX_CTRL	R/W	0h	Interrupt flag 6. BLE mode: Transmitted LL control packet. Write zero to clear flag. Write to one has no effect.
5	TX_ACK	R/W	0h	Interrupt flag 5. BLE mode: Acknowledgement received on a transmitted packet. IEEE 802.15.4 mode: Transmitted automatic ACK frame. Write zero to clear flag. Write to one has no effect.
4	TX_DONE	R/W	0h	Interrupt flag 4. Packet transmitted. (BLE mode: A packet has been transmitted.) (IEEE 802.15.4 mode: A frame has been transmitted). Write zero to clear flag. Write to one has no effect.
3	LAST_FG_COMMAND_DONE	R/W	0h	Interrupt flag 3. IEEE 802.15.4 mode only: The last foreground radio operation command in a chain of commands has finished. Write zero to clear flag. Write to one has no effect.
2	FG_COMMAND_DONE	R/W	0h	Interrupt flag 2. IEEE 802.15.4 mode only: A foreground radio operation command has finished. Write zero to clear flag. Write to one has no effect.
1	LAST_COMMAND_DONE	R/W	0h	Interrupt flag 1. The last radio operation command in a chain of commands has finished. (IEEE 802.15.4 mode: The last background level radio operation command in a chain of commands has finished.) Write zero to clear flag. Write to one has no effect.

Table 26-205. RFCPEIFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	COMMAND_DONE	R/W	0h	Interrupt flag 0. A radio operation has finished. (IEEE 802.15.4 mode: A background level radio operation command has finished.) Write zero to clear flag. Write to one has no effect.

26.11.2.6 RFCPEIEN Register (Offset = 14h) [Reset = FFFFFFFFh]

RFCPEIEN is shown in Table 26-206.

Return to the Summary Table.

Interrupt Enable For Command and Packet Engine Generated Interrupts

Table 26-206. RFCPEIEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31	INTERNAL_ERROR	R/W	1h	Interrupt enable for RFCPEIFG.INTERNAL_ERROR.
30	BOOT_DONE	R/W	1h	Interrupt enable for RFCPEIFG.BOOT_DONE.
29	MODULES_UNLOCKED	R/W	1h	Interrupt enable for RFCPEIFG.MODULES_UNLOCKED.
28	SYNTH_NO_LOCK	R/W	1h	Interrupt enable for RFCPEIFG.SYNTH_NO_LOCK.
27	IRQ27	R/W	1h	Interrupt enable for RFCPEIFG.IRQ27.
26	RX_ABORTED	R/W	1h	Interrupt enable for RFCPEIFG.RX_ABORTED.
25	RX_N_DATA_WRITTEN	R/W	1h	Interrupt enable for RFCPEIFG.RX_N_DATA_WRITTEN.
24	RX_DATA_WRITTEN	R/W	1h	Interrupt enable for RFCPEIFG.RX_DATA_WRITTEN.
23	RX_ENTRY_DONE	R/W	1h	Interrupt enable for RFCPEIFG.RX_ENTRY_DONE.
22	RX_BUF_FULL	R/W	1h	Interrupt enable for RFCPEIFG.RX_BUF_FULL.
21	RX_CTRL_ACK	R/W	1h	Interrupt enable for RFCPEIFG.RX_CTRL_ACK.
20	RX_CTRL	R/W	1h	Interrupt enable for RFCPEIFG.RX_CTRL.
19	RX_EMPTY	R/W	1h	Interrupt enable for RFCPEIFG.RX_EMPTY.
18	RX_IGNORED	R/W	1h	Interrupt enable for RFCPEIFG.RX_IGNORED.
17	RX_NOK	R/W	1h	Interrupt enable for RFCPEIFG.RX_NOK.
16	RX_OK	R/W	1h	Interrupt enable for RFCPEIFG.RX_OK.
15	IRQ15	R/W	1h	Interrupt enable for RFCPEIFG.IRQ15.
14	IRQ14	R/W	1h	Interrupt enable for RFCPEIFG.IRQ14.
13	FG_COMMAND_STARTED	R/W	1h	Interrupt enable for RFCPEIFG.FG_COMMAND_STARTED.
12	COMMAND_STARTED	R/W	1h	Interrupt enable for RFCPEIFG.COMMAND_STARTED.
11	TX_BUFFER_CHANGED	R/W	1h	Interrupt enable for RFCPEIFG.TX_BUFFER_CHANGED.
10	TX_ENTRY_DONE	R/W	1h	Interrupt enable for RFCPEIFG.TX_ENTRY_DONE.
9	TX_RETRANS	R/W	1h	Interrupt enable for RFCPEIFG.TX_RETRANS.
8	TX_CTRL_ACK_ACK	R/W	1h	Interrupt enable for RFCPEIFG.TX_CTRL_ACK_ACK.
7	TX_CTRL_ACK	R/W	1h	Interrupt enable for RFCPEIFG.TX_CTRL_ACK.
6	TX_CTRL	R/W	1h	Interrupt enable for RFCPEIFG.TX_CTRL.
5	TX_ACK	R/W	1h	Interrupt enable for RFCPEIFG.TX_ACK.
4	TX_DONE	R/W	1h	Interrupt enable for RFCPEIFG.TX_DONE.
3	LAST_FG_COMMAND_DONE	R/W	1h	Interrupt enable for RFCPEIFG.LAST_FG_COMMAND_DONE.
2	FG_COMMAND_DONE	R/W	1h	Interrupt enable for RFCPEIFG.FG_COMMAND_DONE.
1	LAST_COMMAND_DONE	R/W	1h	Interrupt enable for RFCPEIFG.LAST_COMMAND_DONE.
0	COMMAND_DONE	R/W	1h	Interrupt enable for RFCPEIFG.COMMAND_DONE.

26.11.2.7 RFCPEISL Register (Offset = 18h) [Reset = FFFF0000h]

RFCPEISL is shown in Table 26-207.

Return to the Summary Table.

Interrupt Vector Selection For Command and Packet Engine Generated Interrupts

Table 26-207. RFCPEISL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	INTERNAL_ERROR	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.INTERNAL_ERROR interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
30	BOOT_DONE	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.BOOT_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
29	MODULES_UNLOCKED	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.MODULES_UNLOCKED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
28	SYNTH_NO_LOCK	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.SYNTH_NO_LOCK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
27	IRQ27	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.IRQ27 interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
26	RX_ABORTED	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_ABORTED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
25	RX_N_DATA_WRITTEN	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_N_DATA_WRITTEN interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
24	RX_DATA_WRITTEN	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_DATA_WRITTEN interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector

Table 26-207. RFCPEISL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
23	RX_ENTRY_DONE	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_ENTRY_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
22	RX_BUF_FULL	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_BUF_FULL interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
21	RX_CTRL_ACK	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_CTRL_ACK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
20	RX_CTRL	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_CTRL interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
19	RX_EMPTY	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_EMPTY interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
18	RX_IGNORED	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_IGNORED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
17	RX_NOK	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_NOK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
16	RX_OK	R/W	1h	Select which CPU interrupt vector the RFCPEIFG.RX_OK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
15	IRQ15	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.IRQ15 interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
14	IRQ14	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.IRQ14 interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector

Table 26-207. RFCPEISL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13	FG_COMMAND_STARTED	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.FG_COMMAND_STARTED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
12	COMMAND_STARTED	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.COMMAND_STARTED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
11	TX_BUFFER_CHANGED	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_BUFFER_CHANGED interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
10	TX_ENTRY_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_ENTRY_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
9	TX_RETRANS	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_RETRANS interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
8	TX_CTRL_ACK_ACK	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_CTRL_ACK_ACK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
7	TX_CTRL_ACK	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_CTRL_ACK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
6	TX_CTRL	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_CTRL interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
5	TX_ACK	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_ACK interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector

Table 26-207. RFCPEISL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	TX_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.TX_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
3	LAST_FG_COMMAND_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.LAST_FG_COMMAND_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
2	FG_COMMAND_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.FG_COMMAND_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
1	LAST_COMMAND_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.LAST_COMMAND_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector
0	COMMAND_DONE	R/W	0h	Select which CPU interrupt vector the RFCPEIFG.COMMAND_DONE interrupt should use. 0h = Associate this interrupt line with INT_RF_CPE0 interrupt vector 1h = Associate this interrupt line with INT_RF_CPE1 interrupt vector

26.11.2.8 RFACKIFG Register (Offset = 1Ch) [Reset = 00000000h]

RFACKIFG is shown in Table 26-208.

Return to the Summary Table.

Doorbell Command Acknowledgement Interrupt Flag

Table 26-208. RFACKIFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ACKFLAG	R/W	0h	Interrupt flag for Command ACK

26.11.2.9 SYSGPOCTL Register (Offset = 20h) [Reset = 00000000h]

SYSGPOCTL is shown in Table 26-209.

Return to the Summary Table.

RF Core General Purpose Output Control

Table 26-209. SYSGPOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	GPOCTL3	R/W	0h	RF Core GPO control bit 3. Selects which signal to output on the RF Core GPO line 3. 0h = CPE GPO line 0 1h = CPE GPO line 1 2h = CPE GPO line 2 3h = CPE GPO line 3 4h = MCE GPO line 0 5h = MCE GPO line 1 6h = MCE GPO line 2 7h = MCE GPO line 3 8h = RFE GPO line 0 9h = RFE GPO line 1 Ah = RFE GPO line 2 Bh = RFE GPO line 3 Ch = RAT GPO line 0 Dh = RAT GPO line 1 Eh = RAT GPO line 2 Fh = RAT GPO line 3
11-8	GPOCTL2	R/W	0h	RF Core GPO control bit 2. Selects which signal to output on the RF Core GPO line 2. 0h = CPE GPO line 0 1h = CPE GPO line 1 2h = CPE GPO line 2 3h = CPE GPO line 3 4h = MCE GPO line 0 5h = MCE GPO line 1 6h = MCE GPO line 2 7h = MCE GPO line 3 8h = RFE GPO line 0 9h = RFE GPO line 1 Ah = RFE GPO line 2 Bh = RFE GPO line 3 Ch = RAT GPO line 0 Dh = RAT GPO line 1 Eh = RAT GPO line 2 Fh = RAT GPO line 3
7-4	GPOCTL1	R/W	0h	RF Core GPO control bit 1. Selects which signal to output on the RF Core GPO line 1. 0h = CPE GPO line 0 1h = CPE GPO line 1 2h = CPE GPO line 2 3h = CPE GPO line 3 4h = MCE GPO line 0 5h = MCE GPO line 1 6h = MCE GPO line 2 7h = MCE GPO line 3 8h = RFE GPO line 0 9h = RFE GPO line 1 Ah = RFE GPO line 2 Bh = RFE GPO line 3 Ch = RAT GPO line 0 Dh = RAT GPO line 1 Eh = RAT GPO line 2 Fh = RAT GPO line 3

Table 26-209. SYSGPOCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	GPOCTL0	R/W	0h	RF Core GPO control bit 0. Selects which signal to output on the RF Core GPO line 0. 0h = CPE GPO line 0 1h = CPE GPO line 1 2h = CPE GPO line 2 3h = CPE GPO line 3 4h = MCE GPO line 0 5h = MCE GPO line 1 6h = MCE GPO line 2 7h = MCE GPO line 3 8h = RFE GPO line 0 9h = RFE GPO line 1 Ah = RFE GPO line 2 Bh = RFE GPO line 3 Ch = RAT GPO line 0 Dh = RAT GPO line 1 Eh = RAT GPO line 2 Fh = RAT GPO line 3

26.11.3 RFC_PWR Registers

Table 26-210 lists the memory-mapped registers for the RFC_PWR registers. All register offset addresses not listed in Table 26-210 should be considered as reserved locations and the register contents should not be modified.

Table 26-210. RFC_PWR Registers

Offset	Acronym	Register Name	Section
0h	PWMCLKEN	RF Core Power Management and Clock Enable	Section 26.11.3.1

Complex bit access types are encoded to fit into small table cells. Table 26-211 shows the codes that are used for access types in this section.

Table 26-211. RFC_PWR Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

26.11.3.1 PWMCLKEN Register (Offset = 0h) [Reset = 00000001h]

PWMCLKEN is shown in Table 26-212.

Return to the Summary Table.

RF Core Power Management and Clock Enable

Table 26-212. PWMCLKEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Reserved
13	DEMOD	R/W	0h	Enable clock to the Demodulator
12	MOD	R/W	0h	Enable clock to the Modulator
11	IQRAM	R/W	0h	Enable clock to IQ RAM in coherent demodulator
10	RFCTRC	R/W	0h	Enable clock to the RF Core Tracer (RFCTRC) module.
9	FSCA	R/W	0h	Enable clock to the Frequency Synthesizer Calibration Accelerator (FSCA) module.
8	PHA	R/W	0h	Enable clock to the Packet Handling Accelerator (PHA) module.
7	RAT	R/W	0h	Enable clock to the Radio Timer (RAT) module.
6	RFERAM	R/W	0h	Enable clock to the RF Engine RAM module.
5	RFE	R/W	0h	Enable clock to the RF Engine (RFE) module.
4	MDMRAM	R/W	0h	Enable clock to the Modem RAM module.
3	MDM	R/W	0h	Enable clock to the Modem (MDM) module.
2	CPERAM	R/W	0h	Enable clock to the Command and Packet Engine (CPE) RAM module. As part of RF Core initialization, set this bit together with CPE bit to enable CPE to boot.
1	CPE	R/W	0h	Enable processor clock (hclk) to the Command and Packet Engine (CPE). As part of RF Core initialization, set this bit together with CPERAM bit to enable CPE to boot.
0	RFC	R	1h	Enable essential clocks for the RF Core interface. This includes the interconnect, the radio doorbell DBELL command interface, the power management (PWR) clock control module, and bus clock (sclk) for the CPE. To remove possibility of locking yourself out from the RF Core, this bit can not be cleared. If you need to disable all clocks to the RF Core, see the PRCM:RFCCLKG.CLK_EN register.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

DATE	REVISION	NOTES
March 2023	*	Initial Release

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265